Migrating Objects from a Model Repository to a PowerCenter Repository
Abstract
Informatica supports the migration of mappings, mapplets, and logical data object models created in Informatica Developer to PowerCenter. This article explains how you can migrate objects from a Model repository to a PowerCenter repository. The article also outlines guidelines and restrictions to consider, and notes changes to objects that might occur during migration.

Supported Versions
- Informatica Big Data Management 9.6.1, 10.0, 10.1
- Informatica Data Quality 9.6.1, 10.0, 10.1
- Informatica Data Services 9.6.1, 10.0, 10.1
- Informatica PowerCenter 9.5.1, 9.6.1, 10.1

Table of Contents
Overview ............................................................... 2
What to Know Before You Migrate .................................. 3
  Guidelines ................................................................. 3
  Restrictions ............................................................ 3
  Changes to Imported Objects ....................................... 5
  Setting the PowerCenter Compatibility Level .................... 7
Exporting Objects Directly to a PowerCenter Repository ............ 7
Exporting Objects to an XML File .................................... 8
Importing Objects from an XML File ................................ 8
Migration Results ................................................... 12
  Updating Imported Objects .......................................... 12

Overview
Certain use cases might require you to migrate mappings, mapplets and logical data objects from a Model repository to a PowerCenter repository.

For example, you can use Developer tool to generate a mapping from an SQL query. PowerCenter users who want to use generated mappings must export the mappings from a Model repository to PowerCenter.

As another example, you can use the Developer tool to create logical data object models as an SQL data service to prototype a data mart. When the prototype is complete, you can then export the logical data models to PowerCenter to use in mappings.

You can migrate content from a Model repository to a PowerCenter repository using the following methods:
Export objects directly to the PowerCenter repository.
  You can use the Developer tool to export objects from a Model repository directly into a PowerCenter repository. You must be able to connect to the PowerCenter repository with the Developer tool to use this functionality.
Export objects to an XML file.

You can use the Developer tool to export objects from a Model repository to an XML file. You can then use the PowerCenter Repository Manager to import the objects into a PowerCenter repository.

Due to differences between the Model repository PowerCenter, you might not be able to migrate all dependent objects within a mapping or mapplet. The migration process might also make changes to some objects.

What to Know Before You Migrate

Because the Model repository and the PowerCenter repository support different data structures, certain conditions and restrictions apply to object migration.

Review the guidelines to avoid potential issues during migration. Review the restrictions to understand what mappings, mapplets and dependent objects cannot be exported.

You can set the PowerCenter release compatibility level in the Developer tool to verify that objects are compatible with a certain PowerCenter release. The compatibility level you set applies to all mappings, mapplets, and logical data object models you can view in Developer tool.

The Developer tool performs two validation checks when you validate a mapping, mapplet, or logical data object model. The Developer tool first verifies that the object is valid in Developer tool. If the object is valid, the Developer tool then verifies that the object is valid for export to the selected release of PowerCenter. You can view compatibility errors in the Validation Log view.

Guidelines

You cannot export certain objects from a Model repository to a PowerCenter repository.

Use the following guidelines when you export objects to avoid potential issues:

Verify the PowerCenter version.

Verify that the objects you want to export from the Model repository are compatible in the target PowerCenter version.

Verify that object names are unique.

If you export an object to a PowerCenter repository, the export process replaces the PowerCenter object if it has the same name as an exported object.

Verify that the code pages are compatible.

The export process fails if the Developer tool and PowerCenter use code pages that are not compatible.

Verify precision mode.

By default, the Developer tool runs mappings and mapplets with high precision enabled, while PowerCenter runs sessions with high precision disabled. If you run Developer tool mappings and PowerCenter sessions in different precision modes, they can produce different results. To avoid differences in results, run the objects in the same precision mode.

Restrictions

You cannot export a mapping, mapplet, or logical data object model that contains any object that is not valid in PowerCenter.

You cannot export a mapping, mapplet, or logical data object model when it contains the following objects:

A long name

You cannot import an object if its name exceeds 80 characters.
A dynamic port
You cannot export a mapping or mapplet that contains dynamic ports.

A Data Processor transformation
You cannot export a mapping or mapplet that contains a Data Processor transformation.

A Decision transformation that uses a system parameter
You cannot export a mapping or mapplet that contains a Decision transformation if the transformation script includes a system parameter. The export operation cannot convert the system parameter to a value that PowerCenter can use. Before you export an object with a Decision transformation that uses a system parameter, replace the parameter with an appropriate value.

Mapping output
You cannot export a mapping or mapplet that returns mapping output.

A Joiner transformation with certain join conditions
You cannot export mappings and mapplets that contain a Joiner transformation with a join condition that is not valid in PowerCenter. In PowerCenter, a user defines join conditions based on equality between the master and the detail sources.

In the Developer tool, you can define other types of join conditions. For example, you can define a join condition based on the equality or the inequality between the master and the detail sources. You can define a join condition that contains transformation expressions. You can also define a join condition, such as 1 = 1, that causes the Joiner transformation to perform a cross-join.

The types of join conditions not above are not valid in PowerCenter.

A Lookup transformation with renamed ports
The PowerCenter Integration Service queries the lookup source based on the lookup ports in the transformation and a lookup condition. Therefore, the port names in the Lookup transformation must match the column names in the lookup source.

A Lookup transformation with certain custom SQL queries
The Developer tool uses different rules than PowerCenter to validate SQL query syntax in a Lookup transformation. A custom SQL query written in the Developer tool that uses the AS keyword or calculated fields is not valid in PowerCenter. You cannot export mappings or mapplets to PowerCenter that contain a Lookup transformation with an SQL query that uses the AS keyword or calculated fields.

Sources that are not available in PowerCenter
If you export a mapping or mapplet that includes sources that are not available in PowerCenter, the mapping or mapplet fails to export.

You cannot export a mapping or mapplet with the following sources:

- Complex file data object
- ADABAS
- Amazon Redshift
- AzureBlob
- DataSift
- Facebook
- HBase
- IMS
• JD Edwards EnterpriseOne
• LDAP
• LinkedIn
• Microsoft Dynamic CRM
• OData
• Twitter
• Twitter Streaming
• Web Content-Kapow Katalyst

**Concatenated ports**

The export process fails if you export a mapplet that contains a multigroup Input transformation and the ports in different input groups connect to the same downstream transformation.

**Nested mapplets with unconnected Lookup transformations**

The export process fails if you export any type of mapping or mapplet that contains another mapplet with an unconnected Lookup transformation.

**Timestamp with Time Zone or Timestamp with Local Time Zone Data Types**

When you import a mapping that contains data of the Timestamp with Time Zone or Timestamp with Local Time Zone data types from the Developer tool, the PowerCenter Repository Manager fails to convert the mapping.

**Changes to Imported Objects**

The Developer tool updates mappings, mapplets, port names, and data quality transformations in the PowerCenter import XML file to ensure that the objects are valid PowerCenter objects.

**Mappings and Mapplets**

The following changes are made to mappings and mapplets:

**Creates Expression transformations.**

The Developer tool creates an Expression transformation immediately downstream from each Input transformation and immediately upstream from each Output transformation. The Expression transformation contains pass-through ports.

**Assigns default values to Input and Output transformations.**

A Developer tool user can set default values for ports in Input and Output transformations. In PowerCenter, Input and Output transformation ports do not have default values. Therefore, the Developer tool assigns the Input transformation port default values to the Expression transformation immediately downstream from each Input transformation. Similarly, it assigns the Output transformation port default values to the Expression transformation immediately upstream from each Output transformation.

The export process names the Expression transformations as follows:

`Expr_<InputOrOutputTransformationName>`

**Creates multiple Input transformations.**

The Developer tool allows multigroup Input transformations, while PowerCenter does not. Therefore, the Developer tool creates one Input transformation for each input group in a multigroup Input transformation.
Creates Output transformations.

If you export a mapplet and convert targets to Output transformations, the export process creates an Output transformation for each target. The export process names the Output transformations as follows:

\(<\text{MappletInstanceName}>_<\text{TargetName}>\)

Changes to mappings or mapplets that include parameters.

When you export a mapping or mapplet that contains parameters from a Model repository, the parameters resolve to default values when you import the parameters to PowerCenter. The import process can resolve any SQL expression that contains a parameter.

System parameters resolve to the equivalent PowerCenter system parameters. If PowerCenter does not have the equivalent system parameter, the system parameter reference remains in the mapping after you import it to PowerCenter. You need to edit the mapping and change the reference.

Assigns names to Input and Output transformations.

The export process names mapplet Input and Output transformations as follows:

\(<\text{TransformationName}>_<\text{InputOrOutputGroupName}>\)

 Renames mapplet ports.

The export process renames mapplet ports as follows:

\(<\text{PortName}>_<\text{GroupName}>\)

Flattens nested mapplets.

The Developer tool allows nested mapplets, which are mapplets within other mapplets. PowerCenter does not allow nested mapplets. The Developer tool converts nested mapplets to a single mapplet, without nesting.

Exports the mapping without the SAP source.

When you export a mapping with an SAP source, the Developer tool exports the mapping without the SAP source. When you import the mapping into the PowerCenter repository, the PowerCenter Repository Manager imports the mapping without the source. The output window displays a message indicating the mapping is not valid. You must manually create the SAP source in PowerCenter and add it to the mapping.

Port Names

The Developer tool makes the following changes to port names:

Appends group names to port names in multigroup transformations.

In multigroup transformations, the Developer tool appends the group name to the port name. However, in SQL transformations, the Developer tool does not append the group name to the port name.

Appends group names to port names in Input and Output transformations.

The Developer tool appends the Input transformation group name to the port names in Input transformations. It also appends the Output transformation name to the port names in Output transformations. For mappings exported to PowerCenter mapplets, and mapplets with targets converted to Output transformations, the Developer tool appends the target name to the port names in Output transformations.

Data Quality Transformations

The Developer tool converts Address Validation, Consolidation, Key Generator, and Match transformations to mapplets.
Setting the PowerCenter Compatibility Level

Set the compatibility level to validate mappings and mapplets before beginning a migration. If you select none, the Developer tool skips release compatibility validation when you validate an object.

1. Click **Edit > Compatibility Level**.
2. Select the compatibility level.

   The Developer tool places a dot next to the selected compatibility level in the menu. The compatibility level applies to all mappings and mapplets you can view in the Developer tool.

Exporting Objects Directly to a PowerCenter Repository

You can use the Developer tool to export objects directly to a PowerCenter repository. You must be able to connect to the PowerCenter repository with the Developer tool to use this functionality.

Before you export an object, set the compatibility level to the appropriate PowerCenter release. Validate the object to verify that it is compatible with the PowerCenter release.

1. Click **File > Export**.

   The **Export** dialog box appears.

2. Select **Informatica > PowerCenter**.
3. Click **Next**.

   The **Export to PowerCenter** dialog box appears.

4. Select the project in the Model repository from which you want to export objects.
5. Select the PowerCenter release to which you want to export the objects.
6. Select **PowerCenter repository**, and then click **Browse** to specify the connection details for the PowerCenter repository.
7. Select a target folder in the PowerCenter repository. You can also select the `pmrep` control file that defines how to import objects into PowerCenter.
8. Select **Convert exported mappings to PowerCenter mapplets** to convert the Developer tool mappings to mapplets in PowerCenter.
9. Select **Convert Target mapplets** to convert data objects used as targets in a mapplet to Output transformations in the PowerCenter mapplet.
10. Select **Export Reference Data** to export any reference table data used by a transformation in an object you export.
11. If you want to export the reference data, specify the location for the reference table data that the Developer tool exports.
12. Select the code page of the PowerCenter repository.
13. Click **Next**.

   The Developer tool prompts you to select the objects to export.
14. Select the objects to export and click **Finish**.

   The Developer tool exports the objects to the file you specified.
Exporting Objects to an XML File

You can export objects from a Model repository to an XML file. You can then use the PowerCenter Repository Manager to import the object from the XML file into the PowerCenter repository.

Before you export an object, set the compatibility level to the appropriate PowerCenter release. Validate the object to verify that it is compatible with the PowerCenter release.

Warning: Make sure that any metadata you define in the XML file is valid. You must be able to create the object you define in the Designer or Workflow Manager. For example, if you edit the metadata for a mapplet, make sure the source is not a VSAM source. The Designer marks mapplets with VSAM sources as invalid.

1. Click File > Export.
   The Export dialog box appears.
2. Select Informatica > PowerCenter.
3. Click Next.
   The Export to PowerCenter dialog box appears.
4. Select the project in the Model repository from which you want to export objects.
5. Select the PowerCenter release to which you want to export the objects.
6. Click File, then specify the name and location of an XML file.
7. Select Convert exported mappings to PowerCenter mapplets to convert the Developer tool mappings to mapplets in PowerCenter.
8. Select Convert Target mapplets to convert data objects used as targets in a mapplet to Output transformations in the PowerCenter mapplet.
9. Select Export Reference Data to export any reference table data used by a transformation in an object you export.
10. If want to export the reference data, specify the location for the reference table data that the Developer tool exports.
11. Select the code page of the PowerCenter repository.
12. Click Next.
   The Developer tool prompts you to select the objects to export.
13. Select the objects to export and click Finish.
   The Developer tool exports the objects to the location you selected. Use the PowerCenter Repository Manager to import the objects from the XML file.

If you export reference table data, copy the reference data files to the PowerCenter directory structure on the machine that hosts Informatica services. The reference data file locations must correspond to the reference table object locations in the Model repository.

For example, copy the reference data files to the following location:

<PowerCenter installation directory>\services\<Model repository project name>\<Folder name>

Importing Objects from an XML File

You can use the PowerCenter Repository Manager to import objects from an XML file into a PowerCenter repository. You can compare objects when importing objects with the Import Wizard. If conflicts occur, you can use the Conflict Resolution Wizard to resolve specific object conflicts.

1. Launch the PowerCenter Repository Manager and connect to the target PowerCenter repository.
2. Click **Repository > Import Objects.**
   The Import Wizard opens to guide you through the process of importing the objects into the target folder.

3. In the Import Wizard, click **Browse** to locate the XML file. Navigate to the directory where the XML file is located. Select the XML file and click **OK.**

4. Click **Next.**

5. Select the objects to import and click **Add.**

   When you select a node in the Objects in File pane and click Add, the Import Wizard adds all mappings, mapplets, and dependent objects listed under that node. The following image shows the MRS_66 node selected, and all of the mappings and dependent source and target objects within it moved to the Objects to Import pane.

   ![Select objects to import](image)

   To remove an object from the Objects to Import pane, select the object and click **Remove.**

   You can right-click an object and choose **Properties** to view the properties associated with an object.

   You can filter which objects to view in the Objects in File pane. Select a folder or repository in the Folders field.

6. Click **Next.**

   The Match Folders step of the Import Wizard appears when you import objects using the Repository Manager. You can match folders listed in the XML file to folders in the destination repository.

7. Click the **Open** button for a folder listed in the Import Wizard.

   The Folder Selection dialog box appears.

8. Select a folder in the destination repository and click **OK.**

   You must select a different folder for each folder listed in the Import Wizard.
Tip: You can create a new folder in the destination repository by clicking Create Folder. Specify the folder properties in the Create Folder dialog box.

9. Click Next.

The Specify Rules for Conflict Resolutions step of the Import Wizard appears when you import objects using the Repository Manager. You can create rules to resolve general object conflicts. You can apply rules to objects with a certain label, objects listed in an object query, objects of the same type, or all objects.

10. To create a new rule, click New Rule. Choose to which objects to apply the rule and select a resolution.

11. Click Next.

The Import Wizard opens the Conflict Resolution Wizard for objects in one of the folders listed in the XML file. The Conflict Resolution Wizard is similar to the Copy Wizard. Use the Conflict Resolution Wizard to resolve specific object conflicts.

The following image shows a conflict detected with an imported mapping.

12. Click Compare Conflict to compare conflicting objects in the XML file and target repository.

Use the window that opens to uncover the conflict. The following image compares the imported mapping with a mapping in the PowerCenter repository folder, indicating that a mapping with the same file name exists in the folder.
You can save the comparison as a text or HTML file.

13. Resolve object conflicts in the Conflict Resolution Wizard. Click Next to proceed through the Conflict Resolution Wizard.

The following image shows the conflict resolved by renaming the imported mapping.

14. Click Close when you resolve all the conflicts for this folder.
The Import Wizard opens the Conflict Resolution Wizard for objects in any other folder listed in the XML file. When you resolve conflicts for all objects in all folders, the Import Wizard continues with the import process. You can click View/Edit to view or edit the object conflicts for the objects in that folder.

**Note:** If you cancel the Conflict Resolution Wizard for a folder, the Import Wizard displays the status of that folder as unresolved. Click Resolve in the Action column for that folder to open the Conflict Resolution Wizard and resolve the object conflicts.

15. Click **Import** in the Import Wizard to import the objects into the repository. The PowerCenter Repository Manager imports the objects into the destination repository, and displays the progress of the import process.

The Output window displays the results of the import process. Errors and warnings are designated by colored text.

16. Click **Done**.

**Migration Results**

When you import mapping or mapplet dependent objects, the Import Wizard places the objects in the appropriate node in the Navigator.

For example, the Import Wizard places mapping targets in the Targets node. If you import a mapping with an ODBC source, the Import Wizard places the dependent source in the ODBC source node in the Navigator, not in the node associated with the data source name.

You might need to update objects after you import them. You might also notice differences between the Developer tool and PowerCenter objects.

**Updating Imported Objects**

After you import objects from the Developer tool, ensure that PowerCenter sessions run.

Complete the following steps to ensure that PowerCenter sessions that use imported objects run:

1. Re-establish key relationships.

   The Developer tool export process does not retain key relationships between source and target tables if the primary key and foreign key tables are in different mappings. After you import mappings in which the primary key and foreign key tables are in different mappings, you must reestablish the key relationships.

2. If the mapping uses an SQL query override, verify that the columns in the query are connected.

   The Developer tool allows you to run a mapping with an SQL query override if the columns in the query are not connected. In PowerCenter, the columns you use in the query must be connected.

3. Create sessions, workflows, and connection objects.

   The Developer tool does not have sessions or workflows. When you import mappings from the Developer tool, you must create PowerCenter sessions and workflows. You must also specify connection information for sources and targets.

4. Verify that mapping sources use the same connection.

   A Developer tool mapping source can join relational data objects that use different connections. A PowerCenter mapping source can join data from related tables when the tables use the same connection. If a Developer tool mapping source joins relational data objects that use different connections, the exported mapping contains a single Source Qualifier transformation. Verify that the tables joined in the Source Qualifier transformation originate from the same relational database and that they use the same connection. If they do not, replace the Source Qualifier transformation with multiple source qualifiers, and join the sources with a Joiner transformation.
5. **Verify the precision mode.**

   By default, the Developer tool runs mappings with high precision enabled, while PowerCenter runs sessions without high precision. If you want PowerCenter sessions to produce the same results as the corresponding Developer tool mappings, run them in the same precision mode.

6. **Verify that reference tables exist.**

   PowerCenter requires that reference tables exist in the directory defined in the INFA_CONTENT environment variable. If INFA_CONTENT is not set, the reference tables must exist in the following PowerCenter services directory:

   `$INFA_HOME\services\<Developer Tool Project Name>\<Developer Tool Folder Name>`

   The PowerCenter administrator can set the INFA_CONTENT environment variable on the machine that hosts PowerCenter services. If INFA_CONTENT is set, copy the reference tables to the INFA_CONTENT directory. If INFA_CONTENT is not set, copy the reference tables to the PowerCenter services directory.

---

**Author**

Dan Hynes  
Principal Technical Writer