Microsoft Dynamics® AX 2012 R2

# Upgrade best practices

White Paper

This document describes recommended practices to make upgrade as fast and predictable as possible.

November 2012

www.microsoft.com/dynamics/ax/

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

# Table of Contents

# Overview

This paper describes the additional practices that Microsoft recommends that you follow to speed up and add predictability to an upgrade from Microsoft Dynamics® AX 4 or Microsoft Dynamics AX 2009 to Microsoft Dynamics AX 2012. In addition, the paper describes the practices that Microsoft recommends to speed up and add predictability to an upgrade from Microsoft Dynamics AX 2012 or Microsoft Dynamics AX 2012 Feature Pack to Microsoft Dynamics AX 2012 R2.

## Terminology

| Term | Definition |
|---|---|
| Source system | The Microsoft Dynamics AX 4 SP2 database or Microsoft Dynamics AX 2009 SP1 database |
| Target system | The Microsoft Dynamics AX 2012 database |
| "Source-to-Target" upgrade model | Parallel execution of data copy, data transformations, and sync-on-the-fly to the target system |
| Preprocessing framework | A new upgrade approach to allow data upgrade on a live source system within the staging area (with no impact on the live source system) and to manage running tasks |
| Delta preprocessing | A series of upgrade scripts run on new or updated records after an initial preprocessing run has been executed. |
| Single-user mode | No active user transactions (downtime) |
| Post-processing | Post-synchronization upgrade scripts |
| In-place upgrade | A new upgrade approach that enables upgrades on existing source Microsoft Dynamics AX 2012 RTM systems |
| Partitioning | The use of a logical barrier that isolates its business data from other partitions |

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

# Analyzing customizations

Before you start your data upgrade, you must analyze all customizations and create plans for your code upgrade. All customizations should be evaluated to see whether they are still necessary in Microsoft Dynamics AX 2012, or whether a complete refactoring is required. Work with your value-added reseller (VAR) and independent software vendors (ISVs) to ensure that they have the necessary upgrade scripts in place for your upgrade.

As part of your planning, write the data upgrade scripts that are required to support your customizations. For more information, see Microsoft Dynamics AX 2012 White Paper: How to Write Data Upgrade Scripts for Microsoft Dynamics AX 2012.

Examples of the types of upgrade scripts required include:

- Readiness checks – It is very important to consider data readiness checks for all custom tables that interact with core Microsoft Dynamics AX tables. Any type of ledger account or dimension, address information, or inventory dimension information should always have a readiness script written to check for the existence of the related data.
- Preprocessing and delta scripts – These scripts are required to start to prepare the application data for upgrade by creating shadow tables to map any new fields, and to assign key relations to

the new ledger, inventory, and address data that is required for custom fields and tables. These scripts also should be used to facilitate a change for most of the relationships between custom tables and core Microsoft Dynamics AX tables, so that these relationships use RecIDs instead of the typical string ID value.

- Single-user steps and all target-side operations – The code upgrade should be fully completed by the time single-user processing and target-side processing starts. The only exception would be a test upgrade that is run for the purpose of migrating data for developer testing, but that can only occur after all data upgrade scripts are written. The data upgrade scripts on the target side must include all table and field mapping information in order to proceed.

## Purging and archiving data with the Intelligent Data Management Framework

Prior to upgrade, we recommend that you consider purging or archiving data from the production Microsoft Dynamics AX database. The Intelligent Data Management Framework for Microsoft Dynamics AX (IDMF) is a tool that you can download from Microsoft Dynamics InformationSource. For more information, see the InformationSource services page.

## Analyzing space requirements for databases

Analyzing and adjusting the space available for your databases can significantly improve performance during data upgrade. If you do not increase the size of the databases, database resizing occurs during the upgrade and significantly slows down all operations.

- Increase the size of the source database and transaction logs by at least 35 percent. We recommend that you investigate the actual growth of the database on a test run to more precisely determine how much additional space to allocate.
- Determine the appropriate size for the Microsoft Dynamics AX 2012 database. We recommend that you increase the size by at least 30 percent. We recommend that you closely monitor the actual growth of the database on a test run to more precisely determine how much additional space to allocate.
- Increase and monitor the size of the TempDB database during a test run to determine sizing for the upgrade. A rough estimate for sizing your TempDB database is 20 to 25 percent of the expanded Microsoft Dynamics AX 2012 database. Optimal performance may require splitting the TempDB database into separate files. For more suggestions, see the article Optimizing tempdb Performance.

  **Note:** Microsoft Dynamics AX 2012 does not support Oracle. To upgrade an Oracle database, use the Oracle to SQL migration tool on the source Microsoft Dynamics AX 4 or Microsoft Dynamics AX 2009 Oracle database first, and then upgrade the SQL database to Microsoft Dynamics AX 2012.

## Creating project plans for testing

We recommend that you list each step of the overall upgrade process in a project plan that details each task, the expected duration, and the actual duration. This project plan should include tasks that are run on both the source system and the target system.

The results of the project plan can be used to demonstrate performance changes between test cycles for each source system checklist item up to the **Single-user processing** section.

After test runs, ensure that you validate:

- Key amounts, such as master record balances.
- All types of data.

For every test run, document all the findings identified in each phase. Incorporate your findings into the overall project plan, and address them within the database or application prior to running the next test cycle. Restarting a new testing cycle without addressing all the errors may invalidate the results of the new test cycle and lead to delays.

In your overall upgrade project plan, before you run the final upgrade on the live system, we recommend that you require two complete successful test runs. Performing two successful runs can prevent last-minute problems. In order to achieve two successful test runs, you will probably have to execute multiple partial test runs. The preliminary test runs are used to find errors, understand how to optimize performance, and allow you to create the smallest possible downtime window for the upgrade.

The following table is a very basic example of a project plan. Most project plans are more complex.

Note that the steps for the target system before **Step 12 Start the data upgrade** can be performed ahead of time to minimize the downtime window.

| Step | Data upgrade step | Expected duration | Actual duration |
|---|---|---|---|
| | **Source system** | | |
| 1 | Set Application Object Server (AOS) flags | | |
| 2 | Set to single-user mode – Run in parallel with target system activities | | |
| | **Target system** | | |
| 3 | Set **max degree of parallelism** | | |
| 4 | Load the license | | |
| 5 | Set customer feedback | | |
| 6 | Connect to the source database | | |
| 7 | Presynchronize | | |
| 8 | Create tables | | |
| 9 | Generate table mappings | | |
| 10 | Generate upgrade task prioritization | | |
| 11 | Set batch threads to twice the number of cores. For example, with 8 cores, set the number of batch threads to 16. | | |
| 12 | Start the data upgrade | | |
| 13 | Finalize the Enterprise Portal for Microsoft Dynamics AX upgrade | | |
| 14 | Specify the Role Center website | | |
| 15 | Assign a primary address to parties | | |
| 16 | Upgrade Application Integration Framework (AIF) code | | |
| 17 | Upgrade additional features | | |

| Step | Data upgrade step | Expected duration | Actual duration |
|---|---|---|---|
| 18 | Restart AOS | | |
| 19 | Set **max degree of parallelism** to **1** | | |
| 20 | Assign user security roles | | |
| 21 | Ensure that batch jobs are running | | |
| 22 | Validate data post-upgrade | | |
| 23 | 32-bit clients go live | | |
| 24 | Test connectivity and clients | | |

### Integrating the upgrade plan with user acceptance testing and end-user training

Include user acceptance testing on a test copy of the upgraded dataset in the time allotted for end-user training. The user acceptance testing should be outlined in a clear, well-defined project plan.

### Creating a detailed project plan to test activities performed during the downtime window

We recommend that you create a more granular project plan to outline each individual step required for the downtime window (the single-user processing steps on the source database and each step on the target database). Use the more granular plan for the final upgrade to help ensure that there are no surprises, and that everything works as expected.

It is very important that your project plan for the downtime window be as thorough as possible. It should include testing of all daily activities, all period-end activities, and all critical integration points with other applications or business processes.

Create a smaller version of the user acceptance testing plan that can be run as part of the downtime window testing to validate that the system is ready to go live.

The downtime testing for the full test runs should include at least the following steps:

- Run single-user processing on the source database.
- Launch the Data upgrade cockpit functionality on the target.
- On the target, perform all steps that follow Data upgrade cockpit on the Data upgrade checklist.
- Set up security – Import or populate user and role associations from a previous test database to finalize security setup. Roles and privileges should be created ahead of time as part of the code upgrade.
- Validate that key amounts are accurate, and that all types of data have been upgraded.

## Recommended upgrade process

The key to an optimal data upgrade experience is to plan well in advance, run multiple test cycles that build on lessons learned, and then plan the live data upgrade in complete detail, including time for unexpected last-minute issues.

- Resolve all production readiness errors that arise from readiness checks. We strongly recommend that all readiness errors (including warnings and advisory errors) be resolved on the production system.

- Write and execute readiness scripts, where appropriate, for modified and custom tables.
- Define a robust strategy for creating backups or database snapshots before each step of the upgrade process.

  Backups and snapshots can improve your efficiency if you run multiple test passes during the testing phases. We strongly recommend that you use them during the steps performed during the live upgrade downtime window, in case of unexpected errors (out of disk space, network failures, and so on).
- Use all available performance tools to help debug performance slowdowns (for example, Performance Monitor, Microsoft® SQL Server® dynamic management views (DMVs), and Performance Analyzer for Microsoft Dynamics). Evaluate performance during all phases that rely on batch processing, and then revisit the evaluation with each test run after appropriate adjustments are made.
- Prepare to actively monitor the system for performance bottlenecks during all stages that rely on batch processing. During all test runs, use SQL Server DMVs or Performance Analyzer for Microsoft Dynamics to identify areas where index tuning or code changes are needed. In some cases, an index created on the fly within the test environment can provide immediate benefit to a long-running process.
- Use the preprocessing upgrade state transfer tool. For more information, see Using the preprocessing upgrade state transfer tool. More information about this tool is provided later in the document.

## Preparing the source database

This section lists the steps to prepare the source environment database.

- Review and use best practices for installing and configuring SQL Server for use with Microsoft Dynamics AX:
  - Configure SQL Server and storage settings
  - Top Tips for Maximizing the Performance & Scalability of Dynamics AX 2009 systems on SQL Server 2008
  - Minor Changes in Database Configuration Checklist for Dynamics AX
- Consider implementing trace flag 4199. This flag enables a number of SQL Server performance enhancements that are applicable to Microsoft Dynamics AX. These enhancements should improve the performance of the upgrade readiness and live preprocessing steps of the upgrade. For details about trace flag 4199, review the following KB article: http://support.microsoft.com/kb/974006

  Note that trace flag 4199 is available in the following SQL Server builds:
  - SQL Server 2005 Service Pack 3 Cumulative Update 6
  - SQL Server 2008 Cumulative Update 7
  - SQL Server 2008 Service Pack 1 Cumulative Update 7
  - SQL Server 2008 R2
- Consider increasing the frequency of transaction log backups. Also consider increasing the size of the transaction log to support the additional log space required for live preprocessing and live delta processing.

- Before the **Delta and Single User upgrade** steps, increase the **max degree of parallelism** setting for the server. As a general rule, increase the value to the number of processor cores per processor. For example, on a four-processor server with a total of 24 cores, set **max degree of parallelism** to **6**, as follows.

  ```
  sp_configure 'max degree of parallelism',6
  reconfigure
  ```

  After the **Delta and Single User upgrade** steps, return **max degree of parallelism** to its recommended setting of **1**.

  ```
  sp_configure 'max degree of parallelism',1
  reconfigure
  ```

  **Note: Max degree of parallelism** is a dynamic setting that does not require a restart of the SQL Server instance to take effect.

- During all test runs, use the SQL Server DMVs or Performance Analyzer for Microsoft Dynamics to identify areas where index tuning or code changes are needed. Use the information from those tools to watch for long-running queries, missing indexes, or other index recommendations. Many times, performance can be improved by making changes to the index structure, which only needs to remain modified during the upgrade processing, and can be removed or reset to its original state after the processing is completed.

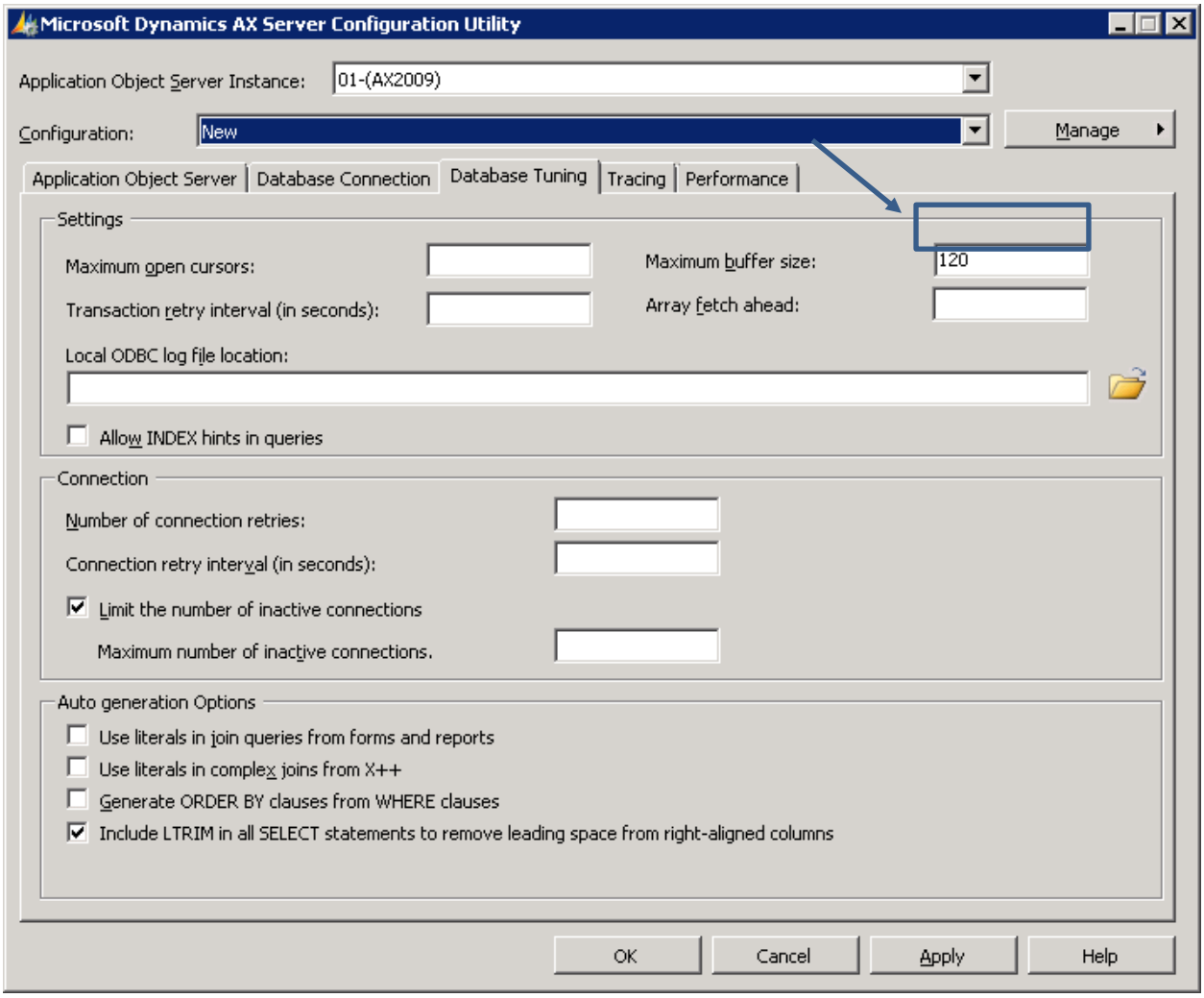
- Creating database snapshots prior to making configuration changes to your database, and prior to launching each phase of the upgrade, will greatly improve recovery time in the event of an error that requires rolling back. For more information, see Typical Uses of Database Snapshots. During testing and the single-user downtime windows, creating and reverting to a database snapshot can be much faster than restoring a backup. Note that we recommend that you continue to perform backups.

# Preparing the source AOS instance

**Note:** We strongly recommend that a dedicated AOS instance be employed for upgrade processing. This not only helps isolate upgrade processing from normal business processing, but it also affords greater flexibility for configuring settings specific to upgrade scripts.

The following recommendations are based on the assumption that they are applied on a dedicated upgrade AOS instance:

- Increase the **Maximum buffer size** value to **120**. This allows Microsoft Dynamics AX to retrieve a much larger set of data with every round trip to the database. Changing this setting enhances the performance of a typical upgrade script, because most upgrade scripts include **while select** statements that run through entire tables.




  This setting particularly benefits the row-by-row operations during the upgrade readiness check and live preprocessing.

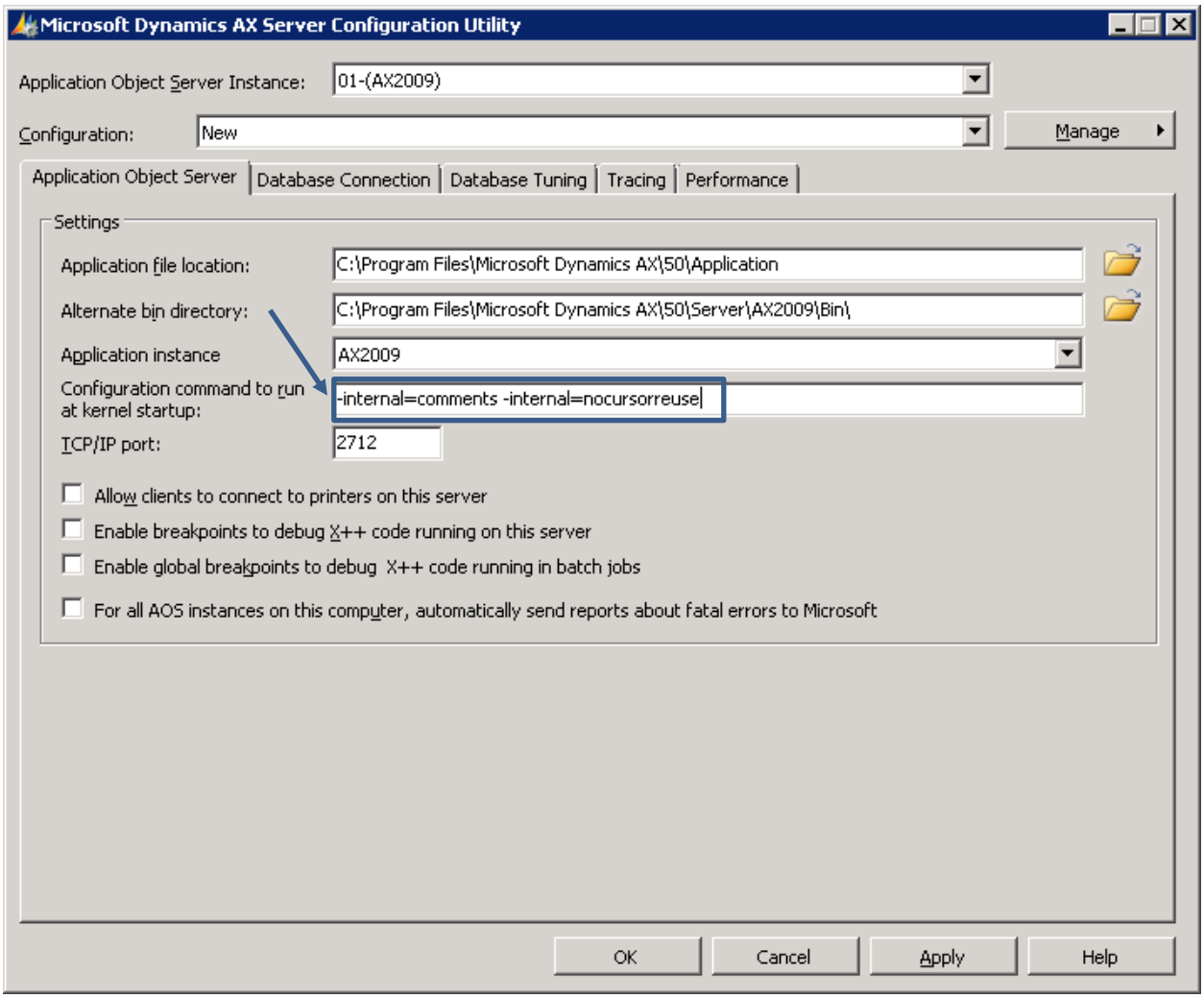
  **Note:** If you change the **Maximum buffer size** setting, you must restart the AOS instance before the change takes effect.

- Batch processing changed between Microsoft Dynamics AX 4.0 and Microsoft Dynamics AX 2009. In Microsoft Dynamics AX 4.0, concurrent batch processing is enabled by launching multiple batch clients to process the DataUpgrade batch group. The number of batch clients that can be used varies, depending on the number of cores and processors on the server, and the availability of client resources to run the batch group.

In Microsoft Dynamics AX 2009, the Batch Framework changed to allow AOS to process batch jobs rather than relying on individual batch clients. The number of threads allocated to batch processing can be changed in the Server configuration window found under **Administration > Setup** inside the Microsoft Dynamics AX client. The number of threads can generally be set to twice the number of cores. For example, with 8 cores, set number of batch threads to 16. However, the performance of AOS and SQL Server should be monitored in order to watch for CPU or I/O bottlenecks, and the number of batch threads should be adjusted accordingly.

- Before the **Delta and Single User upgrade** steps, in the Server Configuration utility, in the **Configuration command to run at kernel startup** field, add the following command:

```
-internal=comments -internal=nocursorreuse
```




**Notes:**

- A change to the AOS kernel startup command requires a restart of the AOS instance to take effect.
- The original purpose of the **–internal=comments** command was to aid debugging by embedding parameter replacement values in the SQL statement string as comments. A side effect of the comments in the SQL text is that they induce a recompile when each SQL

statement is executed. Using this command thereby prevents performance issues caused by parameter sniffing.

- Clear the **Keep Update Objects** configuration key (**Administration** > **Setup** > **System** > **Configuration**). Turning off this key removes all **DEL_** fields tied to prior versions of Microsoft Dynamics AX, and therefore removes several **BLOB** fields (**ntext**) that can cause severe slowdowns during the bulk copy process into the Microsoft Dynamics AX 2012 database.

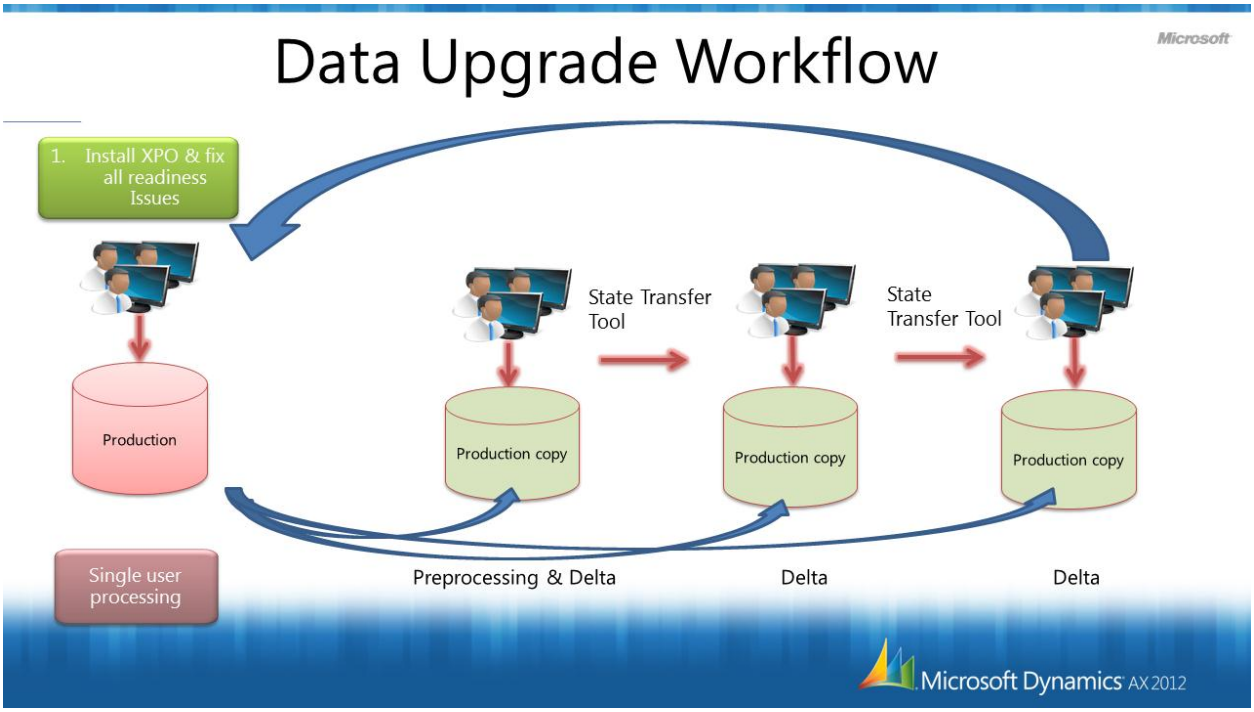
## Using the state transfer tool

The preprocessing upgrade state transfer tool is an optional tool that can be used to transfer the state of the upgrade checklist to different computers. It ships in the source XPO as a private project.

The state transfer tool is intended to help minimize the amount of upgrade preprocessing that must be performed on a production system. Use the state transfer tool if you want to offload the live upgrade preprocessing task to a test server. However, note that use of this tool requires careful consideration and planning. You must plan to use the tool before you reach the tasks in the **Prepare application data for preprocessing** section of the Preprocessing upgrade checklist.

For more information, see Using the preprocessing upgrade state transfer tool.

The following illustration shows the workflow for using the state transfer tool.



## Add temporary indexes on source Microsoft Dynamics AX tables for delete_from statements

Most of the delta scripts run by using a **delete_from** statement with a **notexists** clause to remove rows from the shadow tables when the original record in the Microsoft Dynamics AX table has been deleted or modified. These **delete_from** statements will almost always include a where condition similar to this:

```
custTrans.RecID == Shadow_CustTrans.RefRecID && custTrans.RecVersion ==
Shadow_CustTrans.RecVersionID
```

For some of the larger Microsoft Dynamics AX tables, the delta script may appear to be "stalled" – that is, the iteration value does not change to indicate that the script is processing rows. This can appear even if no records have been altered or deleted in the table being processed. For these "stalled" jobs, check the **Processes** list in the SQL Activity Monitor inside SQL Server Management Studio to see whether a DELETE query is running that is similar to this:

```
declare @P1 nvarchar(5),@P2 nvarchar(5);
set  @P1 = 'dmo';
set  @P2 = 'dmo';
DELETE FROM shadow_inventtransposting
WHERE  (dataareaid = @P1)
AND NOT EXISTS
(SELECT 'x' FROM inventtransposting b
   WHERE  ((b.dataareaid = @P2)
   AND ((b.recid = shadow_inventtransposting.refrecid)
   AND (b.recversion = shadow_inventtransposting.recversionid))))
```

If this type of DELETE query is found, we recommend that you complete the following procedure:

1. Kill the process. Right-click the line in the **Processes** list, and then click **Kill Process**.
2. Add an index to the original source table (inventTransPosting in the preceding example). How you perform this step depends on whether you are running in a test environment with no other users on the system or in a live production environment with other users.
   - If you are running in a test environment with no other users, in the Application Object Tree (AOT), add an index to the source table (InventTransPosting in the preceding example) on RecID, followed by RecVersion. Note that the order is important. Save and synchronize the table to create the new index.
   - If you are running in a live production environment with other users, in SQL Server Management Studio, create a new index on the original source table, on the columns DataAreaID, RecID, and RecVersion (the order is important).
3. Update the upgrade cockpit. Your batch job should have ended with an error – this may take some time if a rollback is needed.
4. Rerun the job that previously stalled. The script should progress fairly quickly past the **delete_from** statement.

After the delta processing is done, remove the indexes from the system, so that they don't add more time to inserts and updates done through normal business processing.

Add the indexes again during the Single-User phase of the upgrade to guarantee quick performance during that critical downtime window.

# Preparing the target database

This section describes the steps to prepare the target database.

On the target database, follow the guidelines listed for the source database. For details, see Preparing the source database.

- Follow best practices for configuring SQL Server.
- Implement trace flag 4199.
- Before target upgrade processing, increase the **max degree of parallelism** setting.

  After target upgrade processing, return **max degree of parallelism** to the default setting.

Perform the following additional steps:

- Set the Microsoft Dynamics AX 2012 database recovery mode to **simple**.

  ```
  ALTER DATABASE AX2012  SET RECOVERY SIMPLE
  ```

  At the conclusion of all upgrade activities, set the Microsoft Dynamics AX 2012 database recovery mode to **full**.

  ```
  ALTER DATABASE AX2012  SET RECOVERY FULL
  ```

- The bulk copy process used to move data from the source database to the target Microsoft Dynamics AX 2012 database requires careful management of I/O activities related to SQL Server.
- Proper database sizing for the target database and the TempDB database is also extremely important, because it is important to avoid any possibility of database resizing during the bulk copy phase.
  - Determine correct sizing for the Microsoft Dynamics AX 2012 database, and set it before you begin to upgrade. A rough estimate to use as a starting point is 30 percent larger than the expanded size of the source database.
  - Increase the size of the TempDB database before you begin to upgrade. A rough estimate for sizing your TempDB database is 20 to 25 percent of the expanded Microsoft Dynamics AX 2012 database. Optimal performance may require splitting the TempDB database into separate files. For more information, see Optimizing tempdb Performance.
- If the source database and target database are on separate servers, network latency and performance are also important factors to monitor. In some tests, multi-server environments with no network latency issues have shown a 30 percent slowdown over a single-server environment. Extreme slowdowns are possible if there are also latency issues to deal with.
- At the conclusion of upgrade activities, the indexes in the Microsoft Dynamics AX 2012 database will be highly fragmented. Before you start normal processing activities, we recommend that you investigate possible fragmentation issues and rebuild indexes where appropriate. For example, you can use the sys.dm_db_index_physical_stats DMV to look for and rebuild indexes. For more information, see example D in sys.dm_db_index_physical_stats (Transact-SQL).

# Preparing the target AOS instance

Before the data upgrade phase, follow these guidelines (which are also required for the source AOS instance). For details, see Preparing the source AOS instance.

- Increase the **Maximum buffer size** value to **120**.
- Set the **Configuration command to run at kernel startup** value to:

  ```
  -internal=comments -internal=nocursorreuse
  ```

  **Note:** In rare cases, the use of the **internal=comments** command in the target Microsoft Dynamics AX environment may cause SQL errors if data contains SQL comment delimiters (**/*** or ***/**).

  If such errors are encountered, an alternative approach to resolving parameter sniffing issues is to enable SQL Server trace flag 4136. For more information, see

http://support.microsoft.com/kb/980653. Trace flag 4136 is only available in SQL Server 2008 R2 Cumulative Update 2, SQL Server 2008 Service Pack 1 Cumulative Update 7, and beyond.

Disable trace flag 4136 after upgrade activities have been completed, unless you have confirmed through testing that it provides performance benefits.

**Important:** After data has been upgraded, remove these settings. If these settings are not removed, performance can be significantly impacted.

Perform the following additional steps:

- During the upgrade phase, monitor which scripts are taking the longest time, and adjust their priority.

  Often, a small subset of the upgrade processing scripts consumes the majority of the data upgrade processing time. During testing, investigate bottlenecks caused by table dependencies to determine whether particular scripts are blocking the remaining scripts from processing. If you find a script that is blocking others, you may be able to adjust the priority for that script in the Generate upgrade task prioritization window. The dependencies for that script should also be evaluated to ensure that they are required.

  In some cases, the built-in company dependency on a company-specific upgrade script should be evaluated to determine whether other companies can continue processing rather than waiting for the long-running company to finish. Keep in mind that removing or changing the company dependency may lead to contention issues if multiple companies are running through the same upgrade script at the same time in different companies. For more information about dependencies, see the Microsoft Dynamics AX 2012 Upgrade Guide.

- Set the bulk copy processing block size to 10,000.

  Within the Launch data upgrade cockpit, use the **Configuration** tab to manage the settings used for bulk copy processing. Generally, we recommend that you start with a block size of 10,000, and then adjust up or down based on performance monitoring on the SQL Server.

# Upgrade considerations for inheritance tables in Microsoft Dynamics AX 2012 R2

In Microsoft Dynamics AX 2012, the table inheritance feature was introduced. A key property of tables in an inheritance relationship is the **SupportInheritance** property. Only tables that have **SupportInheritance**=**Yes** can participate in inheritance relationships.

# Physical vs. logical

In Microsoft Dynamics AX 2012, the base and derived tables were physically separate tables in the underlying SQL Server database. Each table could have business fields that you add. However, to the X++ programmer who issued SQL **Select** statements against the derived table, it appeared as if the fields on the base table were on the derived table. Application Object Server (AOS) provided this appearance. This appearance made programming easier.

# Physical change in Microsoft AX 2012 R2

Starting in Microsoft Dynamics AX 2012 R2, for performance reasons, the derived tables are removed from SQL Server. All fields that were on the derived tables are now physically stored on the base table. In Microsoft Dynamics AX 2012 R2, when you use the AOT to add a business field to the derived table, the field is physically created on the base table.

This physical change affects the correct way to perform data upgrade of customizations. This only affects direct SQL scripts.

AOS still makes it appear that there are two tables, and no functionality has changed from the perspective of the X++ programmer. There is no change to how the two logical tables are displayed in the AOT.

## Comparison examples

The following example shows a base and derived table in the AOT. TableDog extends TableAnimal.

In the AOT on each system, you add a custom field to each of the two tables:

- **NickName** – Added to the base table TableAnimal.
- **ColorOfFur** – Added to the derived table TableDog.

The following table has a row for both versions of Microsoft Dynamics AX. In the later R2 version, both custom fields are on the same table, the base table TableAnimal, and the derived table is gone from SQL Server. The Comments column highlights the changes.

| Version of Microsoft Dynamics AX | Table/Column lists | Comments |
|---|---|---|
| Microsoft Dynamics AX 2012 | Table-Name Column-Name<br>---------- ----------<br>TABLEANIMAL DATAAREAID<br>TABLEANIMAL INSTANCERELATIONTYPE<br>TABLEANIMAL **NICKNAME**<br>TABLEANIMAL RECID<br>TABLEANIMAL RECVERSION<br>TABLEANIMAL RELATIONTYPE<br><br>TABLEDOG **COLOROFFUR**<br>TABLEDOG DATAAREAID<br>TABLEDOG RECID<br>TABLEDOG RECVERSION<br>TABLEDOG RELATIONTYPE | Notice that the two custom fields, **NickName** and **ColorOfFur**, are on different physical tables in SQL Server.<br>This matches the appearance in the AOT. |
| Microsoft Dynamics AX 2012 R2 | Table-Name Column-Name<br>---------- ----------<br>TABLEANIMAL **COLOROFFUR**<br>TABLEANIMAL DATAAREAID<br>TABLEANIMAL INSTANCERELATIONTYPE<br>TABLEANIMAL **NICKNAME**<br>TABLEANIMAL PARTITION<br>TABLEANIMAL RECID<br>TABLEANIMAL RECVERSION<br>TABLEANIMAL RELATIONTYPE | Notice that in Microsoft Dynamics AX 2012 R2, the two custom fields are both on the base table.<br>The derived table no longer exists in SQL Server. However, the derived table is still displayed in the AOT. |

## Upgrade suggestions for inheritance tables in Microsoft Dynamics AX 2012 R2

When running an upgrade, you must run it in physically separate tables (that is, in Table per Type mode), so that direct SQL scripts run correctly. When running Microsoft Dynamics AX R2 in

production, you must run it in Table per Hierarchy mode (that is, in physically combined tables). Specifically:

- Always click **Register for Upgrade** for both code and data upgrade machine installations. This installs a stored procedure to run Microsoft Dynamics AX in Table per Type mode. Always be in Table per Type database mode for a code and data upgrade.
- Always be in Table per Hierarchy database mode in production. It is not supported to run Microsoft Dynamics AX in Table per Type mode in production. A checklist item for the major version upgrade and in-place upgrade switches Microsoft Dynamics AX into Table per Hierarchy mode. This is a one-way switch; you cannot go back to Table per Type mode.
- The upgrade task to convert Microsoft Dynamics AX from Table per Type mode to Table per Hierarchy mode is in the checklist. An Infolog message is displayed. You **must** complete the steps in the Infolog message (even though the message says that no action is required).

## Partitioning

If you plan to upgrade into partitions, do not set up partitions (via the Initialization Checklist) on the target environment. The target partitions will be created automatically by the "Connect to Source" task during the target upgrade step.

For more information about partitioning, see http://msdn.microsoft.com/en-us/library/1fdc4428-e235-4cd6-ae2f-f8f58f806db8.aspx.

## Code upgrade best practices

- Investigate your features to see which ones you have to carry forward. Microsoft Dynamics AX 2012 has added significant features, and the base product may have features that replace your existing features, so that you do not need to carry these features forward. It is recommended that you import all elements (via AOD) and, if necessary, delete them in the Microsoft Dynamics AX 2012 environment.
- Never delete Data Dictionary Elements. It is recommended that you import all Data Dictionary Elements and use SYSDeletedObjects. Deleting Data Dictonary Elements will cause data to be lost forever. If you are confident that you do not require the data, you can use IDMF to archive data.
- Always use AOD import (never use XPO export/import). If you absolutely have to use XPO import, make sure that you keep IDs by using the **Export with ID** option when you export XPOs on the source system.
- Understand the Microsoft changes that impact your upgrade. When you move customizations, it is important to have a detailed understanding of the new Microsoft Dynamics AX 2012 changes. See the code upgrade white papers: http://www.microsoft.com/en-us/download/details.aspx?id=20864.
- Investigate what features are not used often (UI Elements). Some metadata (forms and SSRS reports) might be infrequently used, and therefore they do not have to be upgraded.
- Always keep a copy of your source AOD files prior to importing the upgrade XPO files. Make sure to import the AODs for your code upgrade from this backup copy, which does not contain any upgrade objects. If the AOD that you import for your code upgrade contain the source upgrade objects, you will receive errors when you import these AODs.
- Always follow the code upgrade checklists.
- Before you start your code upgrade, make sure that you have the necessary ISV and VAR model files, if there are any.

- When a code upgrade requires Microsoft Dynamics AX 2012 R2 ISV models, make sure that you ask the ISV to provide the ID mapping files **before** you import the ISV models into your code upgrade environment.
- When completing a code upgrade, always complete the lowest working layer upgrade before you upgrade the higher layers.
- When completing a data upgrade, fix your schema first, then fix your code, and then fix your UI (for example, ENUM -> EDT -> Table -> Code -> UI)

# In-place upgrade

This new upgrade approach enables an upgrade from an existing source Microsoft Dynamics AX 2012 RTM or Microsoft Dynamics AX 2012 Feature Pack system.

- In-place upgrade replaces the Microsoft Dynamics AX 2012 RTM and Microsoft Dynamics AX 2012 Feature Pack model files with the Microsoft Dynamics AX 2012 R2 model files. **After setup is completed and the model files are replaced, restart AOS, and run ax32 –startupcmd=kernelcompileall. This is required to ensure consistency between metadata and p-code.**
- No Microsoft Dynamics AX 2012 RTM component (AOS, client, Help Server, and so on) can co-exist with Microsoft Dynamics AX 2012 R2. Therefore, you must upgrade all components because there is no backward compatibility across components.
- The supported upgrade path is from Microsoft Dynamics AX 2012 or Microsoft Dynamics AX 2012 Feature Pack plus cumulative update 3 (CU3) before you move to Microsoft Dynamics AX 2012 R2. We recommend that the production environment be running CU3 before you upgrade to Microsoft Dynamics AX 2012 R2.
- You must keep IDs the same between Microsoft Dynamics AX 2012 RTM and Microsoft Dynamics AX 2012 R2.
- Use SYSDeletedObjects62 only for an in-place upgrade.
- Existing licenses for Microsoft Dynamics AX 2012 RTM or Microsoft Dynamics AX 2012 Feature Pack do not work with Microsoft Dynamics AX 2012 R2. You will need the Microsoft Dynamics AX 2012 R2 license when upgrading.
- Business Intelligence (SSRS and cubes) and Help Server components do not support in-place upgrades. You will need to manually uninstall and then reinstall these components.
- Before starting your data upgrade, increase the AOS **Maximum buffer size** value to **30**. Changing this setting enhances the performance of a typical upgrade script, because most upgrade scripts include **while select** statements that run through entire tables.
- The Microsoft installer will automatically update components to expedite the in-place upgrade and maintain the existing configuration settings. The supported approach is to upgrade these components in this order:

  1. In-place upgrade of the database
  2. In-place Upgrade of AOS
  3. In-place upgrade of the clients
  4. Manual uninstall/install of the other components

- We support an in-place upgrade silent install of the clients. For example, you are upgrading a client on machine A after the database and AOS have been upgraded on machine B. You can use these parameters to run Setup in silent install mode to upgrade the client in machine A:
  - RunMode=Custom

- HideUI=1
- OptInCEIP=0
- AcceptLicenseTerms=1
- InstallClientUI=1

## Updates since initial publication

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

| Date | Change |
|---|---|
| November 2012 | Added sections about partitioning, in-place upgrade, code and data best practices, and temporary indexes on source Microsoft Dynamics AX tables for **delete_from** statements. |
| April 2012 | Updated the illustration on page 11 to read Preprocessing & Delta under the first production copy. |
| January 2012 | Initial publication |

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



Microsoft