Microsoft Dynamics® AX 2012

Deploying customizations across Microsoft Dynamics environments

White Paper

This white paper discusses best practices for deploying customizations from one Microsoft Dynamics AX 2012 environment to another as part of the overall application life cycle management process.

April 2014

microsoft.com/dynamics/ax

Robert Badawy, Senior PM

Send suggestions and comments about this document to adocs@microsoft.com. Please include the white paper title with your feedback.

Microsoft Dynamics

Table of Contents

Overview

As part of the Microsoft Dynamics AX 2012 application life cycle, every implementation should include separate Microsoft Dynamics AX environments, so that you can reliably manage changes and customizations, and control what code ends up running on the production system.

The Microsoft Dynamics AX application life cycle relies on the following environments:

- **Development environments** – Microsoft Dynamics AX development environments typically consist of multiple developers and multiple computers. Development environments are typical of independent software vendors (ISV) and Microsoft partners working on large implementations. Management of development environments is described in the white paper Change management and TFS integration for multi-developer projects and is not within the scope of this document.
- **Test environment** – A test environment is an environment where Microsoft Dynamics AX customizations and solutions, possibly from different vendors, are deployed, integrated, and tested.
- **Staging or pre-production environment** – A staging environment is an environment that is built based on the production environment and typically contains business data from the production system. Microsoft Dynamics AX models and customizations are moved to a staging environment after they have been integrated and tested on a test environment.
- **Production environment** – A production environment is the final environment that customers are using to run their business.

This white paper describes scenarios and best practices for moving Microsoft Dynamics AX 2012 customizations across different environments while minimizing downtime of the production system.

> ***Note:*** *Changes have been made to this paper after it was initially published. For details, see* *Updates since initial publication*.

The following flowchart illustrates the application life cycle.

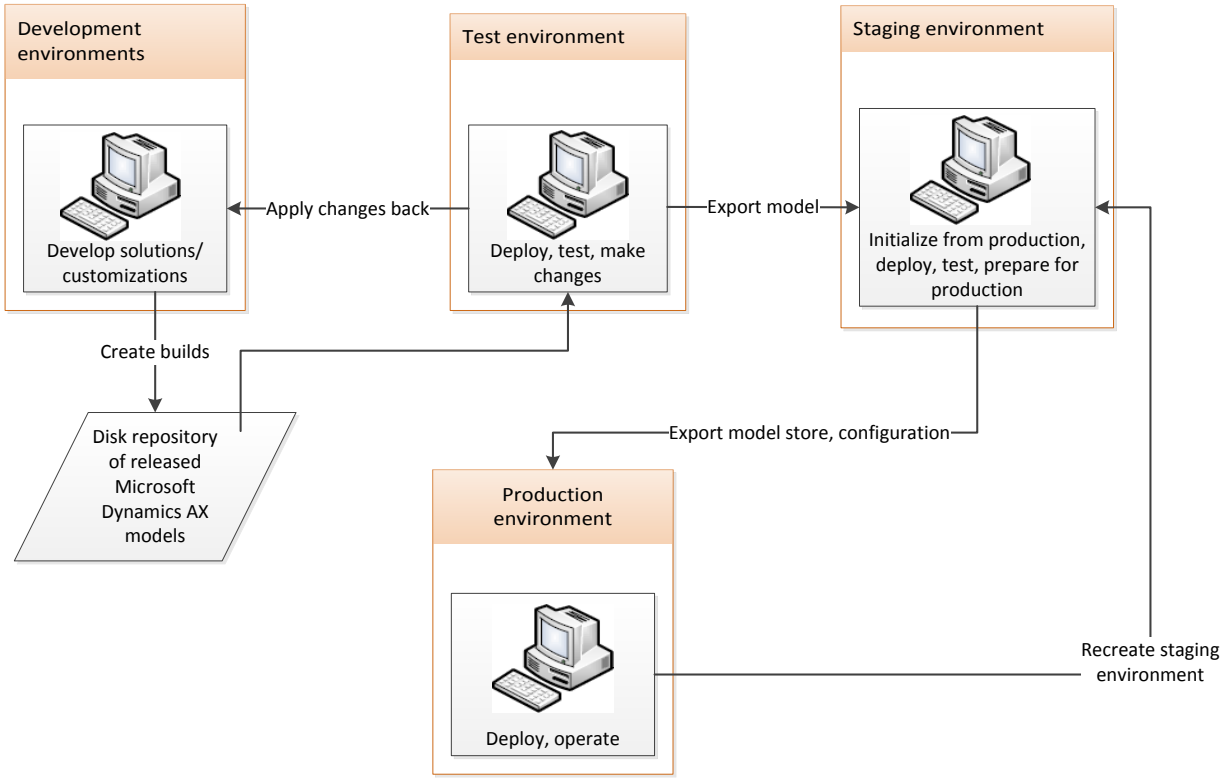


Figure 1 Application life cycle

The preceding flowchart illustrates typical application life cycle management (ALM) scenarios. This document describes many of these scenarios:

- Deploy models on a test environment.
- Initialize a staging environment from the production system.
- Deploy models on the staging environment.
- Prepare the staging system for production.
- Deploy on the production environment.
- Re-base the staging system from production.
- Apply changes back to the development environment.
- Apply XPO fixes on a production system.

We recommend that you first go through the background section of this document to familiarize yourself with key concepts.

Background

This section describes Microsoft Dynamics AX terms and concepts that relate to the deployment of metadata. It provides background information to help you understand the scenarios and best practices covered by this document.

Terminology

The following table explains the Microsoft Dynamics AX terms used in this document.

Term	Definition
Metadata	Information about the properties or structure of data. Metadata is typically represented by elements in the Application Object Tree (AOT). The collection of all metadata elements constitutes the Microsoft Dynamics AX application.
Layer (or application layer)	A single layer of the Microsoft Dynamics AX 2012 application within a model store. Elements in higher layers override elements in lower layers.
Model	A set of elements in a given application layer. Each layer consists of one or more models, of which one is a system-generated layer model.
Model element (or element)	An element of the AOT. For example, a table definition is a model element; a form is also a model element. Any one element in a layer belongs to exactly one model; that is, no element in a layer can belong to two different models.
Model file (.axmodel)	A model that has been exported from a model store. This file is the main vehicle for metadata deployment in Microsoft Dynamics AX 2012.
Model store file (.axmodelstore)	A complete model store that has been exported from the model database. The file includes all metadata, compiled artifacts, CIL code, and security artifacts. The file is used to move the entire application between environments with minimum downtime.
XPO file	A development artifact containing the metadata of one or more elements exported from the AOT. An XPO file is typically used to move metadata between developers.
Model store	A collection of tables in the Microsoft Dynamics AX 2012 model database that house the application metadata. The model store is analogous to the Application Object Data (AOD) file share in Microsoft Dynamics AX 2009.
Common Intermediate Language (CIL)	The Common Intermediate Language instruction set is part of the specification of the Common Language Infrastructure from Microsoft and is more widely known as .NET. An older name for CIL was Microsoft intermediate language (MSIL).
Transactional data	Data that describes business transactions and the state of the business.
Configuration data	Data concerning how Microsoft Dynamics AX 2012 is configured. This data includes the following subcategories: • **Environment** – Computer environment–specific data. An example of environment data is the list of servers running Microsoft SQL Server Reporting Services that Microsoft Dynamics AX 2012 is configured to use. If this data is moved, it needs to be corrected to account for new server locations. • **System** – Parameter settings for Application Object Server (AOS), such as the databases to connect to, or for Enterprise Portal for Microsoft Dynamics AX forms. • **Application** – Number sequences, invoicing profiles, or other data that relates directly to the application.
Master data	The critical data of a business, such as customer, product, location, employee, and asset data. Master data falls into four general groupings: people, things, places, and concepts. It also can be further categorized. For example, within the people grouping, there are customer, employee, and salesperson categories. Within the things grouping, there are product, part, store, and asset categories. Within the concepts grouping, there are categories like contract, warrantee, and licenses categories. Finally, within the places grouping, there are office location and geographic division categories.

Models and the model store

A model is a set of elements in a given layer. Each layer consists of one or more models. Each layer contains at least one system-generated model. Every element in a layer belongs to only one model. In other words, no element can belong to two models in the same layer, and every element must belong to a model.

A model can be exported into a model file (.axmodel) by using either the AXUtil command-line utility or Windows PowerShell cmdlets. To have access to these utilities, you must install the Microsoft Dynamics AX 2012 Management Utilities component by using Setup. These utilities also let you import a model file into a model store. Model files are the principal deployment artifact of Microsoft Dynamics AX metadata.

Models are stored in the model store. The model store is stored in the Microsoft Dynamics AX model database and includes all of the application's metadata – that is, all model elements, including customizations. The model store replaces the AOD files that were used in earlier versions of Microsoft Dynamics AX. Model stores can be exported into a file (.axmodelstore). A model store file includes all metadata, compiled artifacts, CIL code, and security artifacts of a Microsoft Dynamics AX application. The file is used to move the entire application between environments with minimum downtime.

For more details about models and the model store in Microsoft Dynamics AX, see the following TechNet articles: Models, Models, Layers, and the Model Store, and AxUtil and Windows PowerShell Commands for Deploying Models.

Element IDs

When sharing metadata across environments, users must always consider Microsoft Dynamics AX element IDs. Microsoft Dynamics AX elements are associated with corresponding business data based on their element ID. Element IDs are specific to each installation and are sometimes referred to as installation-specific IDs. In other words, the same method, class, or table may have a different element ID in different installations. When two Microsoft Dynamics AX environments share copies of the same business data (such as transaction, configuration, or master data), these environments also need to share the same element IDs. This is typical of staging and production environments.

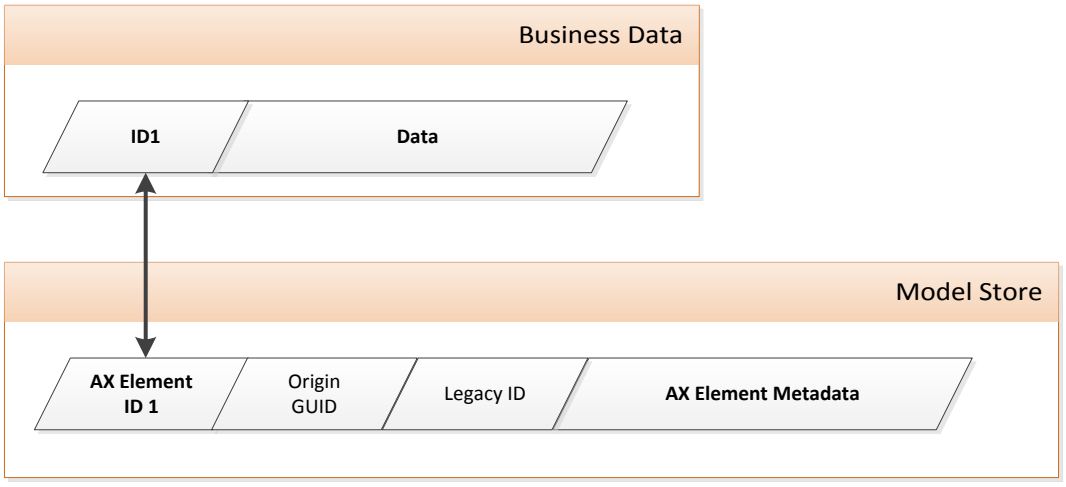


Figure 2 Element IDs

Element IDs are usually assigned when the user installs a model file or imports an XPO file. When the user installs a complete model store, element IDs are equal to the ones stored in the model store file. Some elements also have a **LegacyID** property that is used for backward compatibility with previous Microsoft Dynamics AX versions. **LegacyID** values are also stored in model files and XPO files. Furthermore, all elements with an element ID also have a property named **OriginGUID**. This property is set when an element is created, and it remains static for the lifetime of the element. This property lets you rename or update an element in the future without having to assign it a new element ID.

During the import of model files and XPO files, element IDs are assigned based on the following rules in the following order:

1. If an element already exists that has the same **OriginGUID** value as the imported element, replace the existing element, and reuse its element ID.
2. If an element already exists that has the same **Type**, **Name**, and **ParentID** values as the imported element, replace the existing element, and reuse its ID.
3. Else, if the imported element has a **LegacyID** value > 0, and the same **LegacyID** value is available on the target system, set the element ID to be equal to **LegacyID**.
4. Else, assign a new installation-specific ID from a guaranteed free range that does not collide with any **LegacyID** values:
    1. For fields that are in tables that use table inheritance, use an ID number that is greater than 20000.
    2. For fields that are not in tables that use table inheritance, use an ID number that is greater than 60000.
    3. For indexes, use an ID number that is greater than 60000.
    4. For all other elements, use an ID number that is greater than 1000000.

To maintain installation-specific element IDs when you share metadata between Microsoft Dynamic AX environments, users must strictly follow specific procedures; otherwise, IDs can become randomized, and business data integrity can be affected. Microsoft Dynamics AX environments do not need to share common element IDs unless they need to share business data.

For more details about element IDs, see Maintaining Installation-Specific Element IDs and Element Handles.

Metadata artifacts and features

There are three artifact types that enable the sharing of Microsoft Dynamics AX metadata between environments: XPO files, model files, and model store files. AOD files have been deprecated in Microsoft Dynamics AX 2012 (see Code Upgrade Overview (White paper)).

- **XPO files** are development artifacts. They are typically used to move metadata between development environments.
- **Model files** are deployment artifacts and are the recommended vehicle for distributing solutions to customers, and for deploying builds on a test or staging environment.
- **Model store files** contain the metadata of your entire application, including element IDs and all other element metadata. Model store files are recommended when you deploy a solution from a staging environment to a production environment. When using model store files, you must maintain common element IDs between the source and target systems, as described later in this document.

The following table describes features of these three artifact types.

	XPO files	Model files	Model store files
Installation tool	Microsoft MorphX	AXUtil.exe or Windows PowerShell cmdlets	AXUtil.exe or Windows PowerShell cmdlets
The files can be uninstalled.	No, but elements can be individually deleted.	Yes	No
The files can be signed.	No	Yes	No
Element IDs are preserved in the target model store.	All elements that already exist in the model store preserve their IDs (XPO files don't contain IDs). For new elements, new IDs are generated.	All elements that already exist in the model store preserve their IDs. For new elements, new IDs are generated.	All element IDs on the target system become equal to the IDs stored in the model store file.
Compilation is required after installation.	Yes	Yes	No
CIL generation is required after installation.	Yes	Yes	No

Scenarios

This section describes common ALM scenarios.

Deploy models on a test environment

This section describes the recommended procedure for deploying Microsoft Dynamics AX models to a test environment. In this document, a test system is defined as a system used to do full functional and integration testing of a customer solution that consists of one or more Microsoft Dynamics AX models. It is important that you perform clean deployments on a test system in order to obtain reliable test results.

To deploy customizations on a test system, we recommend that you use model files. You can use either the AXUtil command-line utility or Windows PowerShell to import model files. For more information, see AxUtil and Windows PowerShell Commands for Deploying Models.

Prerequisites and user account privileges

The following components should be available on the test computer:

- Microsoft Dynamics AX client
- Local AOS instance or connectivity to a remote AOS instance
- Connectivity to the database that contains the Microsoft Dynamics AX model store
- Microsoft Dynamics AX Management Utilities

In addition, the user account performing the deployment should have elevated privileges, as described in the User account privileges section of the appendix.

Procedure for deploying models on a test environment

The following flowchart illustrates the procedure for deploying your models on a test environment.

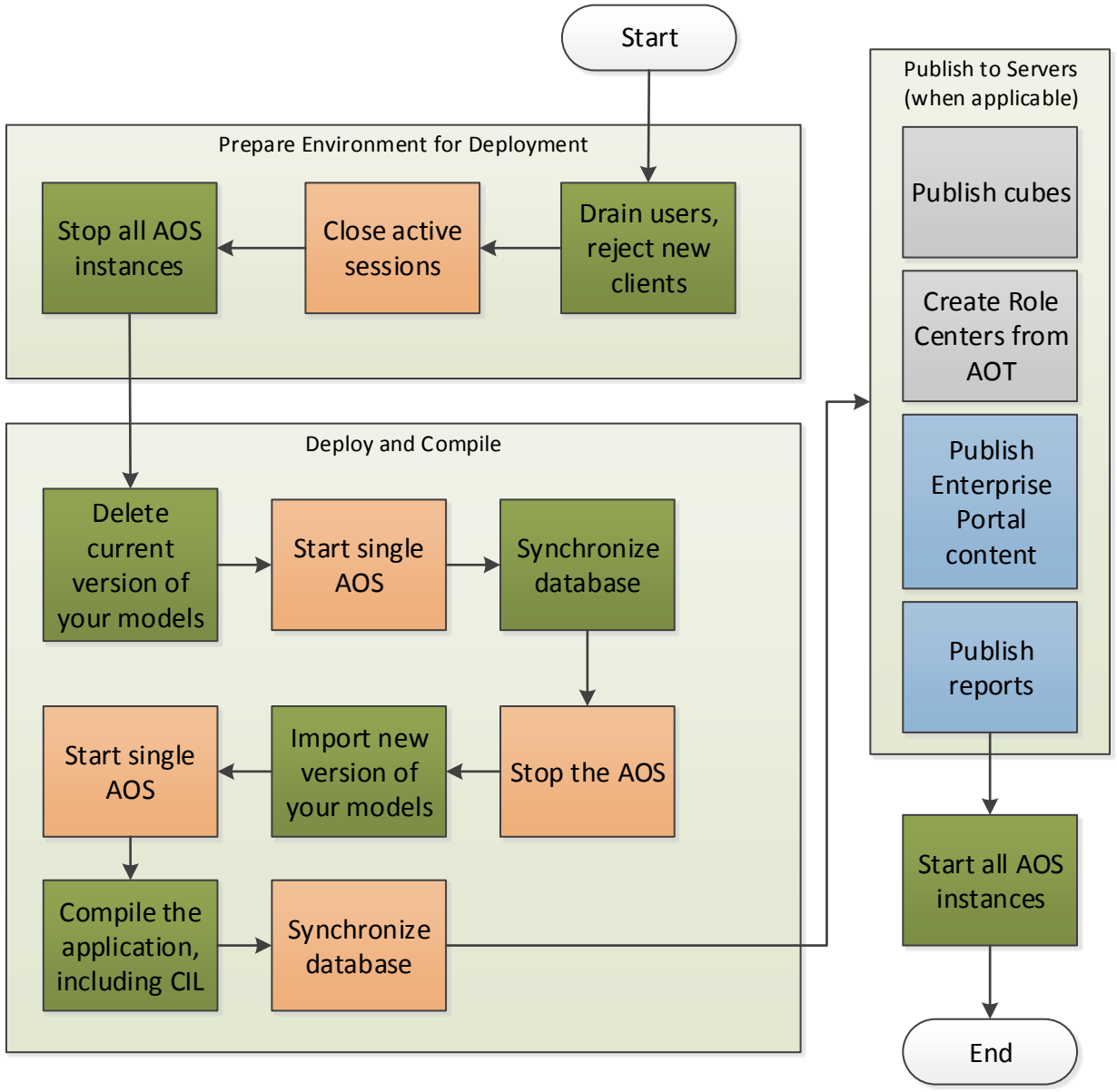


Figure 3 Overview of the recommended deployment procedure on a test environment

The following are the recommended core steps to deploy Microsoft Dynamics AX models on your test system:

Note:

- This procedure deletes the existing instances of your models in order to enable a clean deployment. This means that business data will also be deleted. Element IDs are not maintained when models are deleted; therefore, we recommend that you use test data that you can import after each deployment.
- If you decide not to delete models from a test system, we recommend that you treat it like a staging environment, as described later in this document.

1. Drain users and close active sessions.
2. Stop the AOS instance.
3. Delete the current instance of your models. Use the following commands. For more information, see How To: Remove (Uninstall) a Model.

Tool	Command
AXUtil	AxUtil delete /model:*ModelName*
Windows PowerShell	Uninstall-AXModel

4. Start the AOS instance.
5. Synchronize the business database (that is, perform a database synchronization). At the command line, run the following command:

   ax32.exe startupcmd=synchronize

6. Stop the AOS instance.
7. Import the model files. Import the models in ascending order of layers. An ISV model must be imported before a CUS model. For models that belong to the same layer, the order should be based on inter-model dependencies. For example, if ModelB depends on elements in ModelA, import ModelA first. Use the following commands. For more information, see How to Export and Import a Model.

Tool	Command
AXUtil	Axutil import /file:*ModelName*.axmodel
Windows PowerShell	Install-AXModel

8. Start the AOS instance.
9. Compile the application. This step includes CIL compilation.
10. Synchronize the business database (that is, perform a database synchronization). At the command line, run the following command:

    ax32.exe startupcmd=synchronize

11. Publish the following to servers, if applicable:
    - Publish cubes.
    - Create Role Centers.
    - Publish Enterprise Portal content.
    - Publish reports.
12. Start all other AOS instances, if applicable.

13. Import test data, when applicable, by using **System administration** > **Common** > **Data export/import** > **Import**. Alternatively, at the command line, run the following command:

    ```
    Ax32.exe -startupcmd=autorun_c:\configuration.xml
    ```

    The following is an example of a configuration.xml file where DAT is the company ID you are importing to:

    ```
    <xml version="1.0" ?>
    <AxaptaAutoRun
        exitWhenDone="false"
        version="4.0"
        logFile="c:\temp\AXAutorun.log">
    <DataImport companyId="DAT" file="c:\temp\testdata.dat" />
    </AxaptaAutoRun>
    ```

Models with conflicts

If the installation of a model causes a conflict with another model on the same layer, we recommend that you rerun the import, and that you use the **-Conflict Push** option to push the elements that cause the conflicts to the related update layer. You can then resolve the conflicts. For more information, see How to: Resolve Conflicts After Importing a Model.

When you resolve conflicts, we recommend that you resolve them in the model element itself (on the model layer) rather than in the update layer. You can then export the updated versions of your models to enable a clean deployment on a different system.

In some cases, you may not want to modify the original model, because it may be a signed ISV model. In that case, we recommend that you create your own "conflict resolution" model on a higher layer. After creating a conflict resolution model, you will be able to install the original models with the **–Conflict Overwrite** option and then install your own conflict resolution model. For more information about how to work with models, see Working with Models in AOT.

Make changes on the test environment

If you make changes to your model elements on the test environment to resolve a conflict or fix a bug, finalize your changes as follows:

- If your models were developed in-house on separate development environments, export the changed elements into XPO files. These XPO files should be imported on a development computer (and checked in to source control, if applicable), so that they make it into the next build of your solution.
- Export the changed models into new model files by using the AXUtil **export** command or the Windows PowerShell cmdlet **Export-Axmodel**. For more information, see How to Export and Import a Model. The exported models can then be deployed on your staging environment.

Automation

Microsoft has released Windows PowerShell scripts to help you automate the deployment of a Microsoft Dynamics AX build that can be used to deploy models on a test system. The scripts are described in the *Windows PowerShell scripts: Deploy a Microsoft Dynamics AX build* section of the white paper Change management and TFS integration for multi-developer projects. The scripts are available for download from the Windows PowerShell gallery.

Important: These scripts delete from your model database the current versions of the specified models, as well as all business data affected by the models. Do not use them in environments in which you need to maintain element IDs and business data, such as a staging environment.

Initialize a staging environment from a production system

This section describes the recommended procedure for initializing the model store of the staging environment from the production system. This section assumes that both the staging and production environments have already been set up with a standard Microsoft Dynamics AX installation.

The Microsoft Dynamics AX model store and model store files (.axmodelstore) contain all the metadata elements that define a Microsoft Dynamics AX application. Environments that are initialized from the same model store will have common element IDs and can therefore share copies of the same business data, such as master, configuration, and transactional data. Using model store files to move metadata will also enable minimum downtime when you move from staging to production.

If you use model store files to move metadata between two environments, it is critical to preserve common element IDs between both environments. To preserve common IDs between staging and production, adhere to the following guidelines:

- Initialize the model store of the staging system by importing the model store file from the production environment.
- Do not import new elements (by using XPOs or model files) or create new elements on the production system; otherwise, they will have a different element ID than they have on the staging system.
- Do not delete and re-import models to avoid generating new element IDs. Instead, import models without deleting the existing ones to update the metadata.
- Do not import a model store to the staging environment from a source other than the production system's model store.

If you do not follow these guidelines, you will have to recreate the staging environment from the production environment.

Prerequisites

The user account that you are using to import and export metadata needs certain security privileges on both environments. These are described in the User account privileges section of the appendix.

Procedure for initializing a staging environment from a production environment

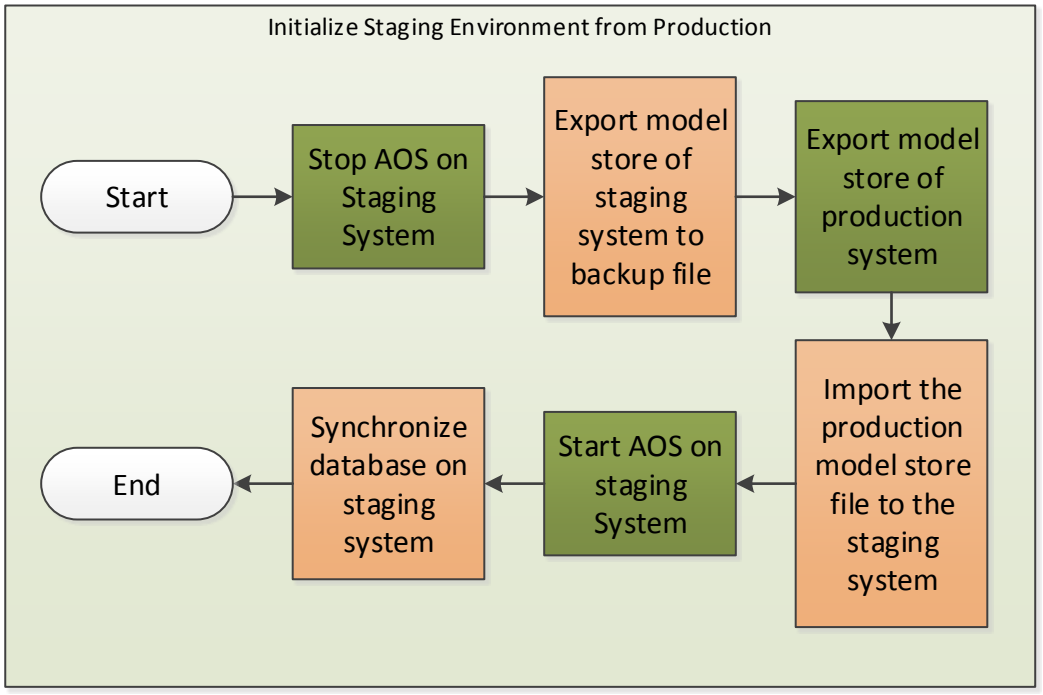


Figure 4 Overview of the recommended procedure for initializing the staging model store

The following are the recommended core steps to initialize the staging model store from the production system:

1. Stop the AOS instance on the staging system.
2. Export the model store from the staging system into a backup file.
3. Export the model store from the production environment into a model store file.
4. Import the production model store file to the staging system.
5. Start the AOS instance on the staging system.
6. Synchronize the database on the staging system.

For more information, see How to: Export and Import a Model Store.

Deploy models on the staging environment

This section describes the recommended procedure for deploying metadata on a staging environment. The procedure is similar to the procedure for deploying models on a test environment described earlier, with one important difference: on a staging environment that shares common element IDs with the production system, it is essential to maintain element IDs. Therefore, when deploying model files, **do not delete the current instance of your models (Skip step 3 of the procedure for deploying models on a test environment, described above)**. Deleting a model and re-importing a new version of it will generate new element IDs, whereas importing a new version of the model on top of the existing one will maintain element IDs.

Prepare to transfer metadata from staging to production

After configuration and testing are completed on the staging system, you must export the metadata and configuration data in preparation for deployment to the production environment.

Procedure for exporting metadata from a staging system

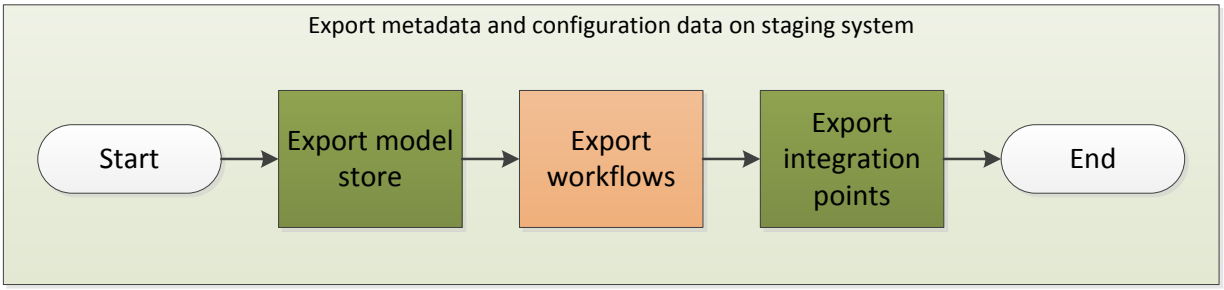


Figure 5 Overview of the export procedure

Re-import all web content into the AOT

Web content in Enterprise Portal may have been modified by using Microsoft SharePoint Server. To ensure that any changes are reapplied to your model store, you must import the web content back into the AOT before exporting the model store.

1. Open the AOT on the source environment.
2. Expand the **Web** > **Web Menu Items** > **URLs** node in the AOT.
3. For each URL that has been modified, right-click the item, and then click **Import Page**.

Export the model store

To export the metadata, use the AXUtil **exportstore** command or the Windows PowerShell **Export-AXModelStor**e cmdlet. These commands migrate the entire metadata store. For more information, see How To: Export and Import a Model Store.

AXUtil

AXUtil exportstore -file:%*AxModelStoreFile*% -s:"%*SourceDataBaseServer*%" -db:"%*ModelDataBaseName*%" -verbose

Example:

AXUtil exportstore -file:myapplication.axmodelstore -s:machine1\MSSQLSERVER -db:axmodeldb

Windows PowerShell

Export-AXModelStore -file "$*AxModelStoreFile*" -Server "$*SourceDataBaseServer*" -Database "$*ModelDataBaseName*" -Details -verbose

Export workflows

Workflows must be exported and imported by using the workflow editor. By using this method, users can select workflows from the workflow list page and export them to XML files.

Note: This step assumes that the staging and the target (production) environment have common element IDs; otherwise, additional steps may be needed on the target system to correct the object IDs, depending on the complexity of the workflows.

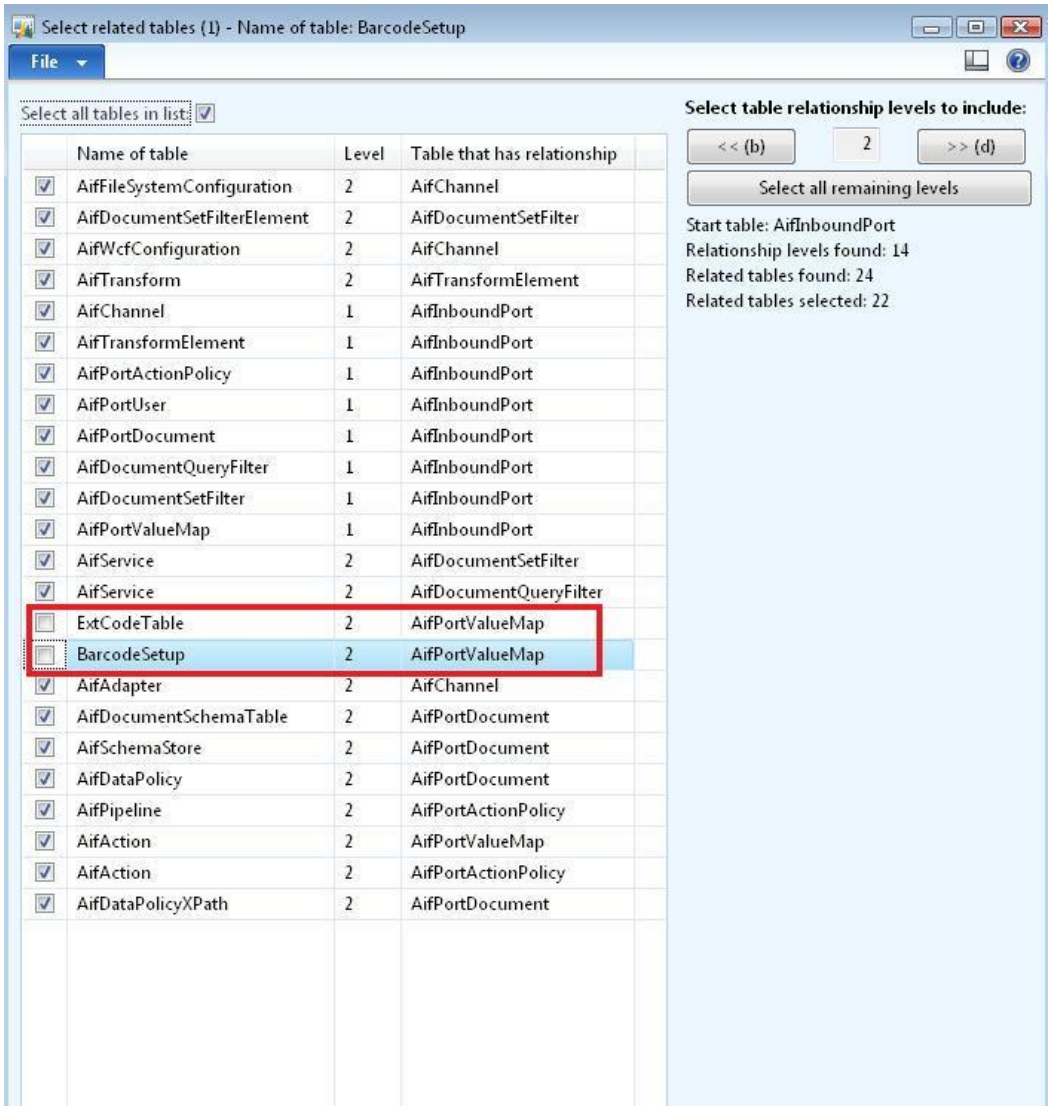
1. Open the Microsoft Dynamics AX client on the source environment.
2. Open the workflow list page for the workflow that you want to export (for example, click **Travel and expense** > **Setup** > **Travel and expense workflows**).
3. Select the workflow that you want to export, and then click **Versions**.
4. In the **Workflow version** dialog box, select a version of the workflow, and then click **Export** on the toolbar.
5. In the **Export to** dialog box, type the file name, and then click **OK**.
6. Repeat steps 1 through 5 for each workflow that you need to export.

Export enhanced integration ports

Basic ports within Microsoft Dynamics AX 2012 are stored as metadata and will be automatically deployed when a model store is deployed. Enhanced integration ports are stored as data and must be exported by using the data export/import utility. The first time a deployment is performed, you will need to create definition groups for the inbound and outbound ports. These definition groups can be reused for future deployments.

1. Click **System administration** > **Common** > **Data export/import** > **Definition groups**. Click **New**.
2. Enter a name and description for the definition group.
3. Click **Clear** to clear all values on the **Options** and **Include table groups** tabs, and then click **OK**.
4. Click **Select tables**. In the **Name of table** list, type or select **AifInboundPort**.
5. Select **Apply criteria** and **Specify related tables** to make the **Export criteria** and **Select related tables** buttons available.
6. Click **Select related tables**. In the **Select related tables** form, set the value of **Select table relationship levels to include** to **2**.

7. Clear the check boxes for the **ExtCodeTable** and **BarcodeSetup** tables.



8. Click **Select all remaining levels**, and then click **Close**.

Name of table	Level	Table that has relationship
AifFileSystemConfiguration	2	AifChannel
AifDocumentSetFilterElement	2	AifDocumentSetFilter
AifWcfConfiguration	2	AifChannel
AifSchemaStore	6	AifXsltRepository
AifLookupEntry	6	AifLookupTable
AifXsltRepository	5	AifXmlTransformConfig
AifLookupTable	5	AifValueSubstitutionConfig
AifDocumentField	5	AifValueSubstitutionConfig
AifValueSubstitutionCompon...	4	AifPipelineComponent
AifXmlTransformConfig	4	AifPipelineComponent
AifValueSubstitutionConfig	4	AifPipelineComponent
AifDocumentField	4	AifDataPolicyLegalValue
AifAppShareFile	3	AifTransform
AifDataPolicyLegalValue	3	AifDataPolicy
AifSchemaStore	3	AifDocumentSchemaTable
AifDocumentField	3	AifDocumentSchemaTable
AifDocumentField	3	AifDataPolicy
AifPipelineComponent	3	AifPipeline
AifAction	3	AifService
AifTransform	2	AifTransformElement
AifChannel	1	AifInboundPort
AifTransformElement	1	AifInboundPort
AifPortActionPolicy	1	AifInboundPort
AifPortUser	1	AifInboundPort
AifPortDocument	1	AifInboundPort
AifDocumentQueryFilter	1	AifInboundPort
AifDocumentSetFilter	1	AifInboundPort
AifPortValueMap	1	AifInboundPort
AifService	2	AifDocumentSetFilter
AifService	2	AifDocumentQueryFilter
ExtCodeTable	2	AifPortValueMap
BarcodeSetup	2	AifPortValueMap
AifAdapter	2	AifChannel
AifDocumentSchemaTable	2	AifPortDocument
AifSchemaStore	2	AifPortDocument
AifDataPolicy	2	AifPortDocument
AifPipeline	2	AifPortActionPolicy
AifAction	2	AifPortValueMap
AifAction	2	AifPortActionPolicy
AifDataPolicyXPath	2	AifPortDocument

9. Click **Export criteria**. In the **Criteria** field, select the name **AifInboundPort**, and then click **OK**.

10. Repeat steps 1 through 9 to create a second definition group for outbound ports. Repeat the same process, but use **AifOutboundPort** as the root table, and set the export criteria to filter on the name **AifOutboundPort**.
11. To export the data, click **System administration** > **Common** > **Data export/import** > **Export to**.
12. On the **General** tab, select the definition group that you want to use for the export.
13. Enter the name and location of the file that you want to export the data to on a local or network share, and then click **OK**.

Deploy metadata on the production environment

This section describes the recommended procedure for deploying metadata and configuration data on a production environment. It assumes that the production environment and the staging environment share common element IDs.

Prerequisites

Before you deploy metadata, the following components should be available on the production computer:

- Microsoft Dynamics AX client
- Local AOS instance or connectivity to a remote AOS instance
- Connectivity to the database that contains the Microsoft Dynamics AX model store
- Microsoft Dynamics AX Management Utilities

The user account that you are using to install the model store needs elevated security privileges. These are described in the User account privileges section of the appendix.

Procedure for deploying metadata on the production environment

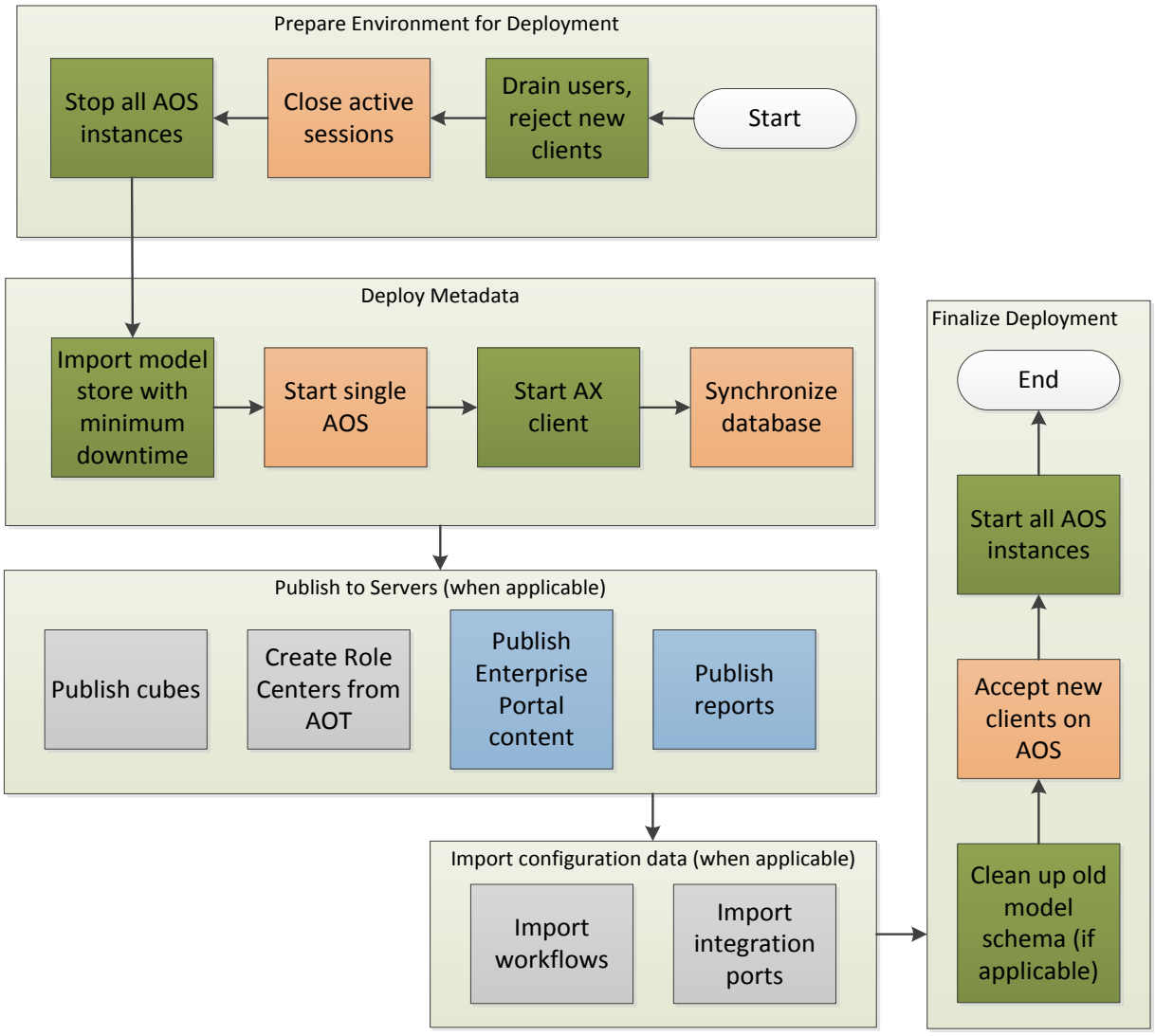


Figure 6 Overview of the recommended deployment procedure on the production system

Note: The recommended procedure uses model store files; however, it is also possible to deploy individual model files on the production environment. If you use model files to deploy metadata to a production system, remember the following points:

- More downtime will be necessary.
- Element IDs may be different. In this case, the production environment may not be able to share all business data with the staging environment.
- Follow the deployment procedure on staging environments described earlier in this document.
- If you import workflows, you will need to make some corrections (described in the section Correct workflows).

Import a model store with minimal downtime

Import the model store that was exported from the staging system into the production environment. To reduce downtime, we recommend that you import the metadata into a new schema next to the old one, and then switch to the active schema. This methodology lets users continue using the system until the AOS instance needs to be restarted so that the new schema can be applied. The steps for importing the new schema can be done before users are drained. For more information, see How to: Export and Import a Model Store.

Import a model store by using AXUtil

1. Run the schema command to create a new schema.

   AXUtil schema /schemaname:%*NewSchema*% /db:%TargetModelDatabase%

2. Import the model store into the temporary schema.

   AXUtil importstore /file:%*AxModelStoreFile*% /schemaname:%*NewSchema*% /db:%*TargetModelDatabase*%

3. When all users are out of the system, stop the AOS instance. At the command line, run the following command:

   sc \\%AOSServer% stop %AOSInstance%

   Example:

   sc \\computername stop AOS60$01

4. Apply the changes to the model store to move from the temporary schema to the **dbo** schema.

   AXUtil importstore /apply:%*NewSchema*% /db:%*TargetModelDatabase*% /verbose

Import a model store by using Windows PowerShell

1. Run the schema command to create a new schema.

   Initialize-AXModelStore -SchemaName "$*NewSchema*" -Database "$*TargetModelDatabase*" -Details

2. Import the model store into the temporary schema.

   Import-AXModelStore -File "$AxModelStoreFile" -SchemaName "$NewSchema" -Database "$TargetModelDatabase" -Details

3. When all users are out of the system, stop the AOS instance.

   Set-Service -computername $*AOSServer* -name $*AOSInstance* -status stopped

4. Apply the changes to the model store to move from the temporary schema to the **dbo** schema.

   Import-AXModelStore -Apply "$*NewSchema*" -Database "$*TargetModelDatabase*" -Details

Publish to other servers

The installation is not completed unless the metadata for cubes, Enterprise Portal, and report servers has also been deployed. For more information, see the following procedures in the appendix:

- Publish cubes
- Create Role Centers
- Publish Enterprise Portal content
- Publish reports

Import workflows

Workflows are imported by using the workflow editor in the Microsoft Dynamics AX client, as follows:

1. Open the Microsoft Dynamics AX 2012 client.
2. Open the workflow list page for the workflow that you are going to import. For example, click **Travel and expense** > **Setup** > **Travel and expense workflows**.
3. Click **Import**.
4. In the **Import** dialog box, select the file name, and then click **OK**.
5. Repeat steps 1 through 4 for all workflows that you want to import.

Correct workflows

If you deployed the metadata by using models files or XPOs instead of the model store file, the element IDs may have changed, and the workflows may need to be corrected.

1. If you have a parent workflow that contains sub-workflows, you need to re-link the sub-workflow to the parent workflow:
    1. Open the parent workflow in the workflow editor.
    2. Select the sub-workflow element.
    3. In the **Properties** window, select the appropriate sub-workflow and its field.
    4. Save the parent workflow.
2. If you have a header workflow that contains line item workflows, you need to re-link line item workflows to the header workflow:
    1. Open the header workflow in the workflow editor.
    2. Select the line-item workflow element.
    3. In the **Properties** window, select the appropriate line-item workflow.
    4. Save the header workflow.
3. Repeat steps 1 and 2 for all workflows that you want to import.

Import enhanced integration ports

Import the enhanced integration ports by going to **System administration** > **Common** > **Data export/import** > **Import** and selecting the .dat file that you previously created.

Finalize the deployment

After the customizations are verified as working on the production environment, you must finalize the deployment.

1. Clean up the old metadata schema.

    If you used the **BackupSchema** parameter when you imported the model store, you created a backup of the initial model store. When you are satisfied with the new model store, you should delete the backup schema.

    AXUtil

    AXUtil schema /drop:%*OldSchema*% /db:%*TargetModelDatabase*%

    Windows PowerShell

    Initialize-AXModelStore -Drop "$*OldSchema*" -Database "$*TargetModelDatabase*"

2. Set the AOS instance to accept new clients. For more information, see the following procedure in the appendix: Set the AOS instance to accept new client connections.
3. Restart all production AOS instances.

Recreate the staging environment from the production system

When the staging and production systems do not share common element IDs, model store files cannot be used to deploy metadata. You must recreate the staging system from the production system to enable model store file deployment.

Even if the staging environment was originally based on the model store of the production system, scenarios such as the following may lead to a loss of element ID commonality:

- Importing new elements (by using XPOs or model files) or creating new elements on the production system
- Deleting and reinstalling models, which will regenerate new element IDs
- Installing a model store on the staging environment from a source other than the original model store

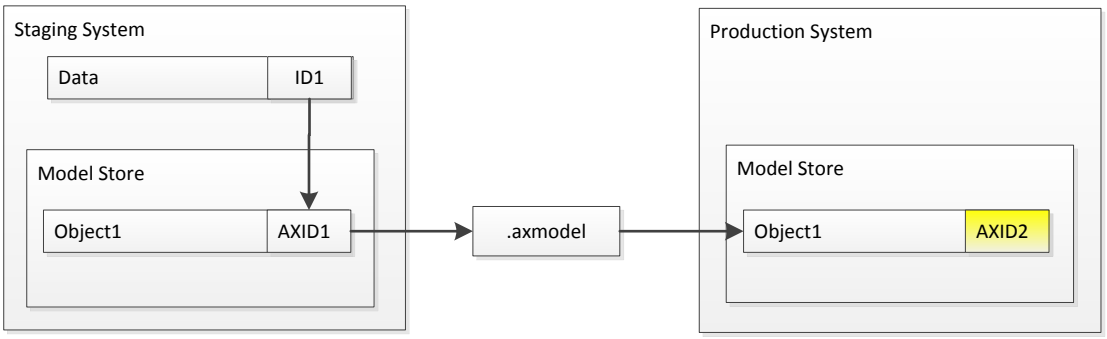
When the two environments don't have common element IDs, importing a model store on the production environment may cause ID conflicts or errors during database synchronization. Follow the next procedure to recover from this condition.

Example: Element ID conflict

The following illustrates one way that element ID conflicts can arise:

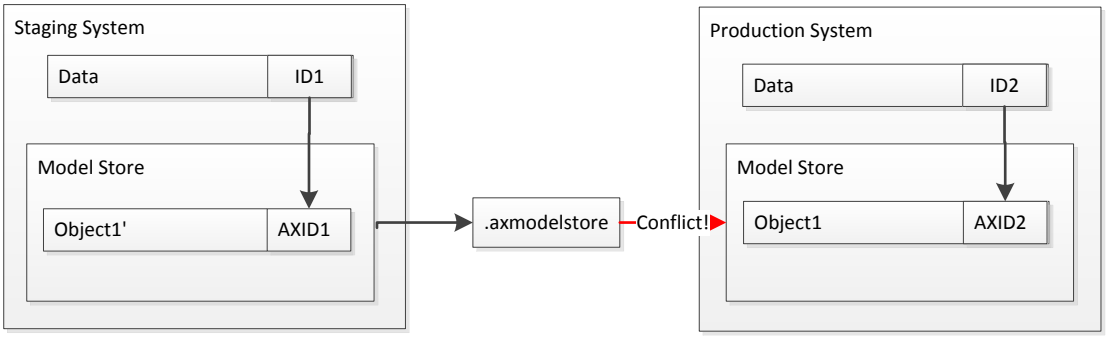
1. First, initialize the staging system model store from the production system.
2. On the staging environment, install some customizations that include new elements (Object1).

3. On the production system, import the model that contains Object1. Because this is a new object that does not already exist in the production system, Microsoft Dynamics AX may give the object an ID that is different from the ID in the source system.



4. Someone runs the production system and adds some data, which creates transactional records that reference AXID2.

5. Make another customization on the staging system, and deploy the entire model store on production for minimal downtime. To do this, import the model store file, which includes the IDs.



Unfortunately, we will run into conflicts, because the IDs in the two systems are different.

Prerequisites

Before you deploy metadata, the following components should be available on the staging computer:

- Microsoft Dynamics AX client
- Local AOS instance or connectivity to a remote AOS instance
- Connectivity to the database that contains the Microsoft Dynamics AX model store
- Microsoft Dynamics AX Management Utilities

The user account that you are using to install and export metadata needs elevated security privileges on both environments. These are described in the User account privileges section of the appendix.

Procedure for recreating the staging environment from the production environment

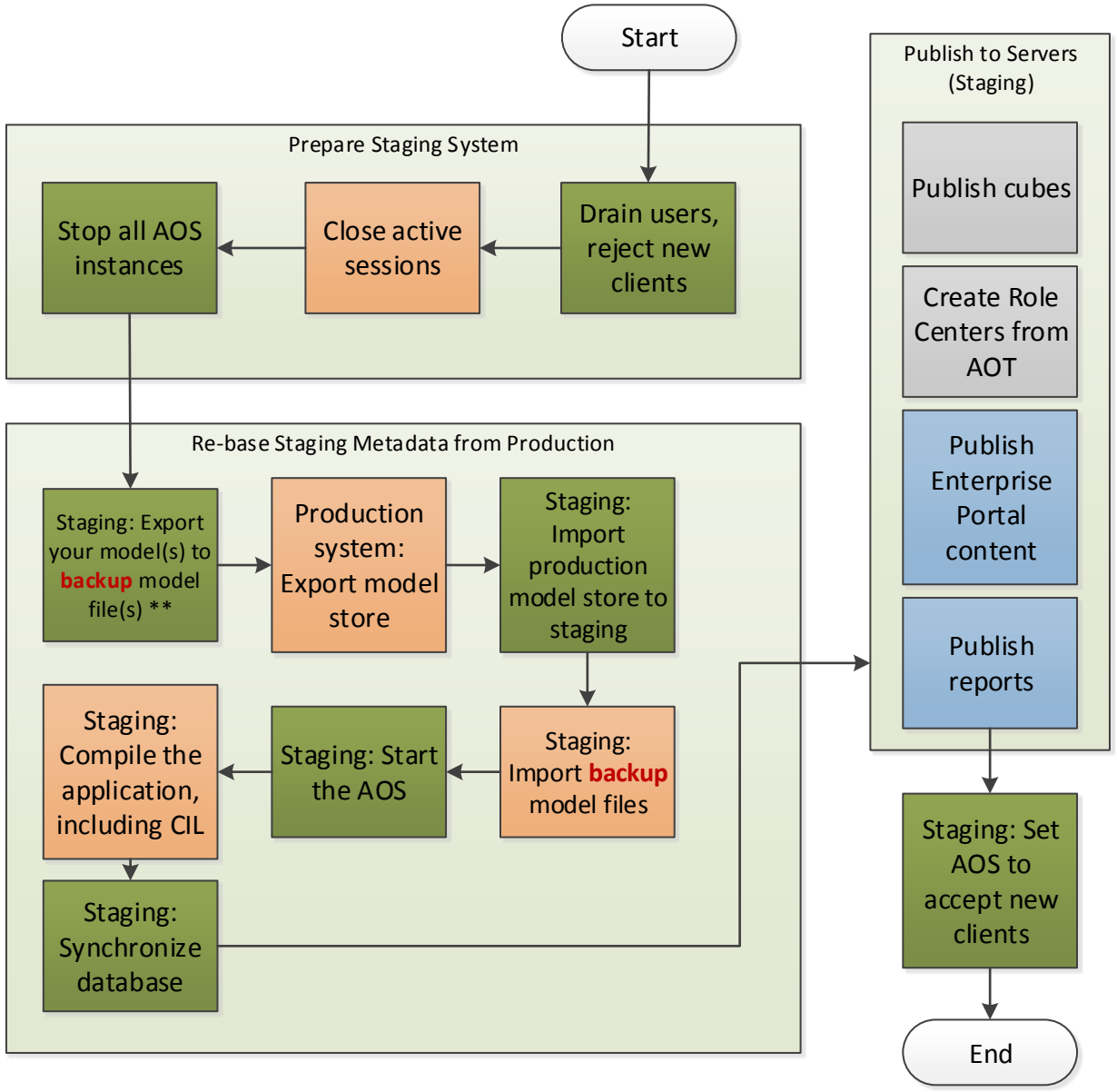


Figure 7 Overview of the recommended procedure for re-basing the staging system from production

Note: This procedure assumes that the version of the application on the staging environment is newer than the one on production, which requires backing up then re-importing models.

1. Drain users, and close the active session on the staging system. For details, see the following procedure in the appendix: Drain active users and close active sessions.
2. On the staging system, stop the AOS instance.
3. On staging system, export your custom models to model files. For more information, see How to export and import a model.
4. On the production system, export the model store. For more information, see How to export and import a model store.
5. On the staging system, import model store file from step 4.
6. On the staging system, import model files from step 3.
7. Start the AOS instance.
8. On the staging system, compile the application, including CIL.
9. On the staging system, run database synchronization.
10. On the staging system, publish to servers, if applicable. For more information, see the following procedures in the appendix:
    - Publish cubes
    - Create Role Centers
    - Publish Enterprise Portal content
    - Publish reports
11. On the staging system, start all other AOS instances.

Apply changes back to the development environment

If you make metadata changes directly to the staging or production system, we recommend that you export these changes as XPO files and load them to the development environment.

1. Export your changes to an XPO file.
2. On a Microsoft Dynamics AX development environment, start the Microsoft Dynamics AX client in the desired layer, and select the appropriate model as your current model.
3. Load the XPO from step 1.
4. If your Microsoft Dynamics AX installation is integrated with a version control system such as Team Foundation Server (TFS), follow these steps:
    - If the XPO contains new elements, add these elements to version control.
    - Check all changed elements in to version control.

Apply XPO files to a production environment

It is possible to use XPO files to quickly apply fixes from a partner or customer developer to a Microsoft Dynamics AX staging or production environment. When you do so, it is important to consider the following information.

Element ID commonality

When you load an XPO file that contains new elements on a Microsoft Dynamics AX environment (staging or production), the new elements are assigned new element IDs. If the same XPO is applied on another environment, the two environments will have different IDs for the same element. When the staging and production environments have different IDs for common elements, model store files cannot reliably be used as a deployment vehicle between the two environments. In this case, you may need to re-create the staging system based on the model store of the production environment.

Layers and customized elements

When an XPO is loaded, all elements in the XPO are applied to the current layer (specified by the Microsoft Dynamics AX client configuration). However, if the XPO contains elements that are customized in a higher layer than the current layer, the changes are applied to the highest layer, and therefore the elements will end up in a model that may be different from what the user intended.

For example, Table1 exists in the SYS layer and is also customized in the CUS layer. If you start Microsoft Dynamics AX in the ISV layer and load an XPO that adds a new field to Table1, the new field is applied to the CUS layer of Table1, not to the ISV layer.

Therefore, we do not recommend that you use XPO files when applying fixes to middle layers, such as the ISV layer. Use model files instead.

Migrating security between environments

Changes to the behavior of roles, duties, and privileges also change the metadata stored in the AOT. These changes are reflected in the model that the person making the change is working in.

Important: The metadata stored in the AOT contains the information about the behavior of security roles, but user-to-role mapping information is stored in the business database. To move user-to-role mapping information, you must use Microsoft Dynamics AX data import/export to migrate the UserInfo and SecurityUserRole tables from the business database.

People in the following roles are most likely to modify security:

- Security administrators with the responsibility for making changes to the security objects by using the Microsoft Dynamics AX client.

  Changes made by the security administrator are most likely to be stored in the USR model.

- Microsoft Dynamics AX developers.

  Changes made by developers are most likely to be stored in the CUS or VAR models.

It is essential that the security administrator and developer collaborate to make security changes.

Scenario 1: Move security changes from production to staging

We recommend that you migrate security changes from the production to the staging environment before making further changes in the staging environment.

Follow the steps in the section Recreate the staging environment from the production system.

The advantage of this approach is that all the changes made from the production environment are brought over to staging and can be tested, compiled etc. in the staging environment before they are reapplied. Note that changes that were made in the production environment by an administrator cannot be uniquely identified in the staging environment.

Scenario 2: Move security changes from staging to production

To ensure that recent security changes made by the administrator will not be lost in the production system, we recommend that you back up the changes from the production environment, apply the changes from the staging environment to the production environment, and then recent security changes to the production environment

When you deploy on production, instead of the step where you import the staging model store, do the following:

1. Back up the security changes from the production environment by exporting the USR model from the USR layer of the production system (or the layer in which the security administrator made the changes).
2. Import the model store of the staging system that contains changes to security artifacts.
3. Reapply the original production security changes by importing the USR model file from step 1.
4. Recompile the production environment.

Recommended resources

White papers

- Code Upgrade Overview (White paper)
- Change management and TFS integration for multi-developer projects (White paper)
- Data Migration for Microsoft Dynamics AX 2012 (White paper)

Technical reference

- Models, Layers and the Model Store
- Deploying Customizations and Solutions by using Models and the Model Store
- Working with Models in the AOT

Updates since initial publication

The following table lists changes made to this document after it was initially published.

Date	Change
April 2014	Updated text in several flowcharts to make them easier to follow. Updated steps in the Procedure for recreating the staging environment from the production environment section to correct an error.
February 2013	Updated the section Models with conflicts: • Defined a test system and clarified some recommendations in the deployment procedure in a test environment. • Clarified the difference between the model database and the business database because they were split in Microsoft Dynamics AX 2012 R2. Added the section Migrating security between environments.
October 2011	Updated to be organized around Application Lifecycle Management scenarios. Added section describing how to apply XPO files to production environment.
July 2011	Initial publication

Appendix

Stop/start the AOS instance

- To start the AOS instance from the command line, run the following command:

  sc \\AOShostname start %*AOSinstance*%

  Example:

  sc \\computername start AOS60$01

- To stop the AOS instance from the command line, run the following command:

  sc \\AOShostname stop %*AOSinstance*%

- To start the AOS service from Windows PowerShell, run the following cmdlet:

  Set-Service -computername %*AOSHostname*% -name %*AOSInstance*% -status running

- To stop the AOS service from Windows PowerShell, run the following cmdlet:

  Set-Service -computername %*AOSHostname*% -name %*AOSInstance*% -status stopped

User account privileges

Before you deploy models, the following components should be available on the computer:

- Microsoft Dynamics AX client
- Local AOS instance or connectivity to a remote AOS instance
- Connectivity to the database that contains the Microsoft Dynamics AX model store
- Microsoft Dynamics AX Management Utilities

We recommend that you create a dedicated account for performing deployments, because deployment accounts require elevated privileges. The Windows account that you use to perform the deployment must meet the following requirements:

- It must be able to start and run the Microsoft Dynamics AX client.
- It must have permissions to start and stop every AOS service on the Microsoft Dynamics AX environment.
- It must be able to execute AXUtil.exe or the AXUtilLib.PowerShell cmdlet module. This requires that it have read and execute access to the folder <*Microsoft Dynamics AX installation folder*>\ManagementUtilities.
- It must have administrative permissions on the local computer.
- It must be a member of the Microsoft Dynamics AX role **System administrator**.
- If you are working with Enterprise Portal, the account must be able to run on the server that runs Enterprise Portal. The account must have the following rights:
  - Microsoft SharePoint Server farm administrator
  - Membership in the local Administrators group on the server that runs Enterprise Portal
- If you are working with Reporting Services, the account must have the following rights on the server that runs Reporting Services:
  - System Administrator rights in Reporting Services
  - Membership in the local Administrators group on the server that runs Reporting Services

- It must be given privileges in the Microsoft Dynamics AX model database by running the AXUtil **grant** command or the Windows PowerShell **Grant-AXModelStore** cmdlet.

  Note: If you are running under the same account as the AOS Service account, you do not need to run the **grant** command. We recommend that you use a different account to deploy customizations, and that you run the **grant** command for it.

Run AXUtil grant or Windows PowerShell Grant-AXModelStore

To run the AXUtil **grant** command against a model database, the deployment account must be a member of the following SQL Server roles on the Microsoft Dynamics AX model database server:

- Server role **securityadmin**
- Database role **accessadmin**

To grant these SQL Server privileges, you must be a member of the **SysAdmin** server role on the database instance. You can have a SQL Server system administrator grant you the privileges. If you are a member of the **SysAdmin** server role, use SQL Server Management Studio or the following commands to grant the rights to the account:

```
osql -S %SQLAdministrator% -E -Q "EXEC sp_addsrvrolemember @loginame =
N'%DeploymentAccount%', @rolename = N'securityadmin'" -d master
```

```
osql -S %SQLAdministrator% -E -Q "EXEC sp_addrolemember N'db_accessadmin',
N'%DeploymentAccount%'" -d %DatabaseName%
```

After the account has the correct privileges, you can execute the AXUtil **grant** command or the Windows PowerShell **Grant-AXModelStore** cmdlet.

To grant access by using AXUtil

1. Open a Command Prompt window, and navigate to the <*Microsoft Dynamics AX installation folder*>\ManagementUtilities folder.
2. Run the following command:

   AXUtil grant -aosaccount::%*account*% -s:%SourceDataBaseServer% -db:%*SourceModelDataBaseName*%

   Here, %*account* is the account that you are granting privileges to. The **-s** and **-db** parameters are included in the command to clearly specify which database and server you are granting access to.

   Example:

   AXUtil grant -aosaccount:: domainname\alias -s:hostname\SQLInstance -db:AXModelDB

To get help with AXUtil commands

- Run the following command:

  AXUtil.exe /?

To grant access by using Windows PowerShell

1. On the **Start** menu, point to **Administrative Tools**, and then click **Microsoft Dynamics AX 2012 Management Shell**.
2. Execute the following cmdlet:

   Grant-AXModelStore -aosaccount %*account*% -Server "$*SourceDataBaseServer*" -Database:"$*SourceModelDataBaseName*"

   Here, *%account%* is the account that you are granting privileges to. The **-Server** and **-Database** parameters can be included in the command to clearly specify which database server and model store database name you are granting access to.

Example:

Grant-AXModelStore -aosaccount domainname\alias -Server hostname -Database AXModelDB

To get help with Windows PowerShell cmdlets

- Execute the following cmdlet:

  Get-Help <*CmdletName*> -full

Publish cubes

To deploy cubes, follow these steps:

1. Click **Tools** > **Business Analysis** > **SQL Server Analysis Wizard**.
2. Click **Next**.
3. Click **Deploy**.
4. Select a project from the AOT drop-down menu, and then click **Next**.
5. Click **Deploy the Project**, click **Create New Database**, and then click **Next**.
6. The deployment window opens, which may take a few minutes. Click **Next**.

Create Role Centers

To create Role Centers, follow these steps:

1. Open the Microsoft Dynamics AX 2012 client.
2. Click **System administration**, and then click **User profiles**.
3. Click **File** > **New** to create a new role.
4. Enter a profile ID and description for the role.
5. Select the Role Center from the drop-down menu.
6. Click **Add user** or **Bulk add users**.
7. Select the user IDs to add, and then click **OK**.

Publish Enterprise Portal content

Enterprise Portal web content can be deployed either from the AOT or programmatically by using the AxUpdatePortal.exe command-line utility, which is installed together with the Management Utilities by using Setup.

- All web content can be updated at the same time by using the following command:

  AxUpdatePortal.exe -updateall -websiteurl %*SiteUrl*%

- Individual web content items can be deployed by specifying their location in the AOT:

  AxUpdatePortal.exe -updatewebcomponent -treenodepath %*TreeNodePath*% -websiteurl %*SiteUrl*%

- Additionally, proxies and DLLs can be deployed by using the following command:

  AXUpdatePortal.exe -proxies -websiteurl %*SiteUrl*%

Publish reports

Reports are a part of the model store. To deploy reports, you first import the model store or model file containing the reports, and then deploy the reports by using Windows PowerShell cmdlets or scripts.

Deploy reports by using Publish-AxReport

Run the following Windows PowerShell cmdlet:

Publish-AxReport -Id $*ConfigId* -ReportName "*ReportName*"

Here, $*ConfigID* refers to a configuration ID that was defined in Microsoft Dynamics AX. To view these configuration IDs, open Microsoft Dynamics AX and then open the Report servers form. (Click **System administration** > **Setup** > **Business intelligence** > **Reporting Services** > **Report servers**.) To deploy reports to the desired Reporting Services instance, enter the configuration ID that is associated with that instance.

ReportName is the name of the report to deploy.

To deploy all reports, you can substitute * for *ReportName* in the preceding command.

Deploy reports by running the AxDeploy Windows PowerShell script

The AxDeployReports.ps1 script can be used to deploy all reports from the AOT to the report servers. To run the script, navigate to the <*Microsoft Dynamics AX installation folder*>\ManagementUtilities folder, open a Command Prompt window, and run the following command:

AxDeployReports.ps1 <ConfigID> <LogFilePath>

Here, <*ConfigID*> specifies the Reporting Services instance to use (as in previous section), and <*LogFilePath*> specifies the location where the logs should be stored.

Drain active users and close active sessions

To prepare for AOS downtime, start the process of draining active users by rejecting new client connections. For more information, see Drain users from an AOS.

1. Click **System administration** > **Common** > **Users** > **Online users**.
2. On the **Server instances** tab, select the AOS instance that you want to perform maintenance on.
3. Click **Reject new clients**.
4. When you are prompted, click **OK** to stop the AOS instance from accepting new client connections.

   After 5 minutes, all users receive a message informing them that they must save their work, because the administrator is shutting down the AOS instance. No new client connections are accepted during this time. The server forces client sessions to close after they have been idle for 2 minutes. Client sessions for administrators are never closed. When the number of clients that are connected to the AOS instance is displayed as 0 (zero), you can perform maintenance on the server.
5. Stop load balancers, if applicable. This will prevent new service clients from connecting.
6. Restart all Reporting Services servers. This will force the Reporting Services servers to recycle their idle connections.
7. Open a page on the Enterprise Portal site. This will force SharePoint Server to recycle its idle connections.

It is important to note that ending the sessions for active users will cause them to lose data. Be sure that you have followed the preceding steps before you stop any active client sessions.

To close active client sessions for each AOS instance, use the **Online users** form.

1. Click **System administration** > **Common** > **Users** > **Online users**.
2. On the **Client instances** tab, select the active sessions.
3. Click **End sessions**.

Be sure also to end the sessions for web users.

Set the AOS instance to accept new client connections

To set the AOS instance to accept new client connections, follow these steps:

1. Click **System administration** > **Common** > **Users** > **Online users**.
2. On the **Server instances** tab, select the AOS instance that you want to perform maintenance on.
3. Click **Accept new clients**.

Starting the AOS instance in "Single-user" mode

Although an AOS instance in Microsoft Dynamics AX 2012 can be set to reject new incoming client sessions, this setting is reset when the AOS instance restarts. During maintenance, it is important to prevent other users from connecting to the system. You can use AOS and client configurations to prevent other users from connecting.

Method 1: Create a maintenance configuration

You can create a separate AOS and client configuration for use by the deployment administrator.

1. Use the server configuration utility to create a copy of the active configuration.
2. Change the TCP/IP and WSDL ports to different values.
3. Use the client configuration utility to create a copy of the active client configuration.
4. Change the TCP/IP and WSDL ports to the values used for the server configuration.

During maintenance, change the AOS and maintenance client to use these configurations. This will prevent users from connecting to the maintenance AOS instance.

Method 2: Disable client configurations for your enterprise

If you have configured the client across your enterprise to start by using configuration files that are stored on a central file share, you can revoke read access to these files for everyone except the deployment administrator. This will prevent users from starting the Microsoft Dynamics AX 2012 client.

Microsoft Dynamics is a line of integrated, adaptable business management solutions that enables you and your people to make business decisions with greater confidence. Microsoft Dynamics works like and with familiar Microsoft software, automating and streamlining financial, customer relationship and supply chain processes in a way that helps you drive business success.

U.S. and Canada Toll Free 1-888-477-7989

Worldwide +1-701-281-6500

www.microsoft.com/dynamics