Informatica® PowerCenter
10.5.6

Repository Guide

Informatica PowerCenter Repository Guide
10.5.6
May 2024

NOTICES

This Informatica product (the "Software") includes certain drivers (the "DataDirect Drivers") from DataDirect Technologies, an operating company of Progress Software Corporation ("DataDirect") which are subject to the following terms and conditions:

1. THE DATADIRECT DRIVERS ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT.
2. IN NO EVENT WILL DATADIRECT OR ITS THIRD PARTY SUPPLIERS BE LIABLE TO THE END-USER CUSTOMER FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL OR OTHER DAMAGES ARISING OUT OF THE USE OF THE ODBC DRIVERS, WHETHER OR NOT INFORMED OF THE POSSIBILITIES OF DAMAGES IN ADVANCE. THESE LIMITATIONS APPLY TO ALL CAUSES OF ACTION, INCLUDING, WITHOUT LIMITATION, BREACH OF CONTRACT, BREACH OF WARRANTY, NEGLIGENCE, STRICT LIABILITY, MISREPRESENTATION AND OTHER TORTS.

The information in this documentation is subject to change without notice. If you find any problems in this documentation, report them to us at infa_documentation@informatica.com.

Informatica products are warranted according to the terms and conditions of the agreements under which they are provided. INFORMATICA PROVIDES THE INFORMATION IN THIS DOCUMENT "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING WITHOUT ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND ANY WARRANTY OR CONDITION OF NON-INFRINGEMENT.

Publication Date: 2024-05-29

Table of Contents

Preface

Use the *PowerCenter® Repository Guide* to understand and manage the PowerCenter repository. For additional information on related database connectivity issues not covered by this guide, refer to the documentation accompanying your database products.

Informatica Resources

Informatica provides you with a range of product resources through the Informatica Network and other online portals. Use the resources to get the most from your Informatica products and solutions and to learn from other Informatica users and subject matter experts.

Informatica Network

The Informatica Network is the gateway to many resources, including the Informatica Knowledge Base and Informatica Global Customer Support. To enter the Informatica Network, visit https://network.informatica.com.

As an Informatica Network member, you have the following options:

- Search the Knowledge Base for product resources.
- View product availability information.
- Create and review your support cases.
- Find your local Informatica User Group Network and collaborate with your peers.

Informatica Knowledge Base

Use the Informatica Knowledge Base to find product resources such as how-to articles, best practices, video tutorials, and answers to frequently asked questions.

To search the Knowledge Base, visit https://search.informatica.com. If you have questions, comments, or ideas about the Knowledge Base, contact the Informatica Knowledge Base team at KB_Feedback@informatica.com.

Informatica Documentation

Use the Informatica Documentation Portal to explore an extensive library of documentation for current and recent product releases. To explore the Documentation Portal, visit https://docs.informatica.com.

If you have questions, comments, or ideas about the product documentation, contact the Informatica Documentation team at infa_documentation@informatica.com.

Informatica Product Availability Matrices

Product Availability Matrices (PAMs) indicate the versions of the operating systems, databases, and types of data sources and targets that a product release supports. You can browse the Informatica PAMs at https://network.informatica.com/community/informatica-network/product-availability-matrices.

Informatica Velocity

Informatica Velocity is a collection of tips and best practices developed by Informatica Professional Services and based on real-world experiences from hundreds of data management projects. Informatica Velocity represents the collective knowledge of Informatica consultants who work with organizations around the world to plan, develop, deploy, and maintain successful data management solutions.

You can find Informatica Velocity resources at http://velocity.informatica.com. If you have questions, comments, or ideas about Informatica Velocity, contact Informatica Professional Services at ips@informatica.com.

Informatica Marketplace

The Informatica Marketplace is a forum where you can find solutions that extend and enhance your Informatica implementations. Leverage any of the hundreds of solutions from Informatica developers and partners on the Marketplace to improve your productivity and speed up time to implementation on your projects. You can find the Informatica Marketplace at https://marketplace.informatica.com.

Informatica Global Customer Support

You can contact a Global Support Center by telephone or through the Informatica Network.

To find your local Informatica Global Customer Support telephone number, visit the Informatica website at the following link:
https://www.informatica.com/services-and-training/customer-success-services/contact-us.html.

To find online support resources on the Informatica Network, visit https://network.informatica.com and select the eSupport option.

CHAPTER 1

Understanding the Repository

This chapter includes the following topics:

Understanding the Repository Overview

The PowerCenter repository is a relational database managed by the Repository Service.

The repository consists of database tables that store metadata. Metadata describes different types of objects, such as mappings and transformations, that you can create or modify using the PowerCenter Client tools. The Integration Service uses repository objects to extract, transform, and load data. The repository also stores information such as permissions for users.

All repository clients access the repository database tables through the Repository Service. The Repository Service protects metadata in the repository by managing repository connections and using object-locking to ensure object consistency. The Repository Service also notifies you when another user modifies or deletes repository objects that you are using.

Each Repository Service manages a single repository database. You can configure a Repository Service to run on multiple machines, or nodes, in the domain. Each instance running on a node is called a Repository Service process. This process accesses the database tables and performs most repository-related tasks.

The Repository Service uses native drivers to communicate with the repository database. PowerCenter Client tools and the Integration Service communicate with the Repository Service over TCP/IP. When a repository client connects to the repository, it connects directly to the Repository Service process.

You administer the repository using the Repository Manager client tool, the Informatica Administrator, and the *pmrep* and *infacmd* command line programs.

You can connect to and manage multiple repositories. A *repository domain* is a group of repositories in the PowerCenter Client. Repository domains share metadata through a special type of repository called a global

repository. When you configure shared folders in a repository, you can share the objects in the folder with other repositories in the repository domain. You share objects to reuse metadata.

Note: A repository domain is different from a PowerCenter domain, which is the primary unit of administration for the PowerCenter environment.

If you have the team-based development option, you can enable the repository for version control. You can store multiple versions of objects in a versioned repository. You can also perform change-management tasks such as version comparison, change tracking, labeling, and deployment.

Repository Architecture

The PowerCenter repository resides in a relational database. The repository database tables contain the instructions required to extract, transform, and load data. Repository clients access the repository database tables through the Repository Service. A repository client is any PowerCenter component that connects to the repository.

The Repository Service manages repository metadata transaction requests from repository clients. Each Repository Service manages a single repository. The Repository Service uses object-locking to ensure the consistency of metadata in the repository.

A Repository Service process is a multi-threaded process that fetches, inserts, and updates metadata in the repository database tables. A Repository Service process is an instance of the Repository Service that runs on a particular machine, or node.

The Repository Service accepts client metadata transaction requests from the following PowerCenter components:

- **PowerCenter Client tools.** Use the Designer to create and store mapping metadata in the repository. Use the Workflow Manager to store workflow metadata and connection object information in the repository. Use the Workflow Monitor to retrieve workflow run status information and session logs written by the Integration Service. Use the Repository Manager to organize and secure metadata by creating folders. You can manage the repository from the Administrator tool.
- **pmrep and infacmd.** Use pmrep to perform repository metadata administration tasks, such as listing repository objects. Use infacmd to perform service-related functions, such as creating or removing a Repository Service.
- **Integration Service.** When you start the Integration Service, it connects to the repository to schedule workflows. When you run a workflow, the Integration Service retrieves workflow task and mapping metadata from the repository. During the workflow run, the Integration Service writes workflow status information to the repository.

Repository Connectivity

Repository clients such as the PowerCenter Client, the Integration Service, *pmrep*, and *infacmd* connect to the repository through the Repository Service.

Repository clients communicate with the Repository Service through a specified port over a TCP/IP connection. You configure the TCP/IP port number when you install the Repository Service.

Because PowerCenter services can reside on multiple nodes in the domain, the Repository Service relies on another service called the Service Manager to direct client requests to the appropriate Repository Service process.

The following process describes how a repository client connects to the repository database:

1. The repository client sends a repository connection request to the master gateway node, which is the entry point to the domain.
2. The Service Manager sends back the host name and port number of the node running the Repository Service. If you have the high availability option, you can configure the Repository Service to run on a backup node.
3. The repository client establishes a link with the Repository Service process. This communication occurs over TCP/IP.
4. The Repository Service process communicates with the repository database and performs repository metadata transactions for the client.

Understanding Metadata

The repository stores metadata that describes how to extract, transform, and load source and target data. PowerCenter metadata describes different kinds of repository objects. You use different PowerCenter Client tools to develop each kind of object.

If you enable version control, you can store multiple versions of metadata objects in the repository.

You can also extend the metadata stored in the repository by associating information with repository objects. For example, when someone in your organization creates a source definition, you may want to store the name of that person with the source definition. You associate information with repository metadata using metadata extensions.

Related Topics:

- "Version Control" on page 26
- "Metadata Extensions" on page 163

Objects Created in the Designer

Use the Designer to create and edit the following repository objects:

- **Source definitions.** Detailed descriptions of database objects (tables, views, and synonyms), flat files, XML files, or COBOL files that provide source data. For example, a source definition might be the complete structure of the EMPLOYEES table, including the table name, column names and datatypes, and any constraints applied to these columns, such as NOT NULL or PRIMARY KEY. Use the Source Analyzer tool to import and create source definitions.
- **Target definitions.** Detailed descriptions for database objects, flat files, or XML files to receive transformed data. During a session, the Integration Service writes the transformed data to targets. Use the Target Designer tool to import or create target definitions.
- **Transformations.** A transformation generates, modifies, or passes data through ports that you connect in a mapping or mapplet. When you build a mapping or mapplet, you add transformations and configure them to handle data according to your business purpose.

- **Reusable transformations.** You can design a transformation that you can reuse in multiple mappings or mapplets within a folder, a repository, or a repository domain. Rather than recreate the same transformation each time, you can make the transformation reusable and add instances of the transformation to individual mappings or mapplets. Use the Transformation Developer tool to create reusable transformations.
- **Mappings.** A mapping specifies how to move and transform data from sources to targets. Mappings include source and target definitions and transformations. Transformations describe how the Integration Service transforms data. Mappings can also include shortcuts, reusable transformations, and mapplets. Use the Mapping Designer tool to create mappings.
- **Mapplets.** You can design a mapplet to contain sets of transformation logic to be reused in multiple mappings within a folder, a repository, or a repository domain. Rather than recreate the same set of transformations each time, you can create a mapplet containing the transformations and then add instances of the mapplet to individual mappings. Use the Mapplet Designer tool to create mapplets.
- **User-defined functions.** You can create user-defined functions using the PowerCenter transformation language. Create user-defined functions to reuse expression logic and build complex expressions. User-defined functions are available to other users in a repository.
- **Multi-dimensional metadata.** Multi-dimensional metadata refers to the logical organization of data used for analysis in OLAP applications. Dimensions and cubes are most often used by end users of OLAP applications. Use the Target Designer tool to create dimensions and cubes.

You can also create shortcuts to metadata in shared folders. Use shortcuts to repository objects in shared folders. You can create local shortcuts to shared folders within the same repository and global shortcuts to shared folders in the global repository of the repository domain. Use the Designer to create shortcuts.

Objects Created in the Workflow Manager

Use the Workflow Manager to create and edit the following repository objects:

- **Database connections.** The Integration Service uses database connections to connect to the source and target databases.
- **Sessions.** Sessions are workflow tasks that contain information about how the Integration Service moves data through mappings. You create a session for each mapping you want to run. To run the session, place it in a workflow. Use the Workflow Designer to create sessions.
- **Workflows.** A workflow is a set of instructions, divided into tasks, the Integration Service uses to extract, transform, and load data.
- **Workflow tasks.** Workflow tasks are instructions the Integration Service executes when running a workflow. Workflow tasks perform functions supplementary to extracting, transforming, and loading data. Workflow tasks include commands, decisions, timers, and email notification.
- **Worklets.** Worklets are objects that represent a set of workflow tasks that allow you to reuse a set of workflow logic in several workflows. You can run worklets in workflows and nest worklets in other worklets.

Objects Created in the Repository Manager

Use the Repository Manager to create, edit, and delete folders. Folders organize and store metadata in the repository. You can control access to a folder by configuring folder permissions. You can also configure a folder to share stored metadata with other users.

Global Objects

When you edit a global object, the Repository Service applies the changes at the repository level. You use different PowerCenter Client tools to develop each kind of global object. You can create the following global objects:

- **Labels.** If you have a team-based development option, you can associate labels with any versioned object or group of versioned objects in a repository. Use labels to track versioned objects during development, mark development milestones, improve query results, and organize groups of objects for deployment or import and export. Use the Repository Manager to create and edit labels.
- **Deployment groups.** A deployment group is a set of objects that you copy to a repository. You can create a deployment group that contains references to objects from multiple folders across the repository. You can create a static deployment group that you manually add objects to or create a dynamic deployment group that uses a query to populate the group. Use the Repository Manager to create and edit deployment groups.
- **Object queries.** Use an object query to search for versioned and nonversioned objects in the repository that meet specified conditions. You can save object queries for later use. You can create a private object query, or you can share it with all users in the repository. Use the Designer, Workflow Manager, or Repository Manager to create and run an object query.
- **Connection objects.** You create connection objects in the repository when you define database, FTP, and external loader connections in the Workflow Manager. You can configure and manage permissions within each connection object. Use the Workflow Manager to create and edit connection objects.

Labels, deployment groups, and object queries help you perform version control by grouping versioned objects.

Related Topics:

- "Version Control" on page 26

Dependent Objects

A dependent object is an object used by another object. For example, a source definition referenced by a mapping is a dependent object of that mapping. You can perform the following tasks on dependent objects:

- **Copy.** You can copy dependent objects with the Copy Wizard in the Workflow Manager, Designer, and Repository Manager. When you copy an object, the Copy Wizard also copies all dependent objects.
- **Deploy.** You can add dependent objects to a static deployment group. You use a deployment group to copy objects to another folder or repository.
- **View.** You can view dependent objects before modifying or deleting parent objects in the Repository Manager, Designer, and Workflow Manager.
- **Modify or validate.** When you modify a dependent object, you may cause the parent object to become invalid. For example, if you modify a mapping by updating a port datatype to an incompatible dataype, the session may become invalid.
- **Import or export.** You can choose to import or export a parent object with or without its dependent child objects. You might want to export and import an object without its dependent objects if you change a workflow property, such as a workflow variable, but you did not change any task in the workflow.

Understanding Repository Object Locks

The repository uses locks to prevent users from duplicating or overwriting work.

The Repository Service creates the following types of locks on repository objects when you view, edit, or run them in a workflow:

- In-use lock. Placed on objects you want to view.
- Write-intent lock. Placed on objects you want to modify.
- Execute lock. Locks objects you want to run, such as workflows and sessions.

The Repository Service creates and releases locks on repository objects. The repository allows multiple users to obtain in-use locks on an object. The repository allows one write-intent lock per object. This keeps multiple users from editing the object at one time, thus preventing repository inconsistencies. If you attempt to modify an object that already has a write-intent lock, the repository displays a message box:

```
The [object_type] [object_name] is already locked by [user name].
```

The repository then issues an in-use lock for the object, allowing you to view the object.

The repository allows one execute lock per object. This keeps you from starting a workflow that is already running, which can cause the Integration Service to load duplicate or inaccurate data.

The following table lists each repository lock and the conditions that create it:

Repository Lock	Created When	Maximum per Object
In-use	- Viewing an object in a folder for which you do not have write permission. - Viewing an object that is already write-locked. - Exporting an object.	Unlimited
Write-intent	- Viewing an object in a folder for which you have write permission. - Editing an object in a folder for which you have write permission. - Importing an object.	1
Execute	Starting, aborting, or recovering a workflow.	1

Locking the Same Object

The repository permits multiple in-use locks, one write-intent lock, and one execute lock simultaneously on each repository object. This means that you can edit a session while the Integration Service runs the session and another user views the session.

For example, if you obtain a write-intent lock on a workflow before the Integration Service starts the workflow, the Integration Service runs the version of the workflow existing in the repository when the workflow starts. If you save changes to the repository before the workflow starts, the Integration Service runs the newly-edited workflow. If you save changes after the workflow starts, the Integration Service runs the original workflow and the repository updates the changes after the workflow completes.

When the workflow starts, the Integration Service obtains an execute lock on the workflow and tasks in the workflow. If you try to start the workflow, the repository displays a message stating that the workflow is already running. If you try to edit the workflow or task when another user has a write-intent lock, you receive an in-use lock.

Locking Within Objects

Some repository objects contain other repository objects. For example, workflows contain sessions and tasks, sessions contain mappings, and mappings contain at least one source and target definition.

You obtain an in-use lock on an object when you view it. You can view an object used by another object without affecting the other object. However, if you save changes to an object used by other objects, the repository might mark the other objects invalid. Before using invalidated objects, you must validate them.

For example, you open a mapping used by a session, delete a transformation, and save the changes. When you save the mapping, the repository notes the mapping has been changed, and marks the session and every other session using the mapping invalid. The change might invalidate any workflow containing the session.

Locking with Cubes and Dimensions

Editing or deleting cubes and dimensions can affect many objects in the repository. When you edit a property of a cube or dimension, the Repository Service creates a write-intent lock on all related objects until you save the changes or cancel the edit. Therefore, if an object is a part of a cube or dimension being edited, you might notice the object is locked even when no one is working with it.

For example, if you use the Dimension Editor to change a Level Property field, the Repository Service locks all related dimension tables until you save the changes. Any user who tries to edit a related dimension table receives an in-use lock on the table.

Locking Business Components

To maintain the integrity of the repository data, the Repository Service locks the business component tree while its contents are being edited. This prevents you from copying or editing the business component.

Locking occurs at the root directory of the business component tree. For example, if Finance is the root directory of the tree, with General Ledger and Accounts Receivable as subdirectories, the Repository Service locks the Finance directory while you make changes to the Accounts Receivable or General Ledger subdirectories. The Repository Service releases a lock when you save the repository.

Acquiring Locks During Deployment

When you copy a folder or deployment group to another repository, you must acquire locks on the objects in the target repository. If the object locks are not immediately available, by default the deployment operation waits until either you cancel the deployment or the object locks are acquired.

If you use the pmrep command line program to copy folders or deployment groups, you can specify a timeout for the deployment operation. If pmrep does not acquire object locks in the target repository during the timeout period, the deployment fails.

Modifying Dependent Objects

When you modify a child object, you may cause a parent object to become invalid. For example, if you modify a mapping by changing a port datatype to an incompatible datatype, the session may become invalid.

A repository object can have a valid, invalid, or impacted state. The Repository Service assigns valid and invalid states when you save an object or when you validate an object. The Repository Service assigns an impacted state when it fetches a parent object of a child object modified in a way that may cause

invalidation. The impacted state is an indeterminate state that is resolved when you validate or save an object.

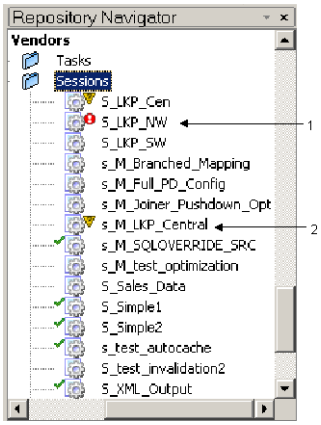
When you modify repository objects, the Repository Service assigns one of three states to the modified object and dependent parent object.

The following table describes the object validation states:

Object State	Assigned	Running the Workflow
Valid	When you save or validate an object.	The object is valid, and workflows run. You do not need to modify the state.
Invalid	When you save or validate an object.	The object is invalid, and workflows will not run. Use the message displayed by the Cloud Data Integration for PowerCenter Client to determine the cause of the invalidation. Modify and validate the object again.
Impacted	If you modify a child object in such a way that it may cause the parent object to become invalid, the Repository Service marks parent objects as impacted. The Repository Service marks the object as impacted when it fetches the parent object.	The object is impacted, and you can perform validation or configure the Integration Service to run the impacted session.

The Repository Service marks dependent objects and shortcuts to parent objects in other folders with warnings to denote the impacted status. A question mark icon denotes impacted status in the Navigator. The Repository Service marks the most immediate parent object as impacted, but it does not mark all related objects as impacted. For example, if you modify a mapping, the Repository Service marks the session as impacted, but it does not mark the workflow as impacted.

The following figure shows invalid and impacted objects:



1. Invalid object.
2. Impacted object.

You can validate impacted sessions, or you can choose to ignore the warning and run the session. To ignore the warning, you must configure the Integration Service to run impacted sessions. At run time, the Integration Service validates the session and determines if the session is valid or invalid. The Integration Service will not run an impacted session if it detects that the session is invalid.

Example

For example, a mapping in Folder A contains a shortcut to an Expression transformation in Folder B. In Folder B, you update the Expression transformation in a way that causes invalidation. The Repository Service marks the parent mappings in Folder B that use the Expression transformation. The Repository Service also marks the mappings in Folder A that use the shortcut to the Expression transformation with a warning. When you run a session that uses the impacted mappings, the Integration Service validates the mappings. If the mappings are valid, the Integration Service runs the session. If the mappings are invalid, the Integration Service marks the session as invalid and it does not run the session.

Rules and Guidelines for Object Compatibility

The Repository Service treats modified dependent objects as compatible when you perform the following tasks:

- Change datatypes in a source, target, or transformation to a compatible datatype. The Repository Service marks objects as impacted only when you change the datatypes to incompatible datatypes.
- Rename ports in a reusable transformation.
- Add a port in a source, target, or transformation.
- Replace objects such as sources, targets, mapplets, and mappings with compatible objects.

When you replace a repository object with another object, the following conditions must be true for the Repository Service to treat the objects as compatible:

Repository Object	Compatibility Requirements
Source, Target, Transformation	- Name of the replacement object must match the original object. - All port names in the original object must be represented in the replacement object. - Datatypes must be compatible.
Mapping	Name and number of the following objects must match the original object: - Targets - Mapplets - Sources - Source Qualifiers - Joiner transformations - Update Strategy transformations - Custom transformations
Mapplet	Name and number of the following objects must match the original object: - Sources - Source Qualifiers - Joiner transformations - Update Strategy transformations - Custom transformations

Administering Repositories

Use the Administrator tool and the *pmrep* and *infacmd* command line programs to administer repositories. Use the Repository Manager and the *pmrep* command line program to manage folders and to manage permissions for folders and global objects.

The Repository Service creates and updates the repository tables. These tables store metadata that the Integration Service and the CDI-PC Client use.

Warning: The CDI-PC repository tables have an open architecture. Although you can view the repository tables, never manually edit them through other utilities. Informatica is not responsible for corrupted data if a customer alters the repository tables or the data within those tables.

Use the Administrator tool to configure security and to copy, back up, delete, and restore repository content. You can back up the repository to a binary file. You can restore the entire repository from a binary file. You can also copy all the repository tables from another database.

Creating the Repository

Before you create a repository, you need a database for the repository tables. You use the database management system client to create the database. The repository database name must be unique.

After you create a database for the repository, you can use the Administrator tool to create a Repository Service to manage the repository. When you create the Repository Service, you can create the database tables for the repository. Alternatively, you can create the Repository Service without creating any database tables. You can create the repository tables later or use existing database tables for the repository. The repository name is the same as the name of the Repository Service.

Creating Folders

After you create the repository, you can add folders to it in the Repository Manager. Use folders to organize repository objects. You can separate different types of metadata and projects into easily identifiable areas. You can configure a folder to be shared so that its content is available to all other folders in the same repository. If you plan to use the same object in several projects, store it in a shared folder.

For example, you use a shared folder to store a definition of the CUSTOMERS table, which provides data for a variety of projects. You make shortcuts to the table in other folders in the same repository. If you are working in a repository domain, you can also make shortcuts to the CUSTOMER table in folders in local repositories that are registered with the repository domain.

Related Topics:

- "PowerCenter Repository Domains" on page 25
- "Creating, Editing, Deleting, and Refreshing Folders " on page 50

Security

You manage users, groups, privileges, and roles on the Security page of the Administrator tool. The Service Manager stores users and groups in the domain configuration database and copies the list of users and groups to the PowerCenter repository. The Service Manager periodically synchronizes the list of users and groups in the repository with the users and groups in the domain configuration database.

When you assign privileges and roles to users and groups for the Repository Service in the Administrator tool or when you assign permissions to users and groups in the PowerCenter Client, the Repository Service stores the privilege, role, and permission assignments with the list of users and groups in the repository.

You manage permissions for repository objects in the PowerCenter Client. Permissions control access to folders and objects in the repository. Even if a user has the privilege to perform certain actions, the user may also require permission to perform the action on a particular object. If the Integration Service uses operating system profiles, the user that runs the workflow must have permission on the operating system profile that is assigned to the workflow or folder that contains the workflow.

To secure data in the repository, you can create folders in the Repository Manager and assign permissions to them. When you create a folder, you are the owner of the folder by default. The owner has all permissions, which you cannot change. The owner can assign permissions to users, groups, operating system profiles, and others in the repository. If the Integration Service uses operating system profiles, an operating system profile must be assigned to the folder to run workflows.

Related Topics:

- "Managing Object Permissions" on page 55

PowerCenter Repository Domains

You can organize, simplify, and manage the process of developing and maintaining multiple data warehouses and other integration projects by using a repository domain. You connect repositories within the repository domain.

A repository domain consists of a single global repository and any number of local repositories. The global repository is used for storing and reusing shared metadata.

You can save any metadata that you plan to share across repositories in the global repository. Local repositories can then use shortcuts to objects in the global repository shared folders, or you can create copies of objects in unshared folders. You can also copy objects in shared folders.

You can share data and metadata between global and local repositories by copying individual repository objects and entire folders within and between repositories. You can also use the Designer to create shortcuts to reference objects in other repositories.

Reusing Metadata

In a repository domain, you frequently need to share metadata across repositories. PowerCenter provides a mechanism for sharing metadata among multiple repositories.

Sharing metadata can help you save time and reduce work by reusing metadata. It also helps enforce standards for the design of transformations at the enterprise level. For example, a sales division develops a standard calculation for the profitability of each product. This calculation is complex. It is based on variables such as production costs and marketing expenses. Because profitability is important information when investing in product development and building a sales strategy, you need everyone in the organization to use the same calculation. If you share the profitability calculation, you ensure that everyone views the value of each product the same way.

When you develop the component of a mapping that performs this calculation, you might want to reuse it in other mappings, even in other repositories. The profitability calculation might appear in several mappings in the sales division repository. The production, marketing, and accounting divisions might also need to use the same calculation in mappings in their repositories.

Shared Folders

You can configure folders in global and local repositories to be shared. After you designate a folder as shared, you can create shortcuts to objects in that folder. Use shortcuts in any other folder in the repository. If the shared folder is in a global repository, use shortcuts to that folder in any repository in the repository domain.

If a folder is not shared, you cannot create shortcuts to objects in the folder. However, you can still create copies of objects in non-shared folders.

Shared folders are useful when you want to use the same repository objects in several projects within the same repository. For example, each folder within a repository might represent a different development project. However, every project in the repository needs to store bug reports in a consistent format, so you might put the source definition for the BUG_REPORTS table in a shared folder.

Reusing Data

The need to share data is just as important as the need to share metadata. Often, several departments in the same organization need the same information. For example, each department may need to read the same product data from operational sources, perform the same profitability calculations, and format this information to make it easy to review.

If each department reads, transforms, and writes this product data separately, the throughput for the entire organization is lower than it could be. A more efficient approach is to read, transform, and write the data to one central data store shared by all users.

A central data store improves throughput at the level of the entire enterprise. To improve performance further, you might want to capture incremental changes to sources. For example, rather than reading all the product data each time you update the central data store, you can improve performance by capturing the inserts, deletes, and updates that have occurred in the PRODUCTS table since the last time you updated the central data store.

You can format data in a standard fashion with the central data store. For example, you can filter employee data that should remain confidential. You can also display date and time values in a standard format. You can perform these and other data cleansing tasks when you move data into the central data store instead of performing them repeatedly.

Version Control

If you have the team-based development option, you can enable version control for the repository. A versioned repository stores multiple versions of an object. Each version is a separate object with unique properties. PowerCenter version control features allow you to efficiently develop, test, and deploy metadata into production.

During development, you can perform the following change management tasks to create and manage multiple versions of objects in the repository:

- **Check out and check in versioned objects.** You can check out and reserve an object you want to edit, and check in the object when you are ready to create a new version of the object in the repository.
- **Compare objects.** The Repository Manager, Workflow Manager, and Designer allow you to compare two repository objects of the same type to identify differences between them. The PowerCenter Client tools allow you to compare objects across open folders and repositories. You can also compare different versions of the same object.

- **Track changes to an object.** You can view an object history that includes all versions of the object. You can also compare any version of the object in the history to any other version. You can see the changes made to an object over time.
- **Delete or purge a version.** You can delete an object so that it no long appears in the PowerCenter Client. However, you continue to store deleted objects in the repository. If you decide later that you need a deleted object, you can recover it from the repository. When you purge an object version, you permanently remove it from the repository.
- **Use global objects such as queries, deployment groups, and labels to group versioned objects.** Object queries, deployment groups, and labels are global objects that exist at the repository level. When you group versioned objects, you can associate multiple objects into logical categories. For example, you can create a deployment group that contains references to objects from multiple folders across the repository.

Related Topics:

- "Team-Based Development with Versioned Objects" on page 70

CHAPTER 2

Using the Repository Manager

This chapter includes the following topics:

Using the Repository Manager Overview

You can navigate through multiple folders and repositories and perform basic repository tasks with the Repository Manager. Menu items in the Repository Manager are enabled or disabled according to the privileges and permissions you have.

Repository Manager Windows

The Repository Manager can display four main windows: the Navigator window, the Main window, the Dependency window, and the Output window. You can dock and undock the Navigator, Dependency, and Output windows. You can also hide and display the Navigator, Dependency, and Output windows.

In the Navigator window, you can connect to a repository, navigate through the folders, and browse repository objects. The Navigator window organizes the repository objects of the same type in each folder in groups called nodes. When you select an object in a node, you can view details for the object in the Main window.

If you configure the Repository Manager to display object dependencies, the Dependency window displays the dependency details when you select an object in the Navigator window. You can view dependency information for sources, targets, mappings, and shortcuts.

The Output window displays detailed information for complex repository operations, such as copying folders. The Output window also displays Repository Service notification messages.

Note: Because the status of the repository changes as users access it, refresh the view of the repository before performing tasks, such as deleting a folder or unlocking an object.

Repository Manager Tasks

Use the Repository Manager to complete the following tasks:

- **Add domain connection information.** You can configure domain connection information.
- **Add and connect to a repository.** You can add repositories to the Navigator window and client registry and then connect to the repositories.
- **Work with PowerCenter domain and repository connections.** You can edit or remove domain connection information. You can connect to one repository or multiple repositories. You can export repository connection information from the client registry to a file. You can import the file on a different machine and add the repository connection information to the client registry.
- **Change your password.** You can change the password for your user account.
- **Search for repository objects or keywords.** You can search for repository objects containing specified text. If you add keywords to target definitions, use a keyword to search for a target definition.
- **View object dependencies**. Before you remove or change an object, you can view dependencies to see the impact on other objects.
- **Compare repository objects.** In the Repository Manager, you can compare two repository objects of the same type to identify differences between the objects.
- **Truncate session and workflow log entries.** You can truncate the list of session and workflow logs that the Integration Service writes to the repository. You can truncate all logs, or truncate all logs older than a specified date.
- **Exchange metadata with other business intelligence tools.** You can export metadata to and import metadata from other business intelligence tools, such as Cognos ReportNet Framework Manager.

Understanding the Repository Manager Windows

The Repository Manager has a main window and a status bar for information about the operation you are performing. The Repository Manager can display the following windows:

- Navigator
- Main
- Dependency
- Output

When the Repository Manager accesses the repository, the status bar reflects the connection in progress with a progress indicator.

Displaying Windows

You can dock and undock the following windows in the Repository Manager:

- Navigator
- Dependency
- Output

Docking or Undocking a Window

To dock or undock a window:

▶ Double-click the title bar. Or, drag the title bar toward the Main window.

The windows that the Repository Manager displays depend on the tasks you perform. When you launch the Repository Manager, the Navigator and the Main windows appear. The Dependency window appears when you want to view dependencies, and the Output window appears when the Repository Manager displays status messages. You can configure the Repository Manager to display or hide any window.

Displaying a Window

To display a window:

1. Double-click the title bar.
2. From the menu, choose View. Then select the window you want to open.

Closing a Window

To close a window:

▶ Click the small x in the upper-right corner of the window.

Navigator Window

Use the Navigator window to connect to a repository and navigate through the folders and objects in the repository. The Navigator window displays the following types of objects:

- **Repositories.** PowerCenter repositories can be standalone, local, or global.
- **Deployment groups.** Deployment groups contain collections of objects for deployment to another repository in the repository domain.
- **Folders.** Folders can be shared or not shared.
- **Nodes.** Nodes contain sessions, sources, targets, transformations, mapplets, workflows, tasks, worklets, and mappings.
- **Repository objects.** Repository objects displayed in the Navigator can include sources, targets, transformations, mappings, mapplets, sessions, tasks, workflows, worklets, workflow logs, and session logs.

Viewing Properties

You can view object properties in the navigator. You can also view license and repository version information.

To view object properties:

1. Connect to a repository.
2. Click on an object in the Navigator.
3. Click the Properties button in the toolbar.

   Tip: You can also right-click the object in the Navigator and select Properties from the shortcut menu.
4. If the object is a repository, click the General tab to view repository version and license information.

   Note: If you enable versioning when you create the repository, you can view all tabs on the Properties dialog box.

Related Topics:

- "Viewing Version Properties" on page 73

Main Window

The Main window of the Repository Manager displays information about the object selected in the Navigator. For example, if you select a repository in the Navigator, the Main window displays all the folders in the repository along with additional folder information, such as whether the folder is shared or in use.

Sorting and Organizing

You can sort items in the Main window by each of the columns. For example, to sort mappings by validity, select the mappings node, and then click the Valid column heading. Click the heading again to reverse the order in which the mappings are sorted.

You can also change the order in which the columns appear. For example, you might want the Valid column to appear first, on the left side of the Main window. To do this, drag the Valid column heading to the location. The Repository Manager displays the columns in the new order until you change the display.

Note: You can double-click an object in the Main window to view its properties.

Viewing Object Details

To view information about repository objects, select a node in the Navigator. Or, to view detailed information about a particular repository object, drill down on a node and select the object.

The following table describes the object details displayed in the Main window:

Node	Information Displayed
Repository Node	Displays properties for each folder in the selected repository.
Deployment Groups Node	Displays properties for each deployment group in the selected repository. Select a static deployment group to view details for deployment group objects.
Sources Node	Displays the properties for each source within the selected node. Select a source definition to view details for each port in the selected source definition.
Targets Node	Displays the properties for each target within the selected node. Select a target definition to view details for each target definition port.
Transformations Node	Displays the properties for each reusable transformation in the selected node. Select a transformation to view details for the specified transformation ports.
Mapplets Node	Displays the properties for each mapplet in the selected node. Select a mapplet to view the Transformations node containing the mapplet.
Mappings Node	Displays the properties for each mapping in the node. Select a mapping to view Sources, Targets, and Transformations nodes that contain the sources, targets, and transformations used in the mapping. Select a target in a mapping to view details for each port in the selected target definition.
Tasks Node	Displays properties for each task in the selected node. Select a task to view the task details.

Node	Information Displayed
Sessions Node	Displays properties for each session in the folder. Select a session to view Session Logs, Source Connections, and Target Connections nodes for the selected session. The Main window also displays information about pre- and post-session email and commands.
Worklets Node	Displays properties for each worklet in the folder. Select a worklet to view the nodes for sessions, tasks, and other objects associated with the worklet.
Workflows Node	Displays properties for each workflow in the folder. Select a workflow to view information for tasks and objects associated with the selected workflow.
Workflow Logs Node	Displays workflow log information for the selected workflow. The Workflow Logs node appears under each workflow in the Workflows node.
Session Logs Node	Displays session log information for the selected session. The Session Logs node appears under each session in the Sessions node.
Source Connections Node	Displays connection properties for each source connection associated with the session. The Source Connections node appears under each session in the Sessions node and under each session associated with a workflow under the Workflows node.
Source File Connections Node	Displays properties for each source file associated with the session. The Source File Connections node appears under each session in the Sessions node and under each session associated with a workflow under the Workflows node.
Target Connections Node	Displays connection properties for each target connection associated with the session. The Target Connections node appears under each session in the Sessions node and under each session associated with a workflow under the Workflows node.
Target File Connections Node	Displays properties for each target file associated with the session. The Target File Connections node appears under each session in the Sessions node and under each session associated with a workflow under the Workflows node.
Transformation Logs Node	Displays log details for each transformation in the session when you select the Transformation Logs node.

Dependency Window

The Dependency window appears when you configure the Repository Manager to display dependencies. You can view dependencies by using the menu items or the dependency buttons on the toolbar. You can also view dependencies using the Dependency dialog box.

When you view dependencies, the left pane of the Dependency window lists the object that has dependency information, and the dependency information appears in the right pane.

The Dependency window can display the following types of dependencies:

- **Source-target dependencies.** When you view source-target dependencies, the Dependency window lists all sources or targets related to the selected object, including relevant information about those sources or targets. For example, if you select a target, you view all sources that provide data for that target, along with information about each source.
- **Mapping dependencies.** When you view mapping dependencies, the Dependency window lists all mappings containing the selected object, and relevant information about those mappings. For example, if you select a reusable transformation, you view all mappings using that transformation and information about each mapping.

- **Shortcut dependencies.** When you view shortcut dependencies, the Dependency window lists all shortcuts to the selected object and relevant details, such as the repository and folder in which the shortcut exists. When you edit an object referenced by shortcuts, use the Repository Manager to see which folders contain the shortcut.

When you open the Dependency window, it displays dependencies for the object selected in the Navigator.

You can also view mapping object dependencies in the Designer. You can view dependencies for sources, targets, transformations, mappings, mapplets, and shortcuts in the Designer.

Related Topics:

- "Viewing Object Dependencies" on page 40

Output Window

The Repository Manager displays status messages in the status bar. For complex operations, the Repository Manager displays detailed information in the Output window.

For example, when you connect to a repository, the status bar displays the following message:

```
Accessing the repository...
```

After you connect to the repository, the status bar displays the word Ready.

When you perform a complex operation, such as copying a folder, the Repository Manager displays details about the operation in the Output window.

The Repository Manager receives notifications when folders are created, modified, or deleted. You must be connected to the repository to receive notifications about folders.

You can change the font type and size displayed in the output window by right-clicking the window and selecting Change Font.

Configuring Repository Manager Options

Use the Options dialog box of the Repository Manager to configure general options. Click Tools > Options to access the general options, which relate to saving Metadata Exchange (MX) data and adding to deployment groups.

The following table describes the general options:

Option	Description
Prompt User While Adding to Deployment Group	Displays the Dependency for Deployment Group dialog box when you add objects to a static deployment group. If you clear this check box, the Repository Manager adds all child objects when you add an object to a static deployment group.
Save All MX Data	Saves all MX data when you use the Repository Manager to import mappings. You can then access the data in MX views to analyze repository metadata or to integrate with third-party repository tools. Default is disabled.
Save Only Source/Target Dependencies	Saves only the MX data related to source/target dependencies when you use the Repository Manager to import mappings. Select this option if you use the Repository Manager to view source/target dependencies, but you do not need to view expressions of fields in MX views. Default is disabled.

Note: Saving MX data can impact performance. Select this option only if you intend to use MX views.

You can also configure the Designer to save MX data when you save mappings in the Designer. The MX data option in the Repository Manager controls the behavior of mapping imports in the Repository Manager only. It does not affect the behavior of the Designer.

When you save MX data for mappings, PowerCenter creates a field expression for each target field in the mappings. The field expression describes the source definition and transformation expression corresponding to the target field. In addition to viewing the MX data in MX views, you can view the field expressions in the Main window of the Repository Manager when you analyze source-target dependencies.

Connecting to Domains and Repositories

Each repository belongs to a PowerCenter domain. You connect to a repository through the domain. Before you can initially connect to a repository, you must provide connection information for the domain. You also need to add the repository to the Navigator in the PowerCenter Client.

Complete one the following tasks before initially connecting to a repository:

- Configure the domain connection information first, and then add a repository.
- Add a repository to the Navigator, and then configure the domain connection information when you connect to the repository.

Configure the domain connection information first if you need to add multiple repositories to the Navigator.

After you create a domain connection, you may need to update or remove it.

You manage PowerCenter domains and repositories in the Administrator tool.

Configuring a Domain Connection

You add domain connection information to the PowerCenter Client so that you can connect to repositories in the domain. After you add a domain connection, you can select from a list of associated repositories to add to the Navigator.

Note: You can also enter domain connection information when you connect to a repository.

To configure a domain connection and add repositories to the Navigator:

1. In a PowerCenter Client tool, select the Repositories node in the Navigator.
2. Click Repository > Configure Domains to open the Configure Domains dialog box.
3. Click the Add button.

   The Add Domain dialog box appears.
4. Enter the domain name, gateway host name, and gateway port number.

   Note: Use the gateway HTTP port number to connect to the domain from the PowerCenter Client. You cannot connect to the domain using the HTTPS port number.
5. Click OK to add the domain connection.

   After you add a domain connection, you can add repositories to the Navigator by selecting them in the list of associated repositories.

   Note: The list of associated repositories might change if a user adds or deletes a Repository Service in the Administrator tool while you are working in the PowerCenter Client. You refresh the list of associated repositories each time you click a domain other than the currently selected domain in the left panel. Click Refresh to update the list of associated repositories for the currently selected domain.
6. If you need to add repositories to the Navigator, complete the following steps:
   - Click a domain name in the left panel of the Configure Domains dialog box.
   - In the right panel, select the repositories to add to the Navigator.
   - Click OK.

Related Topics:

- "Connecting to a Repository" on page 35

Adding a Repository to the Navigator

Add a repository when a repository exists but does not appear in the Navigator. You can add a repository in the Repository Manager, the Designer, the Workflow Manager, or the Workflow Monitor. After you add a repository in one of the PowerCenter Client tools, it appears in the Navigator window of all the tools.

To add a repository to the Navigator:

1. In any of the PowerCenter Client tools, click Repository > Add.
2. Enter the name of the repository and a valid user name.
3. Click OK.

   The repository appears in the Navigator of the PowerCenter Client tools. Before you can connect to the repository for the first time, you must configure the connection information for the domain that the repository belongs to.

Connecting to a Repository

Before you can connect to a repository, you must first add the repository to the Navigator.

To connect to a repository:

1. Launch a PowerCenter Client tool.
2. Select the repository in the Navigator and click Repository > Connect, or double-click the repository.

   The Connect to Repository dialog box appears.
3. Enter the user name and password.

4. Select Native or the name of a specific security domain.

   The Security Domain field appears when the PowerCenter domain contains an LDAP security domain. If you do not know the security domain that your user account belongs to, contact the PowerCenter domain administrator.

5. To connect to a repository that you have connected to before, go to step 12.

   To select an existing domain connection for the repository, go to step 10.

   To enter the domain connection information for the repository, complete steps 6 through 9.

6. If the Connect to Repository dialog box is not expanded, click More to view the connection settings.
7. Click Add.

   The Add Domain dialog box appears.

8. Enter the domain name, gateway host name, and gateway port number.

   Note: Use the gateway HTTP port number to connect to the domain from the PowerCenter Client. You cannot connect to the domain using the HTTPS port number.

9. Click OK.
10. If the connection settings of the Connect to Repository dialog box are hidden, click More to view the connection settings.
11. In the connection settings, select the appropriate domain connection from the list.
12. Click Connect.

Refreshing Repository Objects

You can refresh the repository folder list or a folder to reflect its latest changes. When you refresh a folder, its contents are refreshed.

To refresh a folder, right-click the open folder, and then select Refresh.

To refresh the repository folder list, right-click the repository, and then select Refresh Folder List.

Reconnecting to a Repository and Canceling Auto-Reconnect

After you connect to a repository, the PowerCenter Client can maintain the connection when a temporary network or hardware failure occurs. The ability to maintain the connection is called resilience.

If you perform a PowerCenter Client action that requires a connection to the repository while the PowerCenter Client is trying to reestablish the connection, the PowerCenter Client prompts you to try the operation again after it reestablishes the connection. If the PowerCenter Client is unable to reestablish the connection during the resilience timeout period, the PowerCenter Client prompts you to reconnect to the repository manually.

The PowerCenter Client resilience timeout controls how long the client attempts to reconnect to the repository after the connection is interrupted. The PowerCenter Client resilience timeout is 180 seconds and is not configurable.

In the Designer, Workflow Manager, and Repository Manager, you can temporarily disable PowerCenter Client resilience to prevent the client from attempting to reestablish a repository connection during the resilience timeout period. If you do not want to wait for the resilience timeout to expire, cancel the automatic reconnection. Then, you need to manually connect to the repository again.

After the resilience timeout expires or you cancel the automatic reconnection, you must reconnect to the repository to save changes that you made before the repository connection failed.

To cancel automatic reconnection:

1. Verify that the PowerCenter Client is attempting to reestablish the repository connection.

   Execute an operation that requires a repository connection. If the resilience timeout has not expired, the PowerCenter Client prompts you to retry the operation after the connection is reestablished.

2. Click Repository > Cancel Auto-Reconnect.

   The PowerCenter Client stops attempting to reconnect. To perform actions that require a repository connection, you must manually connect to the repository.

Managing Domain and Repository Connections

You may need to modify or remove domain connection information that is outdated. Similarly, you may need to manage connections to individual repositories. You can connect to multiple repositories at one time. You can perform the following tasks to manage domain and repository connections:

- Edit domain connection information.
- Remove domain connection information.
- Export and import repository connection information.
- Remove a repository from the Navigator.

Editing a Domain Connection

Edit domain connection information when you need to change a gateway host name or gateway port number for a domain connection.

To edit a domain connection:

1. Click Repository > Configure Domains.
2. In the Configure Domains dialog box, select a domain and click the Edit button.

   The Edit Domain dialog box appears.

3. Enter a new gateway host name and gateway port number, if necessary.
4. Click OK.

   Note: You can also edit a domain connection when you connect to a repository.

Removing a Domain Connection

You can remove a domain connection from the PowerCenter Client.

To remove a domain connection:

1. Click Repository > Configure Domain.
2. In the Configure Domains dialog box, select the domain connection that you want to delete.
3. Click the Remove button.
4. In the warning message that appears, click Yes.

   Note: When you remove a domain connection, you terminate connectivity between the PowerCenter Client and all services in the domain. To restore connectivity, add the domain connection, and then add repositories.

Exporting and Importing Repository Connection Information

The Repository Manager saves repository connection information in the registry. To simplify the process of setting up client machines, you can export that information and then import it to a different client machine. Both machine must use the same operating system. The section of the registry you can import and export contains the following repository connection information:

- Repository name
- User name and 7-bit ASCII password
- Security domain
- Gateway host name and port number

Exporting Repository Connection Information

To export the repository connection information from the registry:

1. In the Repository Manager, click Tools > Export Registry.
2. Enter the name of the export file.

   To identify the file, use a file name with the extension.reg, such as MySources.reg.
3. Click OK.

   A dialog box appears, informing you that the Repository Manager successfully exported the repository registry file.

You import this file on other client machines with the same operating systems.

Importing Repository Connection Information

To import the repository connection information to the registry:

1. In the Repository Manager, click Tools > Import Registry.
2. Navigate to the directory containing the import file and select it.
3. Click Open.

Removing a Repository from the Navigator

You can remove a repository from the Navigator of the PowerCenter Client. You might need to remove a repository from the Navigator if the repository no longer exists or if you no longer use the repository.

To remove a repository from the Navigator:

1. In the Navigator of a PowerCenter Client tool, select the repository you want to remove.
2. Press Delete.

After you remove a repository from the Navigator, you can also remove it from the PowerCenter Client registry.

Removing a Repository from the PowerCenter Client Registry

To remove a repository from the PowerCenter Client registry:

1. In the Navigator of the Repository Manager, click Tools > Remove Repository.
2. Select the repository you want to remove from the registry, and click Delete.
3. In the message box that appears, click OK to remove the repository.

Changing Your Password

If the Informatica domain uses native user authentication, the Informatica domain stores your user account credentials. You can use the PowerCenter Repository Manager or the Administrator tool to change your password.

Note: If you change your password, you must update environment variables or *pmcmd* or *pmrep* script files that use the password. Replace the existing password with the new password.

If the Informatica domain uses LDAP or Kerberos authentication, you must log in with your network user account. The network authentication server stores your user account credentials. You can change your password on the network based on the account password rules of your organization.

1. In the Repository Manager, connect to the repository.
2. Click **Security** > **Change Current Password**.
3. Enter the old password.
4. Enter the new password twice to confirm it.

   The password can be between 1 and 80 characters long.
5. Click OK.

Searching for Repository Objects

In the Repository Manager, you can search for repository objects using the following methods:

- Keyword search
- Search text

Perform a keyword search when you have associated a keyword with a target definition. Use Search All when you want to search through text in object names and comments associated with the repository object.

Performing Keyword Searches

After you add keywords to target definitions, use them to perform searches in the Repository Manager.

To search for targets containing a keyword:

1. In the Repository Manager, connect to a repository.
2. Click Analyze > Search by Target Keyword.

   The Search Target Tables For Keyword dialog box appears.

   You can enter a keyword, or you can select one from the list of all keywords applied to target definitions in the repository.
3. If you want to select a keyword, click List Keywords, select a keyword, and then click OK.

   Tip: You can also enter a keyword in the Keyword field.

4. Select the options you want to use in the search:

Option	Description
Exact Match	If selected, the Repository Manager looks for the entire keyword entered. If cleared, the Repository Manager looks for keywords that contain the keyword entered. For example, if you enter REPOS without selecting the Exact Match option, the search would return a keyword match for REPOSITORY.
Ignore Case	If selected, the Repository Manager does not try to match the case of the keyword entered. Otherwise, the keyword search is case sensitive.

5. Click OK.

   The Repository Manager searches for matching keywords in all target definitions in the repository. A list of matching keywords appears at the bottom of the Repository Manager window.

Searching All Repository Objects

Use Search All to search for text in the object name, comments, group name, and owner name associated with repository objects. You can search for text in the repository objects such as transformations, source and target fields, and tasks.

To search for text in all repository objects:

1. In the Repository Manager, connect to the repository.
2. Click Analyze > Search All.

   The Search All dialog box appears.
3. Enter the text you want to find.
4. Select Match Case if you want to perform a case-sensitive search.
5. In the item list, select the repository objects in which you want to search for the text. The Repository Manager selects all objects by default.
6. Select the repository object attributes in which you want to search for the text. The Repository Manager selects all attributes by default.
7. Click Find Next.

   The Repository Manager displays the first repository object matching the search text in the Main window.
8. Click Find Next to find the next repository object matching the search text.
9. Click Close to close the Search All dialog box.

Viewing Object Dependencies

Before you change or delete repository objects, you can view dependencies to see the impact on other objects. For example, before you remove a session, you can find out which workflows use the session. You can view dependencies for repository objects in the Repository Manager, Workflow Manager, and Designer tools.

In the Repository Manager, Workflow Manager, and Designer, you can view dependencies when you perform the following tasks:

- **View object history**. You can view dependencies from the View History window when you view the history of an object.
- **View checkouts**. You can view dependencies from the View Checkouts window when you search for persistent checkouts.
- **View query results**. You can view dependencies from the Query Results window when you search for object dependencies or run an object query.

Note: You can check in objects from the View Dependencies dialog box, but you cannot check in original objects for global shortcuts.

In addition, you can view dependencies from the tools in the Workflow Manager and Designer. For user-defined functions, you can right-click the function or click Tools > User-Defined Functions > Dependencies to view dependencies in the Designer.

The following table lists the tools used to display dependencies and the objects that View Dependencies displays when you view dependencies:

Parent Object	Tool	Types of Child Objects Displayed
Mappings	Mapping Designer	- Global and local shortcuts to the mapping. - Sources the mapping uses. - Targets the mapping uses. - Transformations the mapping uses.
Mapplets	Mapping Designer Mapplet Designer	- Global and local shortcuts to the mapplet. - Sources the mapplet uses. - Targets the mapplet uses. - Transformations the mapplet uses.
Sources	Mapplet Designer Mapping Designer Source Analyzer	- Sources within the same folder that reference or are referenced by the source through a foreign key relationship. - Global and local shortcuts to the source.
Targets	Mapping Designer Target Designer	- Targets within the same folder that reference or are referenced by the target through a foreign key relationship. - Global and local shortcuts to the target.
Transformations	Mapplet Designer Mapping Designer Transformation Developer	- Global and local shortcuts to the transformation.
User-Defined Functions	-	- User-defined functions that use the user-defined function. - Transformations that use the user-defined function. - Mappings with transformations that use the user-defined function - Workflow tasks that use the user-defined function. - Worklets that use the user-defined function. - Workflows that use the user-defined function.

Parent Object	Tool	Types of Child Objects Displayed
Sessions	Task Developer Worklet Designer Workflow Designer	- Sources the session uses. - Targets the session uses. - Mappings the session uses. - Mapplets the session uses. - Transformations the session uses. - Tasks the session uses. - Mapping the session uses. - Session configuration the session uses.
Workflows	Workflow Designer	- Sources the workflow uses. - Targets the workflow uses. - Mappings the workflow uses. - Mapplets the workflow uses. - Transformations the workflow uses. - Sessions the workflow uses. - Tasks the workflow uses. - Schedulers the workflow uses. - Session configuration the workflow uses. - Worklet the workflow uses.
Worklets	Worklet Designer Workflow Designer	- Sources the worklet uses. - Targets the worklet uses. - Mappings the worklet uses. - Mapplets the worklet uses. - Transformations the worklet uses. - Sessions the worklet uses. - Tasks the worklet uses. - Schedulers the worklet uses. - Session configuration the worklet uses.

Note: You can perform all searches in this table from the Repository Manager.

You can view object dependencies when you open objects in the following tools:

- Mapping Designer
- Mapplet Designer
- Workflow Designer
- Worklet Designer

The Dependencies dialog box displays the object name, the object type, and a list of dependent objects and their properties.

When you search for dependencies, you can filter the results by selecting options and object types.

The following table shows the options you can select when you search for dependencies:

Option	Description
Primary/Foreign Key Dependencies	View primary and source object dependencies where there is a primary key-foreign key relationship.
Global Shortcut Dependencies	View global shortcuts across repositories. You can select this option when you search for parents, children, or primary key-foreign key dependencies.
All Children	View the objects that the selected object uses. For example, if you search for the child objects for a workflow, the results might include sessions and worklets.
All Parents	View the objects that use the selected object. For example, if you search for the parent objects for a session, the results might include a workflow or worklet.

When you search for dependencies, the View Dependencies window displays the properties for each dependent object.

The following table describes the object properties that appear in the View Dependencies window:

Properties	Description
Object Name	Name of the dependent object.
Group Name	DBD associated with the source of the object. For example, the group type can be Oracle, DB2, or XML.
Object Type	The type of dependent object. Dependent objects can be any of the following types: - Foreign key dependency - Shortcuts - Mappings - Mapplets - Sessions - Workflows - Worklets - Target definition - Source definition
Version	Version number of the dependent object.
Time Stamp	Time the object was created or saved.
Status	Status of the object, Active or Deleted.
Version Comments	Comments associated with the dependent object.
Folder Name	Folder name where the dependent object resides.
User Name	User who created the object.
Host Name	Host name for the machine hosting the object.
Checkout Type	Type of checkout for object, Persistent or Non-Persistent.

Properties	Description
Purged by User	Name of the user who purged the object.
Is Reusable	Status of object as reusable, Yes or No.
Is Deleted	Status of object deletion, Yes or No.
Repository	Repository hosting the object.

The View Dependencies window also displays output, comments, and label information associated with the object. The Output window displays validation information, and the Comments window displays text entered during object check in or check out. The Labels window displays labels associated with the object and information associated with the label object.

To save the list of dependent objects as an HTML file, click File > Save to File.

Validating Multiple Objects

You can validate multiple objects in the repository without fetching them into the workspace. You can save and optionally check in objects that change from invalid to valid status as a result of the validation. You can validate sessions, mappings, mapplets, workflows, and worklets.

You can select objects to validate from the Navigator window of the Repository Manager. You can also select objects from query results or an object dependencies list. If you select objects from the Navigator, you must select objects of the same type in the same folder. If you select objects from query results or an object dependencies list, you can choose different types of objects to validate.

To validate multiple objects:

1. Select the objects you want to validate.
2. Initiate the validation.

   If you select objects from query results or a list view, you must right-click one of the selected objects and select Validate. If you are selecting objects from the Navigator, you can also click Tools > Validate to initiate a validation. The Validate Objects dialog box appears.
3. Select validation options from the Validate Objects dialog box:
   - **Save validated objects.** If you do not choose to save the objects in the repository, the validation just provides a report.
   - **Choose whether to check in validated objects.** You can check in valid objects. You can select this option if you have selected the Save Objects that Are Successfully Made Valid option and versioning is enabled in the repository.
   - **Check in comments.** If you are checking in validated objects, you must enter check-in comments. Otherwise, this option is disabled.
4. Click Validate.

   The validation displays status in a progress dialog box. The output window shows the status of each object being validated. You can cancel the validation from the progress dialog box. If you cancel, you do not cancel the current operation, but you cancel the remaining operations in the validation.

The Validate Objects results box appears when validation completes. Use the results box to determine how many objects were validated and saved. To view the objects in each total, click the link.

The following table describes the information that appears in the Validate Objects window:

Properties	Description
Objects Provided for Validation	Total number of objects that you selected to validate. The total includes skipped objects.
Skipped Objects	Number of the following types of objects: - Objects that do not require validation, such as sources, targets, transformations, and shortcuts. - Objects that cannot be fetched, such as a deleted mapping associated with a session being validated.
Objects that Were Invalid Before the Validation	Number of invalid objects provided for validation.
Objects Successfully Validated	Number of selected objects that are valid.
Objects Still Invalid	Number of objects that require fixing or editing.
Saved/Checked In	Total number of objects saved. If you do not choose the Save Objects or Check In Objects options, this number is zero.
Cannot Save Objects Due to Lock Conflict	Number of validated objects you cannot save because someone has them locked.

5. Click a link to view the objects in the results group.

   Validation results that include objects provide links to the object details. When you click the link, a window displays each object in the results group you select.

Comparing Repository Objects

You can compare two repository objects of the same type to identify differences between the objects. For example, you can compare two sessions to check for differences. When you compare two objects, the Repository Manager displays their attributes.

You can compare objects across folders and repositories. To do this, you must have both the folders open. You can compare a reusable object with a non-reusable object. You can also compare different versions of the same object.

You can compare the following types of objects:

- **Designer objects.** You can compare Designer objects, such as sources, targets, transformations, mapplets and mappings.
- **Workflow Manager objects.** You can compare Workflow Manager objects, such as tasks, sessions, worklets, and workflows. You can compare schedulers and session configuration objects in the Repository Manager, but not in the Workflow Manager.

You can compare instances of the same type in detail. For example, if the workflows you compare contain worklet instances with the same name, you can compare the instances to see if they differ. You can compare instances of sessions and tasks in a workflow or worklet comparison. You can compare instances of mappings and transformations in a session comparison. Further, you can compare instances of the same type within a mapping or mapplet comparison.

You cannot compare objects of different types. For example, you cannot compare a source definition with a target definition.

To compare repository objects:

1. In the Repository Manager, connect to the repository.
2. In the Navigator, select the object you want to compare.

   Tip: To compare the first two objects in a node, select the node.
3. Click Edit > Compare Objects.

   The Mappings dialog box appears.
4. Click Compare.

If you choose a Designer object, a window shows the result of comparing the objects.

If you choose a Workflow Manager object, such as a session, the Diff Tool window shows the result of comparing two sessions. In the Diff Tool Window:

- You can filter nodes that have same attribute values.
- You can compare object instances.
- Differences between objects are highlighted and the nodes are flagged.
- Differences between object properties are marked.
- The Output window displays the properties of the node you select.

Truncating Workflow and Session Logs

When you configure a session or workflow to archive session logs or workflow logs, the PowerCenter Integration Service saves those logs in local directories. The repository also creates an entry each time that a workflow or a session runs. If you move or delete a session log or workflow log from the workflow log directory or session log directory, you can remove the entries from the repository.

Use the Repository Manager or the pmrep TruncateLog command to truncate the workflow logs for workflows that have completed.

You can truncate all logs for a workflow, or logs that were created before a specified date. You cannot truncate a workflow log for a workflow that is still running. The PowerCenter Repository Service truncates the workflow log list and the session log list at the same time.

When the PowerCenter Repository Service truncates logs for sessions and workflows, it also deletes the following run-time information for the sessions and workflows:

- Workflow details
- Session statistics
- Task details
- Source and target statistics

- Partition details
- Performance details

Note: When you truncate logs from a Microsoft SQL Server repository, verify that no workflow is running. If you truncate logs when a workflow is running, the workflow fails.

To truncate workflow and session logs:

1. In the Repository Manager, select the workflow in the Navigator window or in the Main window.
2. Choose **Edit** > **Truncate Log**.

   The **Truncate Workflow Log** dialog box appears.
3. Choose to delete all workflow and session log entries, or to delete all workflow and session log entries with an end time before a particular date.
4. If you want to delete all entries older than a certain date, enter the date and time.
5. Click **OK**.

   The PowerCenter Repository Service deletes the workflow and session log entries from the repository.

CHAPTER 3

Folders

This chapter includes the following topics:

Folders Overview

Folders provide a way to organize and store metadata in the repository, including mappings, schemas, and sessions. Folders help you logically organize the repository. Each folder has a set of properties that you can configure to define how users access the folder. You can verify folder content by comparing folders.

You can perform the following tasks when you work with folders:

- **Configure folder properties.** When you create a folder, you can configure properties such as name, description, and owner.
- **Compare folders.** You can compare the contents of a folder with other folders to verify content. You can compare objects between two folders in the same repository or in different repositories. You can perform folder comparisons before copying or replacing a folder.
- **Manage permissions.** A permission is the level of access a user has to an object. A user with the privilege to perform certain actions can require permissions to perform the action on a particular object.
- **Copy folders.** You can copy a folder and all of its contents within a repository or to another repository.

Managing Folder Properties

When you create a folder, you can configure folder properties and permissions on the folder. For example, you can create a folder and share the objects within the folder with other users in the repository. You can create shared and non-shared folders.

The following table describes the properties that you can configure for each folder:

Folder Properties	Required/ Optional	Description
Name	Required	Folder name. Do not use the period character (.) in folder names. Folder names with periods can cause errors when you run sessions.
Description	Optional	Description of the folder that appears in the Repository Manager.
Owner	-	Owner of the folder. By default, the folder owner is the user who creates the folder. This field is read-only. You can change the owner on the Permissions tab.
OS Profile	Optional	Operating system profile name. If the Integration Service uses operating system profiles, specify an operating system profile for the Integration Service to use.
Allow Shortcut	Optional	Makes the folder shared.
Status	Conditional	Applies the status applied to all objects in the folder. Required for versioned repositories.

Operating System Profile

If the Integration Service uses operating system profiles, specify an operating system profile for the Integration Service to use. The Integration Service uses the operating system profile to run workflows in the folder. Operating system profiles allow the Integration Service to run a workflow and write output files using the setting of the operating system profile. You can use the Start Workflow Advanced option in the Workflow Manager to override the operating system profile assigned to the folder.

When you copy a folder to another repository, the Copy Folder Wizard removes the operating system profile assignment of the target folder. After you copy a folder, you must assign an operating system profile to the target folder.

When you replace a folder, the Copy Folder Wizard retains the operating system profile assignment of the target folder.

The operating system profile assignment for a folder is retained when you backup and restore a repository.

Shortcuts and Shared Folders

You can designate a folder to be shared. In the Designer, shared folders allow users to create shortcuts to objects in the folder. If you have an object that you want to use in several mappings or across multiple folders, you can place the object in a shared folder. You can access the object from other folders by creating shortcuts to the object.

Shortcuts inherit changes to the original object. To use an object in a shared folder without inheriting future changes, you can copy the existing object.

When you create a shared folder, the folder icon in the Navigator displays an open hand icon.

Note: After you make a folder shared, you cannot reverse the change.

Shared Folders in Global Repositories

Shared folders in global repositories can be used by any folder in the domain. For example, if you are working in a folder in a local repository, you can select the global repository and create a shortcut to a shared folder. As with local shared folders, if the original object is changed, all shortcuts reflect those changes.

Creating, Editing, Deleting, and Refreshing Folders

You can perform the following tasks to manage folder properties:

- **Create a folder.** When you create a folder, you become the folder owner by default.
- **Edit a folder.** When you edit a folder, you can edit the properties, change the owner, and configure the permissions.
- **Delete a folder.** If a folder becomes obsolete, you can delete that folder from the repository.
- **Refresh a folder.**You can refresh a folder to reflect its latest changes.

Creating a Folder

To create a folder:

1. In the Repository Manager, connect to the repository.
2. Click Folder > Create.

   The Create Folder dialog box appears.
3. Enter the information for folder properties.
4. Click the Permissions tab.

   Assign permissions on the folder to users and groups.
5. Click OK.

Editing a Folder

To edit a folder:

1. In the Repository Manager, connect to a repository and select a folder.
2. Click Folder > Edit.
3. Enter the changes, and click OK.

Deleting a Folder

To delete a folder:

1. In the Repository Manager, connect to a repository and select a folder.
2. Click Folder > Delete.
3. In the confirmation message that appears, click OK.

Refreshing a Folder

You can refresh a folder to reflect its changes.

1. In the Repository Manager, connect to a repository and select a folder.
2. Click Folder **> Open.**
3. Right-click the folder, and then select Refresh.

Comparing Folders

Before you copy or replace a folder in a repository, you might want to verify the contents of a folder or compare it with other folders. The Repository Manager lets you quickly and accurately compare the objects in different folders using the Compare Folders Wizard.

If you use a versioned repository, the Repository Manager uses the latest checked in version of an object for comparison.

In the Compare Folders Wizard, you can complete the following comparisons:

- Compare objects between two folders in the same repository.
- Compare objects between two folders in different repositories.

You can specify the following comparison criteria for each comparison:

- **Object types to compare.** You can specify the object types to compare and display between folders. The wizard compares objects based on specific object attributes.
- **Direction of comparison.** The wizard performs directional comparisons. A directional comparison checks the contents of one folder against the contents of the other. You can specify either one-way or two-way comparisons.

The wizard displays the following information:

- Similarities between objects
- Differences between objects
- Outdated objects

You can edit and save the result of the comparison.

Related Topics:

- "Compared Attributes and Object Differentiation" on page 51

Compared Attributes and Object Differentiation

The Compare Folders Wizard compares objects based on specific object attributes.

The following table lists the object types and attributes you can compare:

Object Type	Compared Attribute
Sources	Source name and database name
Targets	Target name and database name
Reusable transformations	Transformation name and type
Mappings	Mapping name
Mapplets	Mapplet name
Source fields	Column names
Target fields	Column names

Object Type	Compared Attribute
Reusable transformation fields	Port names
Sessions	Session name
Session components	Component value
Tasks	Task name
Task instances	Task instance name
Workflows	Workflow name
Workflow events	Workflow event name
Workflow variables	Workflow variable name
Worklets	Worklet name
Shortcuts	Shortcut name and object type
Transformation instances	Transformation instance name and type
Mapping variables	Mapping variable name
External procedure initialization properties	Property name
Schedulers	Scheduler name
Configurations	Configuration name

Some objects you choose to compare also cause the wizard to compare other objects, regardless of whether you select the other objects to compare.

The following table lists objects the wizard compares by default when you select certain objects to compare:

Selected Object	Compared Object
Source field	Source
Target field	Target
Reusable transformation field	Reusable transformation
Mapping variable	Mapping
External procedure initialization properties	Transformation instance
Session component	Session
Task Instance	Workflow and worklet

Selected Object	Compared Object
Workflow event	Workflow and worklet
Workflow variable	Workflow and worklet

The wizard compares the attribute of each object in the source folder with the attribute of each object in the target folder. You can choose to compare based on the following criteria:

- **Different objects.** Object name and type exist in one folder but not the other.
- **Similar objects.** Object name, type, and modification date are the same in both folders.
- **Outdated objects.** Object modification date is older than objects with the same name.

The wizard does not compare the field attributes of the objects in the folders when performing the comparison. For example, if two folders have matching source names and column or port names but differing port or column attributes, such as precision or datatype, the wizard does not note these as different.

One-Way and Two-Way Comparisons

Comparison results depend on the direction of the comparison. One-way comparisons check the selected objects of Folder1 against the objects in Folder2. Two-way comparisons check objects in Folder1 against those in Folder2 and also check objects in Folder2 against those in Folder1.

A two-way comparison can sometimes reveal information a one-way comparison cannot. A one-way comparison does not note a difference if an object is present in the target folder but not in the source folder.

For example, you have two folders, ADS1 and ADS2, in the same repository. If you compare the folders using a one-way comparison, the source definition ORDER_ITEMS, which is present in ADS2 but not in ADS1, is not noted as a comparison. If you compare the folders using a two-way comparison, the absence of ORDER_ITEMS in ADS1 is noted as a difference.

Editing and Saving Results Files

You can edit and save the result of a folder comparison. The Compare Folders wizard displays similar objects in green text, unmatched objects denoting a difference in red text, and outdated objects in blue text. The Compare Folders Wizard always displays the total number of differences, similarities, and outdated objects found during the comparison, even if you do not choose to display differences or similarities in the edit field.

You can save the results of the comparison in the edit field to an .rtf or a .txt file. To retain the color and font attributes of the result, save it as an .rtf file. By default the results file is saved in the My Documents directory.

Steps to Compare Folders

Before comparing folders, verify that you have Read permission for each folder you want to compare. Connect to the repositories containing the folders in the wizard.

To compare folders:

1. In the Repository Manager, click Folder > Compare.
2. Click Next.
3. Connect to the repositories containing the folders you want to compare and select the folders for comparison.

4. Click Next.
5. Select the object types you want to compare.
6. Click Next.
7. Select the display options.
8. Click Next.

   The wizard always displays the number of differences, similarities, and outdated objects.
9. View the results of the comparison.
10. If you want to save the comparison results to a file, select Save results to file.
11. Click Finish.
12. If you chose to save the results to a file, specify the file type, name, and directory.
13. Click Save.

CHAPTER 4

Managing Object Permissions

This chapter includes the following topics:

Managing Object Permissions Overview

Permissions control the level of access a user has to an object. In the PowerCenter Client, you can assign permissions on folders and global objects. Global objects include object queries, deployment groups, labels, and connection objects. You can assign the following permissions to users and groups in the repository:

- **Read permission.** You can view the folder and objects.
- **Write permission.** You can create or edit objects in a folder, maintain object queries or labels, or add or delete objects from deployment groups.
- **Execute permission.** You can run or schedule a workflow in the folder, run object queries, apply labels, or copy deployment groups.

When you create a folder or global object, it is created with one user and one default group:

- The user who creates the object is the owner and has read, write, and execute permissions. You can change the owner, but you cannot change the permissions for the owner.
- The default group represents the minimum level of permissions you can assign to any user or group. It displays as “Others” and is created with read permissions. You can grant write and execute permission to the default group. The permissions assigned to the default group are the default permissions that each user and group receives when added to the object user list.

Note: Permissions work in conjunction with privileges. Privileges are actions that a user performs in PowerCenter applications. A user with the privilege to perform certain actions can require permissions to perform the action on a particular object.

Assigned Permissions

Users and groups receive permissions based on the following conditions:

- When you add a user or group to the object list, the user or group receives default permissions. You can increase the level of permissions, but you cannot decrease the level of permissions beyond the level of default permissions.

  For example, the default group has read and write permissions. When you add a user to the object user list, the user receives read and write permissions. You can grant execute permission to the user, but you cannot remove write permission. To remove write permission, you must remove it from the default group.
- Users and groups that are assigned to a group inherit permission of the parent group. Users and groups that inherit permissions from the parent group do not appear in the object user list.
- Users and groups that are assigned the Administrator role for the Repository Service inherit read, write, and execute permissions. You cannot change the permissions for the administrators. Users and groups that inherit permissions from the Administrator role do not appear in the object user list.
- All users and groups that you do not add to the object user list and who do not have the Administrator role for the Repository Service inherit default permissions.

Accessing Object Permissions

Configure permissions for an object in the tool where you create the object.

The following table shows where you configure permissions for folders and global objects:

Object	Configure Permissions
Folders	Repository Manager
Labels	Repository Manager
Deployment Groups	Repository Manager
Object Queries	Repository Manager, Designer, Workflow Manager
Connection Objects	Workflow Manager

You access folder permissions on the Permissions tab. You access permissions for global objects from the object browser.

Managing Permissions

When you manage permissions, you can perform the following tasks:

- **Maintain the object user list.** The user list for an object is the list of users or groups that have permission on the object. You can add and remove users and groups from the list.
- **Assign permissions.** Assign permissions on objects to users, groups, and all others in the repository. You can assign read, write, and execute permissions.
- **Change the object owner.** Change the object owner to any user in the object user list.

Maintaining the User List

Add users and groups to the user list of an object when you want to assign permissions or change ownership. You can also remove users from the user list.

Adding Users and Groups

When you add a user or group to the object user list, you can use the following search filters:

- **Security domain.** Enter the security domain to search for users or groups.
- **Pattern string.** Enter a string to search for users or groups. The PowerCenter Client returns all names that contain the search string. The string is not case sensitive. For example, the string “DA” can return “iasdaemon,” “daphne,” and “DA_AdminGroup.”

When you add a user or group to the object user list, the user or group receives default permissions. After you add a user to the object user list, you can grant permissions or change the ownership. After you add a group to the object user list, you can grant permissions.

To add users and groups:

1. On the Permissions tab of the folder, click Add.
2. Enter the filter conditions to search for users and groups, and click Go.
3. Choose to list users, groups, or all users and groups.
4. Select a user or group, and click Add.

   You can use Ctrl-click to select multiple users or group, or you can use Shift-click to select a range of users and groups.

Removing Users and Groups

When you remove users and groups, you can choose to remove one user or group at a time, or you can remove all users and groups. When you remove all users and groups, you cannot remove the object owner or the default object group.

To remove users and groups:

1. Select a user or group in the user list of the object.
2. To remove a user or group, select the user or group, and click Remove.
3. To remove all users and groups, click Remove All.
4. Click OK.

Assigning Permissions

When you assign permissions to a user or group, you can assign permissions to any user or group in the list. You can filter the list to show users, groups, or users and groups.

Use the following rules and guidelines when you assign permissions:

- You can increase the level of permission for any user or group.

- You cannot revoke default permissions from any user or group. For example, if default permissions are read and write, you cannot remove write permission from any user or group. To reduce the level of permissions, you must change permissions assigned to the default user group, “Others.”

Note: When you change the permissions for a user that is connected to the repository, the permissions take effect the next time the user connects to the repository.

Assign permissions for a folder on the Permissions tab of the folder. Assign permissions for global objects in the global object.

To assign permissions:

1. Select a user or group in the user list of the object.
2. Select the permissions to assign to the user or group.

   If the user or group is not in the list, click Add to add the user or group to the list.
3. Click OK.

Changing the Object Owner

When you change the owner of a folder or global object, you choose from the list of users associated with the object. Use the same filters to search for users that you use to search for users and groups when you add users and groups to the list. If the user does not appear in the list, you can add the user to the list for the object.

To change the object owner:

1. On the Permissions tab of the folder, click Change Owner.

   Or, in the object, click Change Owner.
2. In the Select Owner dialog box, enter the filter conditions to search for users and groups, and click Go.
3. Select a user.

   If the user does not appear in the list to select, use a different filter condition, or click Select Other User to add a user to the list of object users.
4. Click OK.

CHAPTER 5

Local and Global Shortcuts

This chapter includes the following topics:

Local and Global Shortcuts Overview

Shortcuts allow you to use metadata across folders without making copies. This helps to ensure uniform metadata. A shortcut inherits all properties of the object to which it points. After you create a shortcut, you can configure the shortcut name and description.

When the original object changes, the shortcut inherits those changes. By using a shortcut instead of a copy, you ensure each use of the shortcut matches the original object. For example, if you have a shortcut to a target definition and you add a column to the definition, the shortcut inherits the additional column.

Shortcuts allow you to reuse an object without creating multiple objects in the repository. For example, you use a source definition in 10 mappings in 10 different folders. Instead of creating 10 copies of the same source definition, one in each folder, you can create 10 shortcuts to the original source definition.

Note: In a versioned repository, a shortcut inherits the properties of the latest version of the object that it references.

You can create shortcuts to objects in shared folders. If you try to create a shortcut to an object in a non-shared folder, the Designer creates a copy of the object instead.

You can create shortcuts to the following repository objects:

- Source definitions
- Reusable transformations
- Mapplets
- Mappings
- Target definitions

- Business components

You can create the following types of shortcuts:

- **Local shortcut**. A shortcut created in the same repository as the original object.
- **Global shortcut**. A shortcut created in a local repository that references an object in a global repository.

Shortcuts Versus Copies

One of the primary advantages of using a shortcut is maintenance. If you need to change all instances of an object, you can edit the original repository object. All shortcuts accessing the object inherit the changes. In contrast, if you have multiple copies of an object, you need to edit each copy of the object, or recopy the object, to obtain the same results.

However, some changes can invalidate mappings and sessions. For example, if you use a shortcut to a reusable transformation and then change a port datatype, you can invalidate all mappings with shortcuts to the transformation and all sessions using those mappings.

Therefore, if you want the object to inherit changes immediately, create a shortcut. Otherwise, create a copy.

Shortcuts have the following advantages over copied repository objects:

- You can maintain a common repository object in a single location. If you need to edit the object, all shortcuts immediately inherit the changes you make.
- You can restrict users to a set of predefined metadata by asking users to incorporate the shortcuts into their work instead of developing repository objects independently.
- You can develop complex mappings, mapplets, or reusable transformations and then reuse them easily in other folders.
- You can save space in the repository by keeping a single repository object and using shortcuts to that object, instead of creating copies of the object in multiple folders or multiple repositories.

Related Topics:

- "Exporting and Importing Objects" on page 126

Understanding Shortcut Properties

When you create a shortcut to a repository object, the shortcut inherits the attributes of the object. The shortcut inherits the following properties that cannot be edited in the shortcut object:

- Object business name and owner name
- Port attributes, including datatype, precision, scale, default value, and port description
- Object properties

The shortcut also inherits a name and description which you can edit in the shortcut.

Default Shortcut Name

The Designer names a shortcut after the original object based on the original object name as it appears in the Navigator when you create the shortcut. The Designer uses the following naming convention: Shortcut_To_*DisplayedName*.

By default, the Designer displays all objects in the Navigator by object name. If you create a shortcut while using the default display option, the Designer names the shortcut Shortcut_To_*ObjectName*.

Alternatively, you can enable the Designer tools to display sources and targets by business name. When you enable this option, the Designer displays sources and targets by business names if they exist. If you create a shortcut to a source or target when this option is enabled, the Designer names the shortcut Shortcut_To_*BusinessName*.

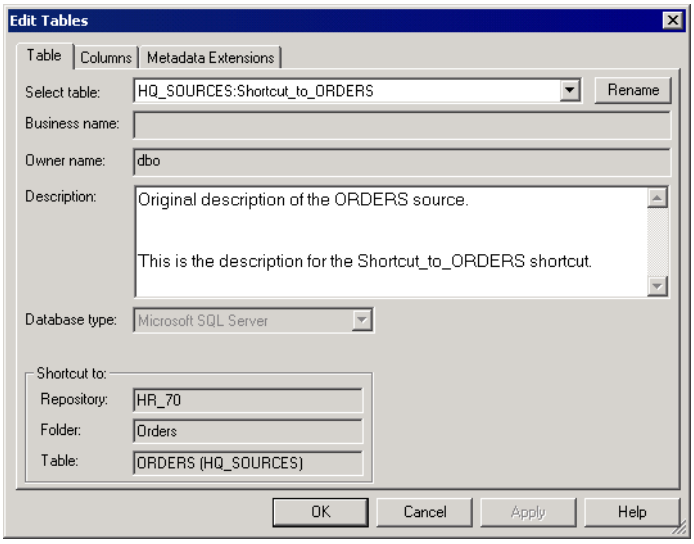
Note: If the business name contains characters that are not allowed as in the object name, the Designer replaces the character with an underscore (_).

You can edit the default shortcut name at any time.

Describing the Object and the Shortcut

Shortcuts inherit the description associated with the referenced object when you first create the shortcut. Afterwards, you can add object descriptions for each shortcut. Since the description is unique to the shortcut, if the description of the referenced object subsequently changes, shortcuts do not inherit those changes.

The following figure shows the shortcut with a description inherited from the original object:



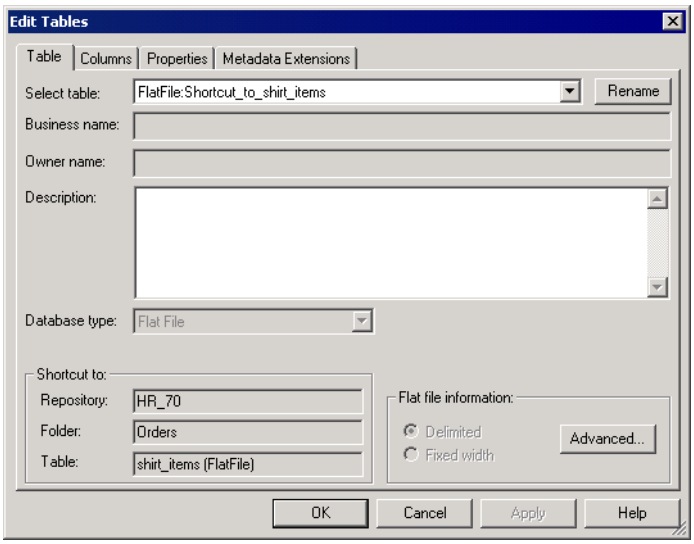
Shortcuts do not inherit edits to the description of the referenced object. However, any shortcuts created after the change contain the latest description.

Locating the Referenced Object

Each shortcut tracks the location of the object it references and displays it in the transformation property sheet. The shortcut object displays the following details about the referenced object:

- Repository name
- Folder name
- Table or transformation name

The following figure shows the referenced shortcut objects:



The original object location displays in the Shortcut To area.

Note: If you move or delete an object referenced by a shortcut, the shortcut becomes invalid.

Creating a Local Shortcut

You can reuse metadata within a single repository by creating a local shortcut. You can create a local shortcut to objects in a shared folder in the same repository. If an object is in a non-shared folder, you can make a *copy* of the object.

You can create a local shortcut in any folder in the repository. After you create a local shortcut, you can reuse it within the same folder. After you create a shortcut, it appears in the Navigator as an available repository object with the shortcut icon. When you drag the resulting shortcut into the workspace, the same shortcut icon appears. The Designer names shortcuts after the original object by default, Shortcut_To_*ObjectName*.

For example, when you create a shortcut to the DetectChanges reusable Expression transformation, the shortcut, named Shortcut_To_DetectChanges appears in the Transformations node of the folder. When you drag it into the workspace, the shortcut icon indicates the transformation is a shortcut.

Note: When you drag an object from a shared folder to a business component directory, the Designer creates a shortcut if the object does not exist in the destination folder.

You can create a local shortcut in the Navigator or in the workspace.

Creating a Local Shortcut in the Navigator

To create a local shortcut in the Navigator:

1. In the Navigator, expand the shared folder containing the object you want to use and drill down to locate the object.
2. Open the destination folder, the folder in which you want the shortcut.
3. Drag the object from the shared folder to the destination folder.

After you drop the object, the Designer displays the following message:

```
Create a shortcut to <object type> <object name>?
```

Note: If the object is not saved in the repository, the Designer displays a message asking if you want to create a copy of the object. To create a shortcut, cancel the operation, save the object, and then create the shortcut.

4. When prompted for confirmation, click OK to create a shortcut.

   The shortcut now appears in the Navigator.

5. Click Repository > Save.

   You can now use the shortcut in this folder.

Creating a Local Shortcut in the Workspace

To create a local shortcut in the workspace:

1. In the Navigator, expand the shared folder containing the object you want to use and drill down to locate the object.
2. Open the destination folder, the folder in which you want the shortcut.
3. Select the appropriate Designer tool for the shortcut.

   For example, to create a shortcut for a source, choose the Source Analyzer or Mapping Designer tool. To create a shortcut for a target, choose the Target Designer or Mapping Designer tool.

4. Drag the object from the shared folder into the workspace.

   After you drop the object, the Designer displays the following message:

   ```
   Create a shortcut to <object type> <object name>?
   ```

   Note: If the object is not saved in the repository, the Designer displays a message asking if you want to create a copy of the object. To create a shortcut, cancel the operation, save the object, and then create the shortcut.

5. When prompted for confirmation, click OK to create a shortcut, or click Cancel to cancel the operation.

   The shortcut now appears in the workspace and in the Navigator.

6. Click Repository > Save.

   You can now use the shortcut in this folder.

Creating a Global Shortcut

You can reuse metadata between repositories by creating a *global* shortcut. A global shortcut is a shortcut in a local repository that references an object in a global repository.

You can create a global shortcut in any folder in the local repository. After you create the global shortcut in a folder, you can reuse it in the folder as you would any other repository object.

You can create a global shortcut to any object in a shared folder in a global repository. If a folder is not shared, you can make a *copy* of these objects if the global and local repository have compatible code pages.

To create a global shortcut, you must be able to connect to the global repository through the local repository. That is, if you connect to the local repository directly, the global repository must appear in the local repository. Similarly, if you connect to the global repository directly, the local repository must appear in the global repository.

You can use the Designer to connect to both the local and the global repositories individually. However, to create a global shortcut, you must connect to one repository through the other. You can connect to the global repository first and then connect to the local repository directly below it to create a global shortcut. Or, you can connect to the local repository and then connect to the global repository that appears below it.

You can create a global shortcut in the Navigator or in the workspace.

Creating a Global Shortcut in the Navigator

To create a global shortcut in the Navigator:

1. In the Designer, connect to the local repository and open the folder in which you want a shortcut.

   The global repository appears in the Navigator below the local repository. If it does not, the repository to which you are connected is not registered with the global repository.
2. In the Navigator, connect to the global repository appearing below the local repository.

   The Designer expands the global repository, displaying folders for which you have permission.

   If the Designer does not allow you to connect to the global repository, you might need to reconfigure aspects of the repository domain.
3. In the global repository, drill down through the shared folder until you locate the object you want to use.
4. Drag the object into the destination folder in the local repository.

   After you drop the object, the Designer displays the following message:

   ```
   Create a shortcut to <object type> <object name>?
   ```

   Note: If the object is not saved in the repository, the Designer displays a message asking if you want to create a copy of the object. To create a shortcut, cancel the operation, save the object, and then create the shortcut.
5. When prompted for confirmation, click OK to create a global shortcut, or click Cancel to cancel the operation.

   The shortcut now appears in the Navigator.
6. Click Repository > Save.

   You can now use the shortcut in this folder.

Creating a Global Shortcut in the Workspace

To create a global shortcut in the workspace:

1. In the Designer, connect to the local repository and open the folder in which you want a shortcut.

   The global repository appears in the Navigator below the local repository. If it does not, the repository to which you are connected is not registered with the global repository.
2. Select the appropriate Designer tool for the shortcut.

   For example, to create a shortcut for a source, choose the Source Analyzer or Mapping Designer tool. To create a shortcut for a target, choose the Target Designer or Mapping Designer tool.
3. In the Navigator, connect to the global repository appearing below the local repository.

   The Designer expands the global repository, displaying folders for which you have permission.

   If the Designer does not allow you to connect to the global repository, you might need to reconfigure aspects of the repository domain.
4. In the global repository, drill down through the shared folder until you locate the object you want to use.
5. Drag the object from the shared folder into the workspace.

After you drop the object, the Designer displays the following message:

```
Create a shortcut to <object type> <object name>?
```

Note: If the object is not saved in the repository, the Designer displays a message asking if you want to create a copy of the object. To create a shortcut, cancel the operation, save the object, and then create the shortcut.

6. When prompted for confirmation, click OK to create a global shortcut, or click Cancel to cancel the operation.

   The shortcut now appears in the workspace and in the Navigator.

7. Click Repository > Save.

   You can now use the shortcut in this folder.

Working with Shortcuts

After you create a shortcut, you can reuse it in the folder.

When you edit an object referenced by a shortcut, the Designer does not validate mappings using shortcuts to the object. Some changes might invalidate mappings, such as deleting a port or changing the port datatype, precision, or scale. When a mapping is invalid, the Integration Service cannot run the session.

When editing a referenced object, use the View Dependencies features in the Repository Manager to determine which mappings contain shortcuts to the object. To ensure mappings are valid, open and validate the mapping. When validating a mapping, make sure you have the most recent version of the mapping.

You can also view object dependencies in the Designer.

Related Topics:

- "Viewing Object Dependencies" on page 40

Refreshing Shortcut Properties

When working with shortcuts, ensure you have the most recent version of the local or global shortcut in the workspace.

If you launch the Designer and then drag into the workspace a shortcut or a mapping or mapplet that uses a shortcut, you view the current version of the object that the shortcut references. However, if another user then edits and saves changes to the referenced object, the shortcut displayed in the workspace is no longer an accurate description of the referenced object. When you work in this type of environment, verify that you have updated the view of local and global shortcuts.

The Integration Service always uses the latest version of a referenced object. When the Integration Service starts a session containing a shortcut, it accesses the repository to retrieve the mapping. If the mapping contains a shortcut, the Integration Service accesses the repository for details about the original object.

Updating Views of Global and Local Shortcuts

The Designer updates properties for a global or local shortcut when it retrieves object information from the repository. If you think the original object referenced by a global or local shortcut has changed, you can refresh the view of the shortcut by performing one of the following:

- **Open metadata.** When you drag an object into the Designer workspace, the Designer retrieves the object from the repository. If the object is a shortcut or contains a shortcut, the Designer retrieves and displays the most recent version of the shortcut.

  For example, if you open a folder to view a shortcut to a source or a mapping using that shortcut, the Designer displays the most recent version of the source.

  Note: When possible, the Designer uses information in memory. If the mapping was open on the client machine, the Designer might display the version in memory rather than accessing the repository for the latest version. To ensure you have the most recent version, perform one of the following tasks in addition to opening metadata.
- **Revert to saved.** When you use the Designer menu command, Edit > Revert To Saved, the Designer accesses the repository to retrieve the last-saved version of the object. If you select a shortcut or a mapping using a shortcut and then click Edit > Revert To Saved, the Designer displays the last-saved version of the object in the workspace.
- **Close the folder or close all tools.** To ensure you have correct shortcut information, you can clear the Designer memory by closing the folder or closing all tools (Repository > Close All Tools) then reopening the folder or tool.

For example, a mapping includes a shortcut named Shortcut_To_FIL_InsertChanged. This shortcut points to a reusable transformation named FIL_InsertChanged. Another user edits the filter condition in the original transformation, FIL_InsertChanged, and saves changes to the repository. When you open the mapping in the Designer, it retrieves the mapping from the repository. It also retrieves information for Shortcut_To_FIL_InsertChanged (and any other shortcuts used in the mapping). To validate the mapping, click Mappings > Validate.

However, if the mapping is in memory, the Designer uses the version in memory. To ensure you have the correct version, click Edit > Revert to Saved. Designer displays the mapping with the latest shortcut properties. To validate the mapping, click Mappings > Validate.

Copying a Shortcut

You can copy a shortcut to other folders. When the Designer copies a shortcut, it creates another shortcut in the new folder. The new shortcut points to the original object used by the original shortcut.

The Designer cannot copy a shortcut when it cannot find the object that the shortcut references. This might occur if, for example, you copy a local shortcut from one repository to an unrelated repository.

When the Designer cannot successfully copy a shortcut, it creates a copy of the shortcut object. The copy of the shortcut object is identical to the original object the shortcut references. Unlike an actual shortcut, the copy will not inherit any changes to the original object. Use the copy of the shortcut as you would the original object. However, if the object is a source definition, you might need to rename the source definition.

The following table lists the results of copying global and local shortcuts to different repositories:

Shortcut Type	Shortcut Location	Copied to	Designer Creates...
Local shortcut	Standalone repository	Another folder, same repository.	Local shortcut to original object.
Local shortcut	Local repository	Another folder, same repository.	Local shortcut to original object.
Local shortcut	Local repository	Different local repository, same domain.	Copy of the shortcut object. To avoid losing metadata during the copy, the code pages of both repositories must be compatible.
Global shortcut	Local repository	Different local repository, same domain.	Global shortcut to original object. To avoid losing metadata during the copy, the code pages of both repositories must be compatible.
Global shortcut	Local repository	Different repository, different domain.	Copy of the shortcut object. To avoid losing metadata during the copy, the code pages of both repositories must be compatible.
Local shortcut	Global repository	Local repository, same domain	Global shortcut to original object.
Local shortcut	Global repository	Different repository, different domain.	Copy of the shortcut object. To avoid losing metadata during the copy, the code pages of both repositories must be compatible.

For example, if you copy a shortcut named Shortcut_to_Employees from one standalone repository to another, the Designer creates a new source definition in the destination folder named Shortcut_to_Employees. This source definition is a copy of the original shortcut, but is not a shortcut. When you use the source definition in a mapping, the default SQL used to extract data from the source defines the source as Shortcut_to_Employees. If the source table is named Employees in the source database, you must rename the source definition (Employees) or enter an SQL override for the source qualifier connected to the source definition (renaming the source table Employees) for the Integration Service to extract source data.

For example, the fourth row of the table indicates when you copy a global shortcut (a shortcut to an object in a global repository) from one local repository to another local repository in the same domain, the Designer creates a global shortcut to the object in the global repository.

Renaming Source Qualifiers to Shortcut Sources

By default, shortcuts are named after their original object, Shortcut_To_*ObjectName*. If you create a shortcut to a source and you have enabled the Mapping Designer option to create source qualifiers, the Mapping Designer creates a source qualifier based on the source name. If you do not change the name of the source shortcut, the resulting source qualifier is named SQ/ESQ/NRM_Shortcut_To_*SourceName*. Despite the name, however, the source qualifier is not a shortcut.

Tip: If names of source qualifiers created for shortcut sources cause confusion, you might want to rename those source qualifiers.

Tips for Working with Shortcuts

Keep shared objects in centralized folders.

This keeps maintenance simple. This also simplifies the process of copying folders into a production repository.

Create shortcuts to finalized objects.

Changes to an object referenced by shortcuts can invalidate the mappings or mapplets using the shortcut and any sessions using these objects. To avoid invalidating repository objects, create shortcuts objects in their finalized version.

After editing a referenced object, make sure affected mappings are still valid.

If you need to edit an object referenced by a shortcut, use the Analyze Dependencies feature in the Repository Manager to view affected mappings. After editing the object, see if the changes invalidate the listed mappings.

To ensure a mapping is valid, open and validate it in the Designer.

Refresh views of shortcuts when working in a multiuser environment.

To refresh a shortcut in the workspace, click Edit > Revert To Saved. You can also use Repository > Close All Tools in the destination folder then reopen the workspace.

Troubleshooting Shortcuts

The following message appears in the Designer status bar when I try to create a shortcut: “The selected folder is not open.”

You try to create a shortcut from a shared folder to a folder that is not open. Open the destination folder by opening at least one tool in the folder or by clicking Folder > Open before creating the shortcut.

When I try to create a shortcut, the Designer creates a copy instead.

This can occur when one of the following is true:

- The object is not saved in the repository. You can create shortcuts to objects that are in the repository. Save the object to the repository, and then try creating the shortcut again.
- You try to create a shortcut for an object in a non-shared folder. You can create shortcuts to objects in shared folders.
- You hold down the Ctrl key when dragging the object. To create a shortcut, drag the object without holding down any additional keys.
- You try to create a shortcut between two local repositories, or between two repositories that are not in the same domain. You can create a shortcut between repositories in the same domain. In addition, you can create a shortcut in a local repository, referencing an object in a global repository. You cannot create a shortcut in a global repository that references an object in the local repository.

- You drag an object from a shared folder in the global repository to a folder in the local repository, but you are connecting to the repositories separately. To create a global shortcut, you must connect to one repository and then connect to the second repository through the first repository.

CHAPTER 6

Team-Based Development with Versioned Objects

This chapter includes the following topics:

Team-Based Development with Versioned Objects Overview

If you have the team-based development option, you can configure the repository to store multiple versions of objects. You can configure a repository for versioning when you create it, or you can upgrade an existing repository to support versioned objects. With object versioning, you can store copies of previous versions of objects in development, track changes to those objects, and prepare them for deployment to a production environment.

A versioned repository assigns multiple version numbers to versions of the same object. Each time you check in an object, the repository increments the version number by one and stores a new version of the object in the repository database. A repository enabled for versioning can store multiple versions of the following objects:

- Sources
- Targets
- Transformations
- Mappings
- Mapplets
- Sessions
- Tasks
- Workflows

- Worklets
- User-defined functions
- Session configurations
- Schedulers
- Cubes
- Dimensions

You can complete the following tasks when you work with a versioned object:

- **View object version properties.** Each versioned object has a set of version properties and a status. You can also configure the status of a folder to freeze all objects it contains or make them active for editing.
- **Track changes to an object.** You can view a history that includes all versions of a given object, and compare any version of the object in the history to any other version. With the history, you can determine changes made to an object over time.
- **Check out or check in the versioned object.** You can check out an object to reserve it while you edit the object. When you check in an object, the repository saves a new version of the object, and you can add comments to the version. You can also find objects checked out by yourself and other users.
- **View multiple versions of an object in the workspace.** You can view multiple versions of an object in the workspace of the Designer and Workflow Manager.
- **Apply labels to objects**. You can create labels to associate with any versioned object or group of versioned objects in a repository. Use labels to track versioned objects during development, improve query results, and associate groups of objects for deployment or import and export.
- **Group objects for deployment**. You can create groups of versioned objects to deploy to another repository or folder. Use the result set from an object query to group objects for deployment. Or, you can create a static group of objects for deployment.
- **Delete or purge the object version.** You can delete an object from view and continue to store it in the repository. You can recover, or undelete, deleted objects. If you want to permanently remove an object version, you can purge it from the repository.

You can perform these tasks in the Repository Manager, Designer, and Workflow Manager.

Sample Scenario

You are working in an environment that includes a development repository and a production repository. You create and test metadata in the development repository and then deploy it to the production repository. While working in the development repository, you want to exclusively edit objects, retain older versions, and freeze the folder when you are finished with development to prevent further changes to the objects it contains.

Creating the Initial Version

You use the Designer to create a mapping, including source definition, target definition, and transformations. While you are working with the mapping, the repository locks the object for your use. Other users are unable to edit the objects you have checked out.

When the mapping is ready to test, you check it in to the repository. When you check in the mapping, the repository creates a new version of the object and assigns it a version number. The first time you check in the object, the repository assigns it version number one. You also include comments with the checked-in version, noting that it is the initial version of the mapping.

Creating Successive Versions

After creating a session and workflow and testing the initial version of the mapping, you edit the mapping based on the results of the tests. When you finish editing the mapping, you check it in to commit changes to a new version. Each time you check in the mapping, the repository creates a new version and increments the version number by one.

Applying Labels and Deploying Objects

After you finish editing the mapping, you want to move it to the production repository. To track the versioned object, you apply a label to it. You apply this label to all the objects that you are ready to deploy to the target repository. You create a query to identify all objects that use the deployment label, and associate this query with a dynamic deployment group. When you run the dynamic deployment group, the query groups objects that use the label. Later, you use the Copy Deployment Group Wizard to deploy this group of objects to the production repository.

Related Topics:

- "Labels" on page 86
- "Team-Based Development with Deployment Groups" on page 101

Deleting and Purging an Object

You want to remove a transformation instance from the mapping. When you delete the transformation, it no longer appears in the Navigator window, but the repository retains all versions of the transformation in the repository database.

You do not need to use this transformation any more. You purge all versions of the transformation, permanently removing them from the repository and freeing up space in the repository database.

Freezing a Folder

After you finish developing and testing the metadata, you decide to freeze the folder. Freezing the folder prevents other users from checking out objects. You decide to freeze the folder and allow deployment of objects in the folder.

Later, a query locates the object in the repository and includes it in a dynamic deployment group. The deployment group specifies objects in the development repository you want to put into production.

Related Topics:

- "Deleting and Recovering Objects" on page 79

Viewing Results View Windows

With a versioned repository in a team-based development environment, you can query the repository for information about versioned objects. You can view results for the following types of versioned objects:

- **Object dependencies.** View object dependencies to see the impact of changing or deleting an object.
- **Object queries.** You can search the repository for versioned objects that meet conditions you specify.
- **Checked-out objects.** You can view objects you and other users have checked out.
- **Object histories.** Object histories allow you to see the changes made to the object.

To switch between open results view windows, click Window > Results View List.

Customizing Results View Windows

The columns of a results view window show the properties of objects in the window. You can specify the columns that appear in the results view window, and you can change the order of the columns. You can also choose the display font for the elements of the results view window.

To customize a results view window:

1. Open the results view window that you want to customize.

   The Options window displays.
2. To add a column to the results view window, select the object property in the Available list and click the Move button. The object property moves to the Selected list.
3. To remove a column from the results view window, select the object property in the Selected list and click the Remove button. The object property moves to the Available list.
4. To change the order of the columns in the results view window, select an object property in the Selected list and click the up or down arrow.
5. To change the font of a results view window element, complete the following steps:
   - Select a category and click Change.
   - Select the font settings in the Font dialog box and click OK.
6. In the Options dialog box, click OK.

   Note: By default, the timestamp that appears in results view windows shows the last-saved time for an object. You can also display the checkout time and the checkin time by moving those properties from the Available list to the Selected list in the results view window display options.

Working with Version Properties

When you enable version control in a repository, you allow the repository to store multiple copies of the same object as you make changes and save it. The repository assigns each copy, or version, of the object a version number. Each time you check in an object, the repository creates a new version and increments the version number by one.

By default, the Navigator and workspace always display the latest version of an object. You can view the version history of an object or create a query to search for previous versions of an object. If you rename an object during development, different versions of the same object may have different names.

Every version of an object takes up approximately the same amount of space in the repository database. To conserve space, you can purge older object versions.

Viewing Version Properties

Use the Designer, Workflow Manager, or Repository Manager to view the version properties of an object. You can view the version properties, labels applied to the version, and the status of the object in the Properties dialog box.

To access the object properties, select the object in the Navigator and click View > Properties. Alternatively, right-click the object in the Main window or Navigator and click Properties. In a versioned repository, the Properties dialog box for a versioned object has the following tabs: Object, Version, Labels, and Object Status. In a non-versioned repository, the Properties dialog box displays the Object Status tab.

Object Properties

The Object tab of the Properties dialog box shows the object properties. Object properties are associated with the object when you create it. You can also view the current object version number and whether the object is checked out.

Version Properties

On the Version tab, you can view properties that are specific to the latest version of the object. This includes the version number, the user and host that created the version, and any comments associated with the version.

Labels Properties

On the Labels tab, you can view all labels applied to the object. For each label, you can also view the name of the user who applied the label, the time the label was applied, and comments associated with the label.

Object Status Properties

On the Object Status tab, you can view the current status of the object. In the Repository Manager, you can also change the object status on this tab.

The object status defines what actions you and other users can perform on the object. An object can have one of the following statuses:

- **Active.** You and other users can edit the object.
- **Deleted.** The object is marked as deleted and is not visible in the Navigator. You can find a deleted object through a query.

You can manually change the status of an object from the Repository Manager.

Changing Object Status

You can change the status of individual repository objects from the Repository Manager. Each version of an object maintains its own status. You must check in the object for other users to view changes to the object status. This is true even if you delete the object.

If you change the status of an object from Active to Deleted, the repository removes the object from view. No other objects can use or reference the deleted object. You can recover a deleted object by changing its status from Deleted to Active.

You can change the status of an object when you view object history, query results, object dependencies, deployment group contents, or checkouts. To change object status, complete the following steps.

To change the status of an object:

1. In the Repository Manager, right-click the object in the Navigator and click Versioning > View History.

   The View History window appears.
2. Select the latest version of the object, and click Tools > View Properties.

   The object properties appear.
3. In the Properties dialog box, click the Object Status tab.
4. From the Object Status list, select a status for the object.
5. Click OK.

6. Check in the object for other users to view changes to the status.

You can also change the status of folders in the repository.

Changing Folder Status

To configure operations that users can perform, you can change the folder status. To change the folder status, edit the folder in the Repository Manager. You can configure a folder with one of the following statuses:

- **Active.** This status allows users to check out versioned objects in the folder.
- **Frozen, Allow Deploy to Replace.** This status prevents users from checking out objects in the folder. Deployment into the folder creates new versions of the objects.
- **Frozen, Do Not Allow Deploy to Replace.** This status prevents users from checking out objects in the folder. You cannot deploy objects into this folder.

You might change the status of a folder to serve different purposes depending on the configuration of the environment and the development processes. For example, an environment uses a development repository for creating and testing metadata, and it uses a production repository for running the workflows and sessions. In the development repository, you might change the status of a folder from active to Frozen-Do Not Allow Deploy to Replace in a code freeze situation. This prevents other users from checking out objects in the folder and creating new versions. Both of the Frozen statuses allow the Repository Manager to add objects in the folder to a deployment group.

In the production repository, you might change the status of a folder from active to Frozen-Allow Deploy to Replace to ensure that copy deployment group operations successfully complete. The Frozen-Allow Deploy to Replace status prevents other users from checking out objects in the folder, but it allows the Copy Deployment Group operation to create new versions of objects. The Frozen-Do Not Allow Deploy to Replace status prevents the Copy Deployment Group operation from creating new versions of the objects.

Note: Before you change the status of a folder, you might want to verify that other users do not have objects in the folder checked out.

Tracking Changes to Versioned Objects

A repository enabled for version control maintains an audit trail of version history. It stores multiple versions of an object as you check it out, modify it, and check it in. As the number of versions, you may want to view the object version history. You may want to view version history for the following reasons:

- Determine versions that are obsolete and no longer necessary to store in the repository.
- Troubleshoot changes in functionality between different versions of metadata.

To accomplish tasks like these, you can view a history of all of the versions of an object stored in the repository. You can also compare two versions of an object displayed in the history.

Viewing Object History

The history of an object is a record of all of the versions of an object stored in the repository, going back to the initial version, version number one. You can view user changes to an object, the date and time of changes, and comments associated with and labels applied to each version. If you or another user purges a version from the repository, the object history retains a record of the version in the object history and specifies the user who purged the version.

You can view object history in the Designer, Repository Manager, and Workflow Manager.

To view object version history in the Repository Manager, right-click the object in the Main window or the Navigator and click Versioning > View History.

Use the following methods to view the object version history in the Designer or Workflow Manager:

- Right-click the object in the Navigator and click Versioning > View History.
- Right-click the object in the workspace and click Versioning > View History.
- Select the object in the workspace and click Versioning > View History from the menu.

When you click View History, the View History window displays the object version history.

The following table lists additional tasks you can perform from the View History window:

Task	Description
Compare versions.	Compare the selected object with the previous checked-in version.
View version properties.	View the object and version properties of the selected object.
Apply or remove a label.	Apply a label to a versioned object or a group of versioned objects.
Purge object version.	Purge individual versions of objects.
Perform an advanced purge.	Purge obsolete versions of objects based on purge criteria.
Add versioned object to deployment group.	Add an object or a group of objects to a deployment group.
View object dependencies.	View dependencies for the selected object.
Check in object or undo checkout.	Check in or undo the checkout for the selected object.
Save object version history to a file.	To save the object version history to an HTML file, click File > Save to File.
Export object version to an XML file.	Export the object version to an XML file.

Comparing Versions

When you view the version history of an object, you can compare two selected versions of the object. When you compare two versions of an object, the PowerCenter Client displays the attributes of each object.

To compare two versions of an object, select the versions that you want to compare in the object history and click Tools > Compare > Selected Versions. Alternatively, select one version and click Tools > Compare > Previous Version.

Checking Out and Checking In Objects

With a versioned repository in a team-based development environment, check out an object each time you change it, and save to commit the changes to the repository. You must check in the object to allow other users to make changes to it. Checking in an object adds a new version to the object history.

An object is in read-only mode until you or another user checks it out. When you view an object that is in read-only mode, it is available for other users to check in or check out. If another user checks out or checks in an object that you are viewing in read-only mode, a message appears in the Notifications tab of the Output window. If another user has an object checked out, you can open the object as read-only. To update the view of an object with the latest version of the object, select the object in the Navigator, and then click View > Refresh.

You can check out and check in objects in the Designer, Repository Manager, and Workflow Manager.

Checking Out Objects

To edit an object, you must check out the object. When you check out an object, the repository obtains a write-intent lock on the object. No other users can edit the object when you have it checked out. If you disconnect from the repository and do not save the object, it remains checked out to you, but you lose the changes you made to it.

Note: An object is checked out by default when you create, copy, replace, or import it.

To check out an object:

1. Select the object you want to check out.
2. Click Versioning > Check Out.
3. In the Check Out dialog box, enter an optional comment in the comment field.
4. Click OK to check out the object, or Apply to All to apply the checkout comment to multiple objects.

Viewing Checked-Out Objects

You can view objects you and other users have checked out. You might want to view checkouts to check in all of the objects you have checked out or to see if an object is available for you to check out.

You can narrow or broaden the search for checked-out objects in the following ways:

- **By folder.** Search for checkouts in the selected folder, or search all folders in the repository.
- **By user.** Search for objects you checked out, or search for objects checked out by all users in the repository.

To view checkouts:

1. In the Designer, Workflow Manager, or Repository Manager, click Versioning > Find Checkouts.
2. Optionally specify folder or user options to define the search, and click OK.

   The View Checkouts window appears. The results depend on the options you select for the search.

The following table lists additional tasks you can perform from the View Checkouts window:

Task	Description
Compare versions.	Compare the selected checkout with the previous checked-in version.
View version properties.	View the object and version properties of the checkout.
View object dependencies.	View dependencies for the selected checkout.
Check in object or undo checkout.	Check in or undo checkout for the selected unmodified checkouts.
Save object version history to a file.	To save the version history to an HTML file, click File > Save to File.

Task	Description
View object history.	View the object version history for the selected checkout.
Export object version to an XML file.	Export the version to an XML file.

Undoing a Checkout

When you undo a checkout, the repository releases the write-intent lock on the object and removes the checkout version from the repository. The most recently checked-in version of the object becomes the latest version of the object.

You can undo a checkout from the View History, View Checkouts, and Query Results windows.

To undo a checkout, select the checked-out object and click Versioning > Undo Checkout.

Checking In Objects

You must save an object before you can check it in. When you check in an object, the repository creates a new version of the object and assigns it a version number. The repository increments the version number when you check in an object. You must check in an object to purge it.

If you save an object without checking it in, the changes are committed to the repository, and the object remains checked out until you check it in. You can check in objects from the Designer, Workflow Manager, or Repository Manager.

You can also check in an object from the View History, View Checkouts, View Dependencies, and Query Results windows.

To check in an object:

1. Select the object or objects.
2. Click Versioning > Check in.
3. In the Check In dialog box, enter a comment in the comment field
4. Click OK to check in the object or Apply to All to apply the comment to multiple objects.

When you check in an object, the repository creates a new version of the object and increments the version number by one.

Checking Out and Checking In Composite Objects

Use the following rules and guidelines when you check out and check in composite objects:

- The Repository Service does not check out or check in reusable objects when you check out or check in a composite parent object, such as a mapping. For example, if you want to check out a mapping and all objects used in the mapping, you must check out all mapplets and reusable transformations individually.
- The Repository Service treats non-reusable objects as part of the parent object, so you cannot check out or check in individual non-reusable objects. For example, if you have a mapping that contains a non-reusable Aggregator transformation, you cannot check out the Aggregator transformation individually. When you check out the parent mapping, the Repository Service checks out the non-reusable Aggregator transformation.

- When you check out or check in cubes, the child objects (the fact tables) are also checked out or checked in. Likewise, when you check out or check in dimensions, the child objects (the levels) are checked out or checked in.
- You can check out or check in scheduler objects in the Workflow Manager or the Repository Manager:
  - In the Workflow Manager, run an object query. You can also check out a scheduler object in the Scheduler Browser window when you edit the object. However, you must run an object query to check in the object.
  - In the Repository Manager, run an object query. You can also select the Schedulers node in the Navigator and then check out objects from the Main window.
- You can check out or check in session configuration objects in the Workflow Manager:
  - In the Workflow Manager, run an object query. You can also click Tasks > Session Configuration and then check out objects from the Session Config Browser window.
  - In the Repository Manager, run an object query. Or, select the Configurations node in the Navigator and then check out objects from the Main window.

Deleting and Recovering Objects

When you delete an object in a versioned repository in a team based development environment, the repository removes the object from view in the Navigator and the workspace. The repository does not remove it from the repository database. Instead, the repository creates a new version of the object and changes the object status to Deleted. You can recover a deleted object by changing its status to Active.

Deleting a Versioned Object

You can delete a versioned object in the Designer or Workflow Manager. When you delete a versioned object, the repository changes the object status to Deleted and removes the object from view in the Navigator and workspace. After you delete an object, you must check it in for other users to view the changed status. Check in a deleted object in the Find Checkouts dialog box.

You can check out an object before you delete it to keep it as a current checkout object. You can also delete objects without first checking them out. In the Options dialog box, enable the option to delete objects without checkout. When you delete an object, the Repository Service checks out the object to you and then deletes it.

When you delete a composite object that contains non-reusable objects, the Repository Service treats the non-reusable objects as part of the parent object and deletes them. For example, when you delete a mapping, the Repository Service deletes all of the non-reusable child objects associated with the mapping.

The repository retains the metadata information for all versions of a deleted object. To permanently remove the object from the repository, you must purge it.

Recovering a Deleted Object

You can recover a deleted object by changing the object status to Active. This makes the object visible in the Navigator and workspace. Use a query to search for deleted objects.

You use the Repository Manager to recover deleted objects. Complete the following steps to recover a deleted object:

1. Create and run a query to search for deleted objects in the repository. You can search for all objects marked as deleted, or add conditions to narrow the search. Include the following condition when you query the repository for deleted objects:

   ```
   Version Status Is Equal To Deleted
   ```

2. Change the status of the object you want to recover from Deleted to Active.
3. If the recovered object has the same name as another object that you created after you deleted the recovered object, you must rename the object.

Purging Versions of Objects

You can purge specific versions of objects, or you can purge all versions of objects.

To permanently remove an object version from the repository, you must purge it. You need to check in object versions to purge them. You might want to purge a version if you no longer need it and you want to reduce the size of the repository database.

You can purge multiple versions of an object from the repository at the same time. To completely purge an object from the repository, you must purge all versions. If you purge a version that is not the latest version, the repository keeps a record of the purge in the object history. If you purge the latest version, the repository does not keep a record of the purge.

You can revert to an older version of an object by purging more recent versions. You cannot, however, promote an older version to the current version without purging the more recent versions. For example, you create 12 versions of a mapping. You then determine that you need to use version 10 of the mapping instead of version 12. You can purge versions 11 and 12 to make version 10 the current version.

You use the Repository Manager to purge versions. When you purge versions of objects, you can perform the following tasks:

- **Purge individual object versions.** You can select object versions in the View History window or Query Results window to purge the individual object versions.
- **Purge versions based on criteria.** You can purge versions at the repository, folder, or object level based on purge criteria. This type of purge is called an advanced purge. Use advanced purges to purge deleted or active objects. For deleted objects, you can specify the objects to purge based on the date of deletion. For active objects, you specify the versions to purge based on the version number, the check-in date, or both.
- **Preview purge results.** Preview an advanced purge to view the purge results before you purge objects from the repository. You can view summary or detailed information about the purge.
- **Purge composite objects.** You can purge versions of composite objects, and you can purge versions of dependent objects that make up composite objects. View object dependencies before you purge composite objects. You might get unexpected results if you do not determine the dependent object versions that a purge affects.

The following table shows the Repository Manager commands that you can use to purge versions at the object, folder, or repository level:

Purge Type	Single Object Version	Multiple Object Versions	Versions at Folder Level	Versions at Repository Level
By Object Version (View History Window)	yes	yes	no	no
By Object Version (Query Results Window)	yes	yes	no	no
Based on Criteria (Navigator)	yes	yes	yes	yes
Based on Criteria (View History Window)	yes	yes	no	no
Based on Criteria (Query Results window)	yes	yes	no	no

Purging Individual Object Versions

You can select individual versions of objects in the View History window or the Query Results window to purge the versions.

1. In the Navigator, Select an object and click Versioning > View History.

   Or, click Tools > Query, and run a query from the Query Browser.
2. In the results window, select the object versions to purge.
3. Click Tools > Purge Object Version.
4. In the confirmation message, click Yes.
5. Click OK.

   Warning: When you purge an object version, you might invalidate dependent objects.

Related Topics:

- "Purging Composite Objects" on page 83

Purging Versions Based on Criteria

In the Repository Manager, you can purge object versions based on criteria. This type of purge is called an advanced purge. You can purge object versions at the repository, folder, or object level.

When you purge versions based on criteria, you can perform the following tasks:

- **Purge versions of deleted objects.** Purge versions of checked-in deleted objects to permanently remove the versions from the repository. You can purge all checked-in deleted objects, or you can purge objects that were deleted before a specified date. When you purge deleted objects, you purge all versions of the objects.
- **Purge versions of active objects.** Purge specified checked-in versions of active objects. Active objects are undeleted objects and deleted objects that are not checked in. When you purge versions of active

objects, you specify the number of versions to keep, a purge cutoff time, or both. If you specify a number of versions to keep and a purge cutoff time, you purge versions that meet both conditions.

- **Preview versions before purging.** Before you purge versions based on criteria, you can preview the purge results to verify that the purge criteria produces the expected results.

Note: When you purge versions based on criteria, you cannot purge a dependent object version if it is used in an unpurged composite object.

The following table describes the options in the Advanced Purge window:

Option	Description
Purge Deleted Objects	Purges versions of checked-in deleted objects. Select All to purge versions of all deleted objects in a repository or folder, or select Older Than to purge versions of objects deleted before an end time. You can specify the end time either as the number of days before the current date or in MM/DD/YYYY HH24:MI:SS format.
Purge Active Objects	Purges specified versions of active objects. Select Older Than the Last *n* Versions to specify the number of latest checked-in versions to keep. For example, select 6 to purge all versions except the last six checked-in versions. If the object is checked out, you also retain the checked-out version. Select Older Than and specify a number of days or a date and time to purge versions that were checked in before a specified time.
Save Purge List	Output file to save information about purged object versions. Default is disabled.
Summary Only	Saves summary information in the purge output file and displays summary information in purge previews. Disable to view detailed information about each object version. Default is enabled.

The amount of time that the Repository Service takes to purge versions depends on the size of the repository, the number of deleted and old objects, and the composite objects that are affected. For optimal performance, purge at the folder level or use purge criteria to reduce the number of purged object versions. Avoid purging all deleted objects or all older versions at the repository level.

1. In the Navigator, select a repository to purge versions at the repository level.

   Or, select a folder to purge versions from the folder.

   You can also select one or more objects to purge objects based on criteria.

   Note: You can also use the View History window or the Query Results window to purge based on criteria. Select one or more objects in the window, and click Tools > Advanced Purge.

2. Click Versioning > Advanced Purge.

   Alternatively, right-click the repository or folder and select Advanced Purge, or right-click the selected objects and click Versioning > Advanced Purge.

3. To purge deleted objects, select Deleted Objects, and then specify whether to purge all deleted objects or objects deleted before an end date.

   Or, to purge active objects, select Active Objects, and then specify the versions to keep, the purge cutoff, or both. After you purge an object version, you cannot retrieve it. To ensure that you can revert to past versions, avoid purging all versions of an object.

4. Optionally, click Save Purge List to create an output file for the purge information.
5. Optionally, choose to view and save summary information instead of detailed purge information.
6. Optionally, click Preview to preview the purge.

7. Click Purge to purge the deleted objects.

   Tip: When you use an advanced purge to purge deleted objects, you purge all versions of the objects. To keep recent versions of deleted objects and purge older versions, define a query that returns the deleted objects. Then, use the *pmrep* PurgeVersion command with the -q option to retrieve the deleted objects and specify the versions to purge.

Previewing Purge Results

Before you purge versions based on criteria, you might want to preview the purge results. When you preview purge results, check the purge criteria before you purge versions from the repository. Also, review the affected object versions to verify that the Repository Service removes the obsolete versions and retains the versions that you want to keep.

When you preview a purge, you can view summary or detailed information about the purge.

To preview a purge, configure the purge criteria for an advanced purge. Choose to view and save summary or detailed information. Then, click Preview.

In the Preview window, you can click Purge to proceed with the purge, or you can click Cancel to close the Preview window without purging. Click Save To File to save the purge preview results to an output file.

Purging Composite Objects

When you purge versions based on criteria, the purged objects might include composite objects such as mappings or workflows. Before you purge a composite object, you need to consider object dependencies. Object dependencies can affect the way that dependent reusable objects are purged.

If you purge a composite object that consists of non-reusable dependent objects, you also purge the non-reusable dependent objects. If you purge a composite object that contains reusable dependent objects, you purge the dependent object versions if they are not used in another composite object.

You cannot purge a version of a dependent object if it is used in a version of a composite object that you do not purge. Also, if you cannot purge a particular version of an object, you cannot purge more recent versions of that object, even if the more recent versions are not used in composite objects.

This section provides two examples that show how dependencies can affect purges of active objects. The first example describes a frequently modified composite object with rarely updated dependent objects. The second example describes a composite object with few versions but frequently modified dependent objects.

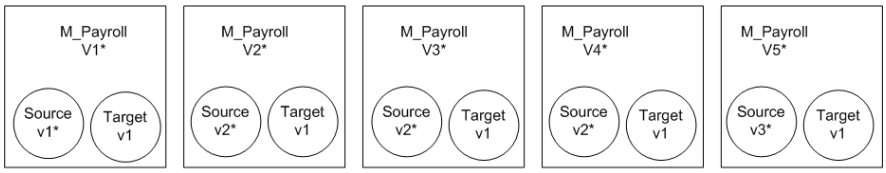
Tip: View dependencies before you purge an object to determine if a dependency might affect the versions that you purge.

Example of a Frequently Checked-Out Composite Object

You update the mapping m_Payroll often, and you frequently check it in and out. Five checked-in versions of the mapping exist. You rarely modify the source and target objects in the mapping. There are three checked-in versions of the source and one checked-in version of the target.

At the repository level, you purge versions based on criteria, and you indicate that you want to keep the last two checked-in versions of objects.

The following figure shows the history of versions 1 through 5 of the mapping:



* Indicates purged versions

The advanced purge produces the following results:

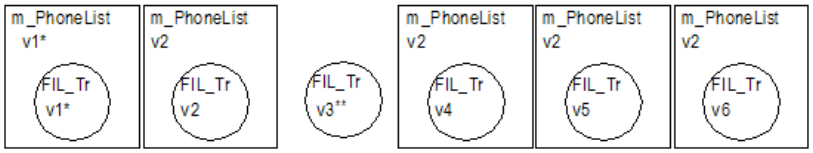
Object	Purged Versions
Mapping m_Payroll	Versions 1 through 3, assuming that no Session task or other composite object uses m_Payroll.
Source	Version 1. Because you purge the version of m_Payroll that uses source version 1, you also purge that version of the source. The purge keeps the last two checked-in versions of objects, so you do not purge versions 2 and 3 of the source.
Target	None. The purge keeps the last two checked-in versions of objects. Only one checked-in version of the target exists.

Example of a Rarely Checked-Out Composite Object

You rarely check in and check out the mapping m_PhoneList. Two checked-in versions of the mapping exist. However, you frequently check in and out the reusable transformation in the mapping. The transformation is a Filter transformation named FIL_Tr. It has six versions.

At the repository level, you purge versions based on criteria, and you specify that you want to keep only the latest checked-in version of objects.

The following figure shows the history of the mapping and transformation versions:



*Indicates purged versions.
**The mapping does not use version 3 of the transformation.

The advanced purge produces the following results:

Object	Purged Versions
Mapping m_PhoneList	Version 1, assuming that no Session task or other composite object uses m_PhoneList.
Transformation FIL_Tr	Version 1. You do not purge versions 2, 4, 5, and 6 of the transformation, because version 2 of m_PhoneList uses those transformation object versions. You do not purge version 3 of the transformation, because you retain version 2, which is an older version.

Note: If you cannot purge an older version of an object, the Repository Service retains all newer versions of the object during an advanced purge.

Rules and Guidelines for Purging Versions of Objects

Use the following rules and guidelines when you purge versions of objects:

- If you purge the latest version of an object and the preceding version has a different name, the preceding version takes the name of purged version. For example, you have the source src_Records. The latest version is named src_Records, but the name of the preceding version in the history is src_RecordsWeekly. If you purge the latest version, the name of the preceding version becomes src_Records.
- When you purge an individual version of a dependent object, you render composite objects invalid if they use the dependent object version. Verify object dependencies before purging individual object versions.
- In an advanced purge of an active object, you cannot purge a version of a dependent object if it is used in an unpurged version of a composite object.
- In an advanced purge of an active object, if you specify a number of versions to keep, you keep the latest checked-in version, even if it was checked in after the purge cutoff time. If the number of versions to keep is greater than the number of object versions, you keep all object versions.

CHAPTER 7

Labels

This chapter includes the following topics:

Labels Overview

A label is a global object that you can associate with any versioned object or group of versioned objects in a repository. You may want to apply labels to versioned objects to achieve the following results:

- Track versioned objects during development.
- Improve query results.
- Associate groups of objects for deployment.
- Associate groups of objects for import and export.

For example, you might apply a label to sources, targets, mappings, and sessions associated with a workflow so that you can deploy the workflow to another repository without breaking any dependency.

You can apply the label to multiple versions of an object. Or you can specify that you can apply the label to one version of the object.

You can create and modify labels in the Label Browser. From the Repository Manager, click Versioning > Labels to browse for a label.

Creating and Editing Labels

When you create or edit a label, you can specify the label name and add comments.

You can also lock the label, which prevents other users from editing or applying the label. You lock a label to limit access to the label or ensure you apply a label once. For example, you might want to apply a label to a group of objects to indicate that you tested the objects and are ready to deploy them. After you apply the label, you can lock it to prevent users from editing the label or applying the label to other objects.

Creating a Label

To create a label, click Versioning > Labels from the Repository Manager to open the Label Browser.

Note: Click a column heading to sort labels by that column.

Click New to open the Label Editor. Select from the options in the Label Editor to create a label object.

Editing a Label

When you edit a label object, you can edit the name or the comments associated with it. You can lock the label to prevent other users from editing or applying the label. When you delete a label, the Repository Service permanently deletes all instances of the label that you applied to versioned objects.

To edit a label, click Edit in the Label Editor, and select the options to change.

Applying Labels

You can apply one or more labels to any versioned object in the repository. You can select any label you have execute permissions for. You can also apply the label to selected dependent objects. For example, if you want to group dependencies for a workflow, you can label all child objects. The Repository Service applies labels to sources, targets, mappings, and tasks associated with the workflow.

If you deploy objects to multiple repositories, you can apply the label to global shortcut dependencies. When you deploy labels to global shortcut dependencies, you can apply the label to dependent objects in a global repository from a local repository. You can also apply the label to dependent objects in all registered local repositories in a global repository.

You can apply labels to objects when you complete one of the following tasks from the Designer, Workflow Manager, or Repository Manager:

- **View the history of an object**. When you view the history of an object, click Tools > Labels > Apply Label from the View History window.
- **Create an object query**. When you run an object query, click Tools > Labels > Apply Label from the Query Results window.

Alternatively, you can apply labels by selecting Versioning > Apply Labels in the Repository Manager. In the Repository Manager, you open the Label Wizard. You can apply labels to groups of objects in the Label Wizard.

The following table describes the label options:

Label Options	Description
Move Label	Moves the label from a previous version of the object to the latest version of the object. If the Repository Service detects the label is applied to another version of the same object, you can move the label to the selected version of the object.
Primary/Foreign Key Dependencies	Applies the label to the source object containing the primary key referenced by the foreign key in the selected source object.
Global Shortcut Dependencies	Applies the label to global shortcut objects. Select one of the previous options such as Label All Children. Select Global Shortcut Dependencies. The Repository Service applies the label to global shortcuts that meet the conditions you specify.

Label Options	Description
Label all Children	Applies the label to all repository objects that the selected object uses.
Label all Parents	Applies the label to all repository objects that use the selected object.
Preview	Previews the group of objects that the Repository Service applies the label to when you apply labels to dependent objects.

When you apply labels to objects, you can apply a label to one version of an object at a time. For example, you apply the Deployment label to version one of a mapping. When you create version two of this mapping, you can move the Deployment label to version two of the mapping, or you can apply a different label. You cannot apply the Deployment label to both versions of the mapping.

When you label parent objects, such as mappings, workflows, and worklets, you must apply the label to nonreusable child objects. If you do not apply labels to nonreusable child objects, the labels for parent and nonreusable child objects may not synchronize.

When you search for an object, view an object history, or view object properties, you can view metadata for the labels applied to an object. You can view the label owner, the time stamp when you applied the label, and the comments you entered when you applied the label to the object.

Note: The Repository Service applies the label to objects that are checked in to the repository. You cannot apply labels to checked-out objects.

Applying Labels to Groups of Objects

In the Repository Manager, use the Apply Label Wizard to apply a label to groups of objects. To open the Apply Label Wizard, click Versioning > Apply Labels and click Next. You can apply labels to the following groups of objects:

- One or more objects in a folder.
- All objects in one or more folders.
- All objects in one or more selected repositories.

After you select objects to label, browse to select the label to apply and choose label options.

Click Preview to view the objects the Repository Service labels.

CHAPTER 8

Object Queries

This chapter includes the following topics:

Object Queries Overview

An object query is a global object that you use to search for repository objects that meet specified conditions. When you run a query, the repository returns results based on those conditions. You can run an object query to locate versioned and non-versioned objects. You can run an object query from the Designer, Workflow Manager, or Repository Manager.

You can create an object query to complete the following tasks:

- **Find and maintain object relationships.** Use object queries to locate parent and child dependencies, shortcut dependencies, and primary and foreign key relationships.
- **Find groups of invalidated objects to validate**. Use a query to locate invalidated objects.
- **Associate a query with a deployment group**. When you create a dynamic deployment group, you can associate an object query with it.
- **Track versioned objects during development**. You can add Label, User, Last saved, or Comments parameters to queries to track versioned objects during development.
- **Find deleted versioned objects to recover**.

To create and run an object query, you configure the query conditions and run the query. Each query condition consists of a parameter, an operator, and a value. You can add, cut, copy, paste, and move query conditions. When you run the query, the Repository Service queries the repository and displays the results in the Query Results window.

Note: You can create queries in the Repository Manager or through the pmrep CreateQuery command.

Related Topics:

- "Configuring Query Conditions" on page 90
- "Running a Query" on page 96

Using the Query Browser

You can create, edit, run, or delete object queries in the Query Browser. You can view the list of available queries in the Query Browser. Click Tools > Queries to browse for a query.

You can also configure permissions for the query from the Query Browser.

Note: Click a column heading to sort queries by that column.

In the Query Browser, click New or Edit to open the Query Editor. You can create, validate, and run a query in the Query Editor. To save the query with another name, click Query > Save As.

When you create a query, you can use the And/Or button to add logical operators. You can make a query personal or shared. You can run any personal object query you own and any shared object query in the repository.

Configuring Query Conditions

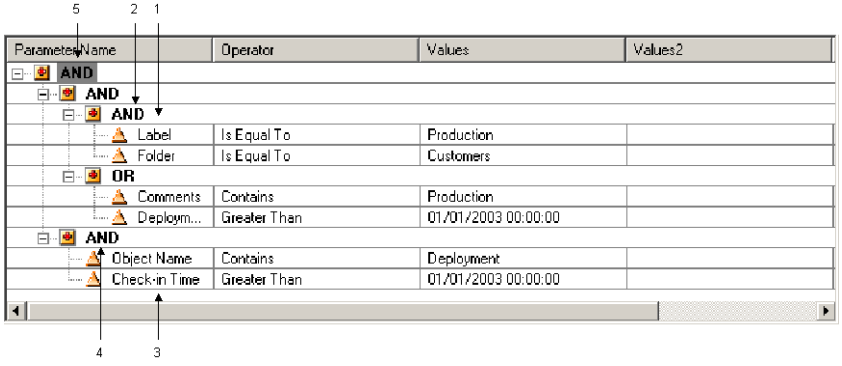
Each query condition consists of a parameter, an operator, and a value. You can add, cut, copy, paste, and move query conditions. Each time you add a query parameter, you specify an operator and a value. You can view the valid operators and values for each parameter when you add the parameter to a query.

You may need to configure multiple query conditions to effectively narrow the results. Use the AND and OR logical operators to add multiple query conditions. For example, you might want to deploy a group of objects with the Deployment_2 label, but you also want to ensure that the objects were saved after a particular date.

When the Repository Service processes a parameter with multiple conditions, it processes them in the order you enter them. To receive expected results and improve performance, enter parameters in the order you want them to run.

If you nest several parameters within a query, the Repository Service resolves each inner parameter conditions before outer parameter conditions.

For example, when you run the following query with nested conditions, the Repository Service resolves the innermost conditions and the next outer conditions until it resolves all parameter conditions. The query shows the order in which the Repository Service resolves query conditions:



Query Parameters

You build queries using query parameters. Each query parameter uses operators and accepts values. Some query parameters are available for versioned objects only. These are query parameters that relate to

configuration management. For example, the Check-In Time query parameter displays checked-in versioned objects for a specified time, before or after a specified time, or within a specified number of days.

The following table describes the query parameters and the valid operators and values for each parameter:

Parameter	Description	Valid Operator	Accepted Values
AND	Joins query conditions or groups of query conditions.	None	None
Business Names	Displays sources and targets based on their business names. For example, the query Business Name is Equal to Informatica, returns sources and targets that contain the Informatica business name and filters out all other objects.	Contains, Does Not Contain, Does Not End With, Does Not Start With, Ends With, Is Equal To, Is Not Equal To, Is Not One Of, Is One Of, Starts With	String
Check-in Time	Displays checked in versioned objects for a specified time, before or after a specified time, or within a specified number of days. You can specify this parameter for versioned repositories only.	Greater Than, Is Between, Less Than, Is Not Between, Within Last (days)	Date/time, Numeric
Check-out Time	Displays checked out versioned objects for a specified time, before or after a specified time, or within a specified number of days. You can specify this parameter for versioned repositories only.	Greater Than, Is Between, Less Than, Is Not Between, Within Last (days)	Date/time, Numeric
Comments	Displays comments associated with a source, target, mapping, or workflow.	Contains, Does Not Contain, Does Not End With, Does Not Start With, Ends With, Is Equal To, Is Not Equal To, Is Not One Of, Is One Of, Starts With	String
Deployment Dispatch History	Displays versioned objects deployed to another folder or repository through deployment groups in a given time period.	Greater Than, Is Between, Less Than, Is Not Between, Within Last (days)	Date/time, Numeric

Parameter	Description	Valid Operator	Accepted Values
Deployment Receive History	Displays versioned objects deployed from another folder or repository using deployment groups in a given time period.	Greater Than, Is Between, Less Than, Is Not Between, Within Last (days)	Date/time, Numeric
Folder	Displays objects in a specified folder.	Is Equal To, Is Not Equal To, Is Not One Of, Is One Of	Folder name
Include Children	Displays child dependent objects.	Where (Value 1) depends on (Value 2)	Source Definition, Target Definition, Transformation, Mapplet, Mapping, Cube, Dimension, Task, Session, Worklet, Workflow, Scheduler, SessionConfig
Include Children and Parents	Displays child and parent dependent objects.	Where (Value 1) depends on (Value 2)	Source Definition, Target Definition, Transformation, Mapplet, Mapping, Cube, Dimension, Task, Session, Worklet, Workflow, Scheduler, SessionConfig

Parameter	Description	Valid Operator	Accepted Values
Include Parents	Displays parent dependent objects.	Where (Value 1) depends on (Value 2)	Source Definition, Target Definition, Transformation, Mapplet, Mapping, Cube, Dimension, Task, Session, Worklet, Workflow, Scheduler, SessionConfig
Include Primary/ Foreign Key Dependencies	Displays primary key-foreign key dependencies.	-	-
Impacted Status	Displays objects based on impacted status. Objects can be marked as impacted when a child object changes in such a way that the parent object may not be able to run.	Is Equal To	Impacted, Not Impacted
Label	Displays versioned objects associated with a label or group of labels. You can specify this parameter for versioned repositories only.	Contains, Does Not Contain, Does Not End With, Does Not Start With, Ends With, Is Equal To, Is Not Equal To, Is Not One Of, Is One Of, Starts With	String
Last Saved Time	Displays objects saved at a particular time or within a particular time range.	Greater Than, Is Between, Less Than, Is Not Between, Within Last (days)	Date/time, Numeric
Latest Status	Displays versioned objects based on the object history. The query can return local objects that are checked out, the latest version of checked-in objects, or a collection of all older versions of objects. You can specify this parameter for versioned repositories only.	Is Equal To, Is Not Equal To, Is One Of	Checked-out Latest, Checked-in Older

Parameter	Description	Valid Operator	Accepted Values
Metadata Extension	Displays objects based on an extension name or value pair. Use this query parameter to find non-reusable metadata extensions. The query does not return user-defined reusable metadata extensions.	Is Equal To, Is Not Equal To	Vendor-defined metadata domain
Object Name	Displays objects based on the object name.	Contains, Does Not Contain, Does Not End With, Does Not Start With, Ends With, Is Equal To, Is Not Equal To, Is Not One Of, Is One Of, Starts With	String
Object Type	Displays objects based on the object type. For example, you can find all workflows in a specified folder.	Is Equal To, Is Not Equal To, Is Not One Of, Is One Of	Cube, Dimension, Mapping, Mapplet, Scheduler, Session, Session Config, Source Definition, Target Definition, Task, Transformation, User-Defined Function, Workflow, Worklet
Object Used Status	Displays objects that are used by other objects. For example, you can find mappings that are not used in any session. If any version of an object is used by another object, the query returns the most recent version of the object. This occurs even when the most recent version of the object is unused. The query does not return workflows or cubes because these objects cannot be used by other objects.	Is Equal To	Unused, Used

Parameter	Description	Valid Operator	Accepted Values
Shortcut Status	Displays objects based on shortcut status. If you select this option, the query returns local and global shortcut objects. Shortcut objects are considered valid regardless of whether the objects they reference are valid.	Is Equal To	Is Not Shortcut, Is Shortcut
Reusable Status	Displays reusable or non-reusable objects.	Is Equal To, Is One of	Non-reusable, Reusable
User	Displays objects checked in or checked out by the specified user.	Is Equal To, Is Not Equal To, Is Not One Of, Is One Of	Users in specified repository
Valid Status	Displays valid or invalid objects. The Repository Service validates an object when you run validation or save an object to the repository.	Is Equal To	Invalid, Valid
Version Status	Displays objects based on deleted or non-deleted status. You can specify this parameter for versioned repositories only.	Is Equal To, Is One of	Deleted, Not deleted

Validating and Saving a Query

After you create the object query and specify the conditions, you validate it.

Click Validate to run the query validation. The Validate Query window displays the validation results. If the validation returns an error, review the error message and validate the query again.

After you validate the query, you can save it for later use. For example, if you regularly run an object query on checkin histories, you might want to save the object query so that you can easily run it again.

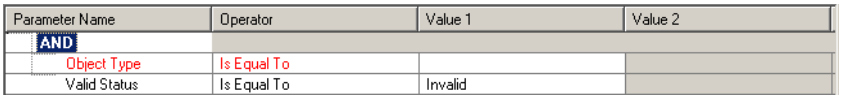
Invalid Queries

The query parameters you can view and use in the Query Editor are determined by the tool you use to create queries. When you create a query in one PowerCenter Client tool, the query may appear invalid when you open it in another tool. For example, you can view query parameters such as workflows, worklets, and sessions in the Workflow Designer. If you open a query that uses Workflow Designer parameters in the Designer, the query may appear invalid.

For example, you create a query in the Workflow Manager using the following parameters:

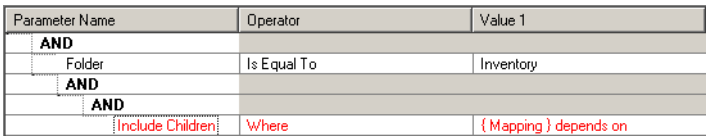
- Object Type is equal to Workflow
- Valid Status is equal to Invalid

The following figure shows the invalid query when you open the query in the Designer:



Queries can be invalidated when you use logical operators with the wrong number or kind of query conditions. For example, an AND logical operator requires at least one parameter to be valid.

The following figure shows a sample query that is invalid because the AND operator has no parameters:



Running a Query

After you create and validate the object query, you can run it. The Repository Service queries the repository and displays the results of the query in the Query Results window.

From the Query Results window, you can complete tasks by selecting the object and clicking Tools.

The following table lists additional tasks you can perform from the Query Results window:

Task	Task Information
View history.	View the object version history for the selected checkout.
Compare versions.	Compare the selected object with the previous checked in version.
Validate multiple objects.	Validate selected objects.
Check in.	Check in an object version.
Undo check out.	Undo an object checkout.
Export to an XML file.	Export the object version to an XML file.
Apply or remove a label.	Apply a label to a versioned object or a group of versioned objects.
View object dependencies.	View dependencies for the selected object.
View version properties.	View the object and version properties of the selected object.
Add version to deployment group.	Add an object or a group of objects to a deployment group.
Change object status.	Change the status of an object from deleted to active.
Purge object version.	Purge individual versions of objects.
Perform an advanced purge.	Purge obsolete versions of objects based on purge criteria.
Save object version history to a file.	To save the version history to an HTML file, click File > Save to File.
Open object in a workspace.	Select this option to open an object in the workspace when the object type is compatible with the tool in which you run the query. For example, you can open a workflow in the Workflow Manager using this option.

Viewing Query Results

By default, when you run an object query in a non-versioned repository, the Repository Service returns reusable objects.

When you run an object query in a versioned repository, the Repository Service returns objects that meet the following criteria:

- Versioned objects are the latest version, either local checked-out versions or checked-in versions.
- Non-versioned objects that are saved to the repository.
- Objects are reusable.
- Objects that are not deleted.

If you have a versioned repository and you want to view deleted or older objects, you must specify these values in the query parameter. You can search for deleted objects using the deleted value in the Version Status parameter. You can search for older versions of objects when you specify the older value in the Latest Status query parameter.

Note: If you run a query without specifying any query condition, the query returns reusable objects in the repository.

Sample Queries

The following sample queries show how to create query conditions with different parameters, operators, and values. When you run the following sample queries in a versioned repository, the query returns the latest checked-in version of the objects that meet the query criteria. When you run the following queries in a non-versioned repository, the query returns the latest saved objects that meet the query criteria.

Finding Object Dependencies

To search for the parent and the child dependencies, use the Include Children and Parents parameter. In this example, select source definitions and target definitions for Value 1 and select mapping for Value 2. To include reusable and nonreusable dependencies, select both reusable and nonreusable dependency in Value 3.

The following figure shows the query that returns sources and targets in the Vendors folder that are dependent on the mapping, Mapping_deploy:

Parameter Name	Operator	Value 1	Value 2	Value 3
AND				
Folder	Is Equal To	Vendors		
Object Name	Contains	Mapping_deploy		
Include Children and Parents	Where	{ Source Definition,Target Definiti...	Mapping	Reusable Dependency,Non-reusable D...

Finding Impacted Mappings

The Repository Service marks a parent object as impacted if you modify a child object in such a way that the parent object may not be able to run. The query returns impacted composite objects such as mapplets, mappings, sessions, worklets, and workflows.

The following figure displays the query returns objects that are impacted *and* are mappings:

Parameter Name	Operator	Value 1	Value 2
AND			
Impacted Status	Is Equal To	Impacted	
Object Type	Is Equal To	Mapping	

Note: Use the impacted query to search for impacted objects and run a validation on multiple objects.

Related Topics:

- "Validating Multiple Objects" on page 44

Finding Invalid Mappings

The Repository Service invalidates objects when you save an object or run validation and it detects changes to the object that cause problems with data flow.

The following query returns objects that are invalid *and* are mappings:

Parameter Name	Operator	Value 1	Value 2
AND			
Object Type	Is Equal To	Mapping	
Valid Status	Is Equal To	Invalid	

Finding the Used Status of Objects

The following query returns objects that are mappings *and* that are not used by any session:

Parameter Name	Operator	Value 1	Value 2
AND			
Object Type	Is Equal To	Mapping	
Object Used Status	Is Equal To	Unused	

Finding Recently Deployed Versioned Objects

The following query returns the latest checked-in version of versioned objects deployed within the last seven days *and* are reusable or non-reusable:

Parameter Name	Operator	Value 1	Value 2
AND			
Deployment Receive...	Within Last (Days)	7	
Reusable Status	Is One Of	Non-reusable,Reusable	

Finding Recently Checked-Out Objects

The following query returns currently checked-out objects that were checked out within the last seven days:

Parameter Name	Operator	Value 1	Value 2
AND			
Latest Status	Is Equal To	Checked-out	
Check-out Time	Within Last (Days)	7	
AND			

Finding Older Versions of Versioned Objects

The following query returns versioned objects that are mappings *and* that are older *and* that are checked in by the Administrator:

Parameter Name	Operator	Value 1	Value 2
AND			
Object Type	Is Equal To	Mapping	
Latest Status	Is Equal To	Older	
User	Is Equal To	Administrator	

Finding Versioned Objects Older than a Specified Date

Use this query to search for older versions of objects to purge. This query ensures that you do not purge the latest versions of objects.

The following query returns object versions that were checked in before a specified date *and* that are not the latest checked-in version:

Parameter Name	Operator	Value 1	Value 2
AND			
Check-in Time	Less Than	8/3/2004 1:10:25 PM	
Latest Status	Is Not Equal To	Latest Checked-in	

Troubleshooting Object Queries

I created a query to return objects from a specified folder. The query returned reusable objects. Why?

By default, when you run an object query, the query returns reusable objects that are visible to the current user.

To find both reusable and nonreusable objects in a specified folder, include the Reusable Status parameter and specify reusable and nonreusable values:

Parameter Name	Operator	Value 1	Value 2
AND			
Folder	Is Equal To	Vendors	
Reusable Status	Is One Of	Non-reusable,Reusable	

I created a query to return objects that use a specified label. The query returned reusable objects that use the specified label. Why?

By default, when you run a query to find objects associated with a label, the query returns labeled reusable objects. To find reusable and nonreusable objects that use a specified label, include the Reusable Status parameter and specify reusable and nonreusable values:

Parameter Name	Operator	Value 1	Value 2
AND			
Folder	Is Equal To	Vendors	
Label	Is Equal To	Ready_to_Deploy	
Reusable Status	Is One Of	Non-reusable,Reusable	

I created a query to search for labeled versioned objects. The query did not return older versions of labeled objects. Why?

By default, when you run a query to find labeled versioned objects, the query returns the latest checked-in version of objects. To find all versions of objects using the specified label, include the Latest Status parameter and specify latest checked-in and older values:

Parameter Name	Operator	Value 1	Value 2
AND			
Folder	Is Equal To	Vendors	
Label	Is Equal To	Ready_to_Deploy	
Reusable Status	Is One Of	Non-reusable,Reusable	
Latest Status	Is One Of	Latest Checked-in,Older	

Do I need to edit the query condition after I change the name of a folder or label?

No. After you change the name of a folder or label, the Repository Service retrieves and uses the folder name or the label name in the query condition. The query returns the same results after you rename a folder or label.

CHAPTER 9

Team-Based Development with Deployment Groups

This chapter includes the following topics:

Team-Based Development with Deployment Groups Overview

If you have team-based development, you can create deployment groups. A deployment group is a global object that consists of objects from one or more folders. You use a deployment group to copy objects to another folder or repository. You can use a deployment group when you want to copy some, but not all, of the objects in a folder. You can also use a deployment group to copy objects from multiple folders.

You can create, edit, or delete deployment groups. You can copy a deployment group and the objects in the deployment group to a target repository.

Deployment Group Tasks

You can complete the following tasks when you work with deployment groups:

- **Create a deployment group**. Create a global object for deploying objects from one or more folders.
- **Edit a deployment group.** Modify a deployment group. For example, you can convert a static deployment group to a dynamic group, or you can convert a dynamic deployment group to a static group.
- **Configure privileges and permissions for a deployment group**. Configure permissions on a deployment group and the privilege to copy a deployment group.
- **View the objects in a static or dynamic deployment group**. Preview the objects that the Repository Service will deploy.
- **Add or remove objects in a static deployment group.** Specify the objects that belong to a static deployment group.

- **Associate a query with a dynamic deployment group.** Assign a query to a deployment to dynamically update the objects that the group contains.
- **View the history of a deployment group**. View the history of a deployment group, including the source and target repositories, deployment date, and user who ran the deployment.
- **Post-deployment validation.** Validate the objects in the target repository after you copy a deployment group to verify that the objects and dependent objects are valid.
- **Roll back a deployment group.** Roll back a deployment group to purge deployed versions of objects from the target repository.

Configuring Privileges and Permissions for a Deployment Group

Configure object permissions when you create, edit, delete, or copy a deployment group. To limit the privilege to perform deployment group operations but provide the privilege to copy a deployment group without write permission on target folders, assign the Execute Deployment Groups privilege. An administrator can assign the Execute Deployment Groups privilege. You must have read permission on source folders and execute permission on the deployment group to copy the deployment group.

Related Topics:

- "Managing Permissions" on page 56

Adding or Removing Objects in Static Deployment Groups

You manually add or delete objects from a static deployment group. You can add checked-in objects to a static deployment group from the Repository Manager. You cannot add checked-out objects to a deployment group. You can add objects to a deployment group when you view the results of an object query or view the results of an object history query from the Repository Manager. To add objects from the Query Results or View History window, click Tools > Add to deployment group.

In the Repository Manager, right-click an object in the Navigator or in a detail window, and click Versioning > View History. In the View History window, click Tools > Add to deployment group.

To add several objects to a deployment group, select the objects in the Navigator and drag them into the deployment group. When you select a static deployment group in the Navigator, the Main window displays the objects within the deployment group.

When you add objects to a static deployment group, you can also add dependent objects to the deployment group. You can specify the following conditions to add dependencies:

- **All dependencies**. Select to deploy all dependent objects.
- **Non-reusable**. Select to deploy non-reusable dependent objects.
- **No dependencies**. Select to skip deploying dependent objects.

When you click All Dependencies, you add all dependent objects to the static deployment group. Dependent objects include dependent objects within a workflow or mapping, original objects that shortcuts reference, and primary key sources where there is a primary-key/foreign-key relationship.

To have the Repository Manager use the recommended setting without prompting you, select the option to prevent the dialog box from appearing again. Alternatively, click Tools > Options, and clear Prompt User While Adding to Deployment Group.

Using Queries in Dynamic Deployment Groups

When you associate an object query with a deployment group, the Repository Service runs the query at the time of deployment. You can associate an object query with a deployment group when you edit or create a deployment group.

To deploy composite objects using a dynamic deployment group, you must deploy all components of the composite object the first time you deploy the deployment group to another repository. For example, if you deploy a mapping, you must also deploy the reusable and non-reusable child objects associated with the mapping. To do this, you must create a query that returns parent objects and their dependent child objects. A common way to group versioned objects for deployment is to use labels to identify the objects you want to deploy.

To find the latest versions of objects in a dynamic deployment group, you must create all mappings in the group with labels. If the dynamic deployment group contains a non-reusable object in an unlabeled mapping, the group will not deploy.

When you use labels to identify versioned objects for a dynamic deployment group, the labels for parent and dependent child objects can become out of sync. When this occurs, queries may return part of a composite object, and the dynamic deployment fails. This can occur in the following cases:

- **You apply a label to a parent object, but do not label the dependent child objects**. When you apply a label to a parent object, the label does not apply to child objects. For example, you apply label 1 to mapping 1 without labeling the dependent child objects. Later, you run a dynamic deployment group using a query that searches for objects in a specified folder that use label 1. The query returns the parent object but not the child objects. The deployment fails because you attempted to deploy only the parent for a composite object. To ensure that dynamic deployment queries return these child objects, manually apply the label to dependent objects each time you apply a label or move a label to a different version of the parent object.
- **You do not apply a specified label to the same version of the parent and child object**. By default, object queries return the latest versions of objects. For example, you apply label 1 to version 1 of a child object and apply label 1 to version 2 of the parent object. In the query, you search for objects that use label 1 and reusable and non-reusable objects. The query returns the parent object but not the child objects because the most recent versions of the child objects do not have the label applied. To ensure that dynamic deployment queries return both parent and child objects when you apply a specified label to different versions of parent and child objects, include a Latest Status parameter in the query and specify the latest checked-in and older values.
- **The dynamic deployment query does not return non-reusable child objects with parent objects.** To ensure that the dynamic query returns reusable and non-reusable child objects, include the Reusable Status parameter in the query and specify reusable and non-reusable values. In addition, include a Latest Status parameter in the query and specify the latest checked-in and older values.

Viewing Deployment History

You can view the following information about groups you have deployed:

- **Date/time**. The date and time you deployed the group.
- **User name**. The user name of the person who deployed the group.
- **Deployment group name**. The name of the deployment group.
- **Source repository**. The repository you deployed the group from.
- **Target repository**. The repository where you deployed the group.
- **Status.** The status of the group as either deployed or not deployed.
- **Rollback time.** The date and time the deployment group was rolled back.

To view the history of a deployment group:

1. Click Tools > Deployment > Groups to open the Deployment Group Browser.
2. Select a deployment group.
3. Click View History to view the history of the deployment group.
4. Optionally, click Details to view details about the objects in the deployment group.
5. Click OK to close the Deployment Group History window.

Validating the Target Repository

Validate the objects in the target repository after you copy a deployment group to verify that the objects or dependent objects are valid. You can also use the *pmrep* Validate command or the Repository Manager to validate the objects.

You can view the validation results in the deployment log. In the Repository Manager, the deployment log appears in the Output window.

Note: Validating objects in the target repository can take a long time.

Rolling Back a Deployment

You can roll back a deployment to purge the deployed versions from the target repository or folder. When you roll back a deployment, you roll back all the objects in a deployment group that you deployed at a specific date and time. You cannot roll back part of a deployment or roll back from a non-versioned repository.

To initiate a rollback, you must roll back the latest version of each object. The Repository Service ensures that the check-in time for the repository objects is the same as the deploy time. If the check-in time is different, then the repository object is not the same as the object in the deployment, and the rollback fails. The rollback also fails if the rollback process causes you to create duplicate object names. This might occur if you rename a deployed object, create a new object with the same name, and attempt to roll back the original deployment.

To roll back a deployment:

1. In the Repository Manager, connect to the target repository where you deployed the objects.
2. Click Tools > Deployment > History.
3. Select a deployment group in the Deployment Group History Browser, and click View History.
4. Select a deployment to roll back.
5. Click Rollback.

The Repository Service checks the object versions in the deployment against the objects in the target repository or folder, and the rollback either succeeds or fails. The rollback results appear at the end of processing. If the rollback fails, the Repository Service notifies you of the object that caused the failure.

Creating and Editing Deployment Groups

You can create the following types of deployment groups:

- **Static.** You populate a static deployment group by manually selecting objects. Create a static deployment group if you do not expect the set of deployment objects to change. For example, you might group objects for deployment on a certain date and deploy all objects at once.

- **Dynamic**. You use the result set from an object query to populate the deployment group. Create a dynamic deployment group if you expect the set of deployment objects to change frequently. For example, you can use a dynamic deployment group if you develop multiple objects to deploy on different schedules. You can run the dynamic deployment group query multiple times and add new objects to the group each time you run the query.

You can edit a deployment group to convert it into another deployment group type. You can view the objects in the deployment group before you copy a deployment group.

Creating a Deployment Group

You use the Deployment Group Editor to create and edit deployment groups.

To create a deployment group:

1. In the Repository Manager, click Tools > Deployment > Groups to view the existing deployment groups in the Deployment Group Browser.
2. Click New to configure the deployment group in the Deployment Group Editor.
3. Enter a name for the deployment group.
4. Select whether to create a static or dynamic deployment group.
5. If you are creating a dynamic deployment group, click Queries to select a query from the Query Browser, and then click Close to return to the Deployment Group Editor.
6. Optionally, enter a comment for the deployment group.
7. Click OK.

After you create a deployment group, it appears in the Deployment Groups node in the Navigator of the Repository Manager.

After you create a static deployment group, you can add objects to it.

Related Topics:

- "Adding or Removing Objects in Static Deployment Groups" on page 102
- "Using Queries in Dynamic Deployment Groups" on page 103

Editing a Deployment Group

You edit a deployment group to convert a static deployment group into a dynamic deployment group, to convert a dynamic deployment group into a static group, or to associate a different query with a dynamic deployment group.

To edit a deployment group:

1. In the Repository Manager, click Tools > Deployment > Groups.
2. In the Deployment Group Browser, select the deployment group, and click Edit.
3. In the Deployment Group Editor, configure the static or dynamic deployment group.
4. Click OK.

Viewing the Objects in a Deployment Group

Before you deploy a static or dynamic deployment group, you can preview the objects that will be deployed.

To view the objects in a deployment group:

1. In the Repository Manager, click Tools > Deployment > Groups.
2. In the Deployment Group Browser, select the deployment group, and click View Group.

   For a static deployment group, the deployment group objects appear in the Deployment Group Contents window. For a dynamic deployment group, the deployment group objects appear in the Query Results window.

Related Topics:

- "Running a Query" on page 96

CHAPTER 10

Copying Folders and Deployment Groups

This chapter includes the following topics:

Copying Folders and Deployment Groups Overview

Use the Repository Manager to copy multiple objects from one repository to another. You can complete the following copy operations:

- **Copy a folder.** You can copy a folder and all of its contents within a repository or from a source repository into a target repository.
- **Replace a folder.** You can copy a folder and all of its contents from a source repository and replace an existing folder in the target repository. The contents of the replaced folder are overwritten.
- **Copy a deployment group.** You can copy the objects in a dynamic or static deployment group to multiple target folders in the target repository. For versioned repositories, the deployment operation creates new versions of existing objects in the target folders. For non-versioned repositories, if the objects in the deployment group exist in the target repository, the deployment operation deletes existing objects and creates new objects.

If you want to archive or share metadata or deploy metadata into production, you can use copy folder to copy an entire folder. If you want to update the metadata in a folder in production, you can replace the folder.

For example, you have a folder called Sales in the development repository. When it is ready for production, you copy the Sales folder into the production repository. After a week in production, you want to make minor changes. You edit the Sales folder in the development repository and test the changes. When the folder is ready for production, you can either copy the folder into the production repository, resulting in two copies of the folder in production, or you can replace the existing Sales folder with the new one. When you replace the folder, you can update the production repository without creating multiple copies of the folder.

If the repository is enabled for versioning, you can also copy the objects in a deployment group from one repository to another. When you copy a deployment group, you can copy objects in a single copy operation

from multiple folders in the source repository into multiple folders in the target repository. You can also specify individual objects to copy, rather than the entire contents of a folder.

You can also use different copy operations together. You might use copy folder and copy deployment group together in the repository environment at different stages of the development process.

For example, you have development and production repositories. When you complete initial development for the metadata in a new folder and you are ready to deploy the objects into production, you copy the folder and all of its objects into the production repository.

As development continues, you make changes to a session in the folder. You do not need to copy all of the metadata in the folder to production, so you add the session to a deployment group. When you copy the deployment group, the Copy Deployment Group Wizard creates a new version of the session in the target folder.

Copying or Replacing Running Workflows, Sessions, and Tasks

When you copy or replace a folder or deploy a deployment group, the Repository Service first copies the folder or deployment group to temporary tables in the target repository database. During this stage of the deployment, you have read-only access to the target repository if you are copying but not replacing a folder, or if you are copying a deployment group. Workflows, sessions, and tasks that are running in the target repository continue to run. You can view them running in the Workflow Monitor after the deployment operation begins.

Note: If you are *replacing* a folder, you cannot view the folder in the target repository after the deployment operation begins. Also, all running workflows, sessions, and tasks are immediately blocked in the target folder, and they remain blocked for the duration of the deployment operation. Running workflows, sessions, and tasks in other folders in the target repository continue to run.

After the Repository Service copies all objects in the folder or deployment group to temporary tables, it moves the data from the temporary tables to the target repository. During this stage of the deployment, you no longer have read-only access to the target repository, and all running workflows, sessions, and tasks are blocked. When an Integration Service tries to access the repository to run a workflow, session, or task while a repository is blocked, the Repository Service denies access and returns the following message:

```
Access to the repository is blocked since a folder/object deployment is in progress. The
current operation will be suspended until the deployment is completed.
```

The error message appears in the Administrator tool, workflow log, or session log, depending on which operation the Integration Service tried to perform. For example, if the Integration Service tried to fetch session information to run a session in a workflow, the message appears in the workflow log.

The Integration Service pauses until the repository completes the deployment. It cannot fetch objects in that repository during this time.

When the repository allows access again, it displays the following message:

```
The folder/object deployment has completed. The current operation will resume.
```

The Integration Service fetches the repository object and completes the workflow, session, or task.

Using the Copy Wizards

The Repository Manager provides a wizard to copy and replace folders and to copy deployment groups. The wizard steps vary depending on the operation and the content of the folder or deployment group you want to copy or the target repository type.

When you copy a folder or deployment group, you perform many of the same actions. You can use the Copy Folder Wizard and the Copy Deployment Group Wizard to complete the following actions:

- **Choose an Integration Service.** Use the Integration Service to run all workflows in the folder if a matching Integration Service does not exist in the target repository.
- **Retain assigned Integration Services.** Retain the assigned Integration Services for workflows configured to run on specific Integration Services.
- **Copy connections.** Copy database, FTP, external loader, and application connection information if matching connection names do not exist in the target repository.
- **Copy plug-in application information.** Copy plug-in application component information that does not exist in the target repository.
- **Copy persistent values.** Copy the saved persistent values for mapping variables used in a session and workflow variables used in a workflow.
- **Copy metadata extension values.** Copy the metadata extension values associated with repository objects.
- **Assign an owner to a folder.** Assign an owner to a folder when you copy a folder.
- **Validate the objects in the target repository.** Validate the objects in the target repository after you copy a deployment group to verify that the objects and dependent objects are valid.
- **Generate a deployment control file.** Generate a deployment control file, encoded in UTF-8 format, that you use with the *pmrep* command line program.

Related Topics:

- "Copying or Replacing a Folder" on page 111
- "Copying a Deployment Group" on page 117

Copy Modes

When you copy a folder or deployment group, you must choose from the following copy modes:

- **Typical.** The wizard uses the defaults for shortcuts to local and global shared folders.
- **Advanced.** You can override the defaults for shortcuts to local and global shared folders. You can choose the shared folder to associate shortcuts. The wizard might have to determine how the folders are related before establishing shortcuts.

Associated Integration Services

Each workflow is configured to be run by an Integration Service associated with the repository. A copied workflow becomes associated with an Integration Service in the target repository in the following circumstances:

- If the target repository is associated with Integration Service names that match the Integration Services configured to run the folder workflows, the wizard associates those workflows with the existing Integration Services. To use Integration Services with the same name in the target repository, you must configure those Integration Services before you copy the folder.
- If an Integration Service with the same name does not exist in the target repository, the wizard lists all of the Integration Services associated with the target repository. You then select one of those Integration Services to run all unassociated workflows.
- If the target repository is associated with one Integration Service, the wizard associates all unassociated workflows with it.
- If the target repository is not associated with an Integration Service, the wizard does not copy Integration Service connection information.

If you associate a different Integration Service with a workflow when you copy a folder, make sure that it uses the same directory structure for service and process variables and other directory paths in the session properties.

Connections

The Copy Wizard copies connections used by sessions in the folder or deployment group. If the connection exists in the target repository, the Copy Wizard uses the existing connection. The wizard does not overwrite connections in the target repository.

When you copy a folder or deployment group, the wizard displays the following information about connections:

- **No Match Found.** No match exists in the target repository. The wizard copies the object. You have access to the object in the source repository.
- **Match Found.** A matching object with the same name exists in the target repository. You have access to the objects in both the originating and target repositories.
- **Match Found - Permission Denied, will copy and rename to [new_name].** A matching object with the same name exists in the target repository. You have access to the object in the originating repository, but no access to the object in the target repository. The wizard copies the object and renames it by appending a number to the name.
- **Permissions Denied.** You have no access to the object in the source repository. All sessions using this connection are invalidated.

Metadata Extensions

When you copy objects to another repository, the Copy Wizard copies the metadata extension values associated with those objects to the target repository. The metadata extension values might or might not be available in the target repository, depending on whether the extensions are non-reusable or reusable.

Non-Reusable Metadata Extensions

Non-reusable metadata extensions apply to single objects such as one source definition or one session. You create non-reusable metadata extensions in the Designer or Workflow Manager.

When you copy an object that contains a non-reusable metadata extension, the Copy Wizard copies the extension to the target repository with the object. The extension becomes non-reusable in the target repository. You can edit it, delete it, or promote it to a reusable extension. If the metadata extension contains a value, the wizard retains the value of the metadata extension in the target repository.

Related Topics:

- "Working with Metadata Extensions" on page 164

Reusable Metadata Extensions

Reusable metadata extensions apply to all repository objects of a certain type, such as all workflows or all Expression transformations. There are two types of reusable metadata extensions that you can copy:

- **User-defined extensions.** Reusable metadata extensions that you create exist in the User Defined Metadata domain. When you copy an object that contains user-defined, reusable metadata extensions, the Copy Wizard copies the extensions to the target repository. If the definition exists in the target repository, the copied extensions become reusable in the target repository. If the definition does not exist in the target repository, the copied extensions become non-reusable.

- **Vendor-defined extensions.** Reusable extensions that other applications create exist in the appropriate vendor domain. When you copy an object that contains vendor-defined metadata extensions, the Copy Wizard copies the extensions to the target repository and retains their values. If the vendor domain exists in the target repository, the metadata extensions become part of that domain in the target repository. Therefore, you can view them or change the values as you do with the other metadata extensions in that domain.

  If the vendor domain does not exist in the target repository, you can cancel the copy or continue in the Copy Wizard. If you continue, the extensions are not available in the target repository. When you install the vendor application, the metadata extensions become available so you can view them and change their values.

Related Topics:

- "Working with Metadata Extensions" on page 164

Copying Plug-in Application Information

When you copy a folder or deployment group, you can copy plug-in application information if the folder or deployment group depends on the plug-in application information. The source folder or deployment group depends on a plug-in application in the following cases:

- The source folder contains metadata extension values from a vendor-defined metadata domain.
- The source folder contains a source or target with a plug-in application database type.
- The source folder uses plug-in application connections.

Copying or Replacing a Folder

In the Repository Manager, you can copy a folder within the same repository. You can also copy a folder into a different repository within the same PowerCenter domain or to a different PowerCenter domain. Use the Copy Folder Wizard to perform the copy or replace operation. Each time you copy or replace a folder, the wizard copies all of the metadata objects in the folder.

You can also copy and replace a folder across repositories. You might replace a folder when you want to update a folder that is in production. Instead of creating a second copy of the folder in the production repository, you can replace the existing folder with the updated data. When you replace a folder, the wizard may overwrite data in the target folder, depending on the options you select. To ensure no metadata is lost, back up the repository before replacing a folder.

If the Integration Service uses operating system profiles, the Copy Folder Wizard retains operating system profile assignments of the target folder. The Copy Folder Wizard does not copy the operating system profiles assignment when you copy a folder.

In the Copy Folder Wizard, you can perform all of the tasks listed in "Using the Copy Wizards" on page 108. When you copy a folder, you can complete the following actions in the Copy Folder Wizard:

- **Reestablish shortcuts.** Maintain shortcuts to objects in shared folders.
- **Compare folders.** Compare folders to determine how they are related.
- **Rename folders.** If a folder in the target repository has the same name as the folder you are copying, you can rename the copy of the source folder that the Copy Folder Wizard creates in the target repository.
- **Copy latest object versions or all object versions.** You can choose to copy the latest version of objects in the source folder, or all versions.

When you replace a folder, the wizard maintains properties of the replaced folder, such as shortcuts, FTP, and external loader connection information. When you replace a folder, you can complete the following additional actions:

- **Retain current values for Sequence Generator and Normalizer transformations and XML generated keys.** You can choose to retain existing values or replace them with values from the source folder. XML generated keys include primary and foreign keys in XML transformations.
- **Retain persistent values for mapping variables.** You can choose to retain existing values or replace them with values from the source folder.
- **Retain persistent values for workflow variables.** You can choose to retain existing values or replace them with values from the source folder.
- **Workflow logs.** You can choose to retain existing workflow logs or replace them with workflow logs from the source folder.
- **Copy latest object versions or all object versions.** If you copy the latest object versions in the source folder, the wizard replaces each object in the target folder with the latest version. The wizard does not retain any older versions in the target folder. If you copy all versions from the source folder, the wizard removes all existing versions of the object from the target folder, including the latest one, and replaces them with all versions from the source folder.

The wizard copies and replaces folders as a single transaction. If you cancel the copy before it completes, the wizard rolls back all changes.

Naming

When you copy a folder, the wizard names the copy after the folder. If the folder name exists in the repository and you choose not to replace it, the wizard appends the date to the folder name, as follows: <*folder_name*>_<mm/dd/yyyy>, where mm=months, dd=days, and yyyy=year.

Locking and Checkouts

To protect the integrity of the repository, the wizard does not allow you to copy a folder when the folder, or objects in the folder, are being saved. Likewise, you cannot save objects in a folder as the wizard copies the folder. Before you copy a folder, view object locks to verify that the folder is not in use.

If you are replacing a folder in a target repository enabled for versioning, you must also verify that no objects in the target folder are checked out.

Related Topics:

- "Working with Version Properties" on page 73
- "Viewing Checked-Out Objects" on page 77

Shortcuts

The folder you want to copy might contain shortcuts to shared folders in the same repository or to shared folders in the global repository. Shortcuts to folders in the same repository are known as local shortcuts.

Shortcuts to the global repository are called global shortcuts. When you copy multiple versions of folders, you must take steps to ensure that you preserve shortcuts.

Related Topics:

- "Reestablishing Shortcuts When Copying Multiple Folder Versions" on page 113

Local Shortcuts

The wizard can reestablish local shortcuts to shared folders if you have a current copy of the shared folders in the target repository. Therefore, you can keep shortcuts intact by copying the necessary local shared folders to the target repository before copying the folder.

If you do not copy the shared folders before copying the folder, the wizard deletes all shortcuts and marks all affected mappings invalid.

If shared folders exist in the target repository, the wizard verifies that the copy is current. In typical mode, if you edit the original shared folder after you copy it to the target repository, the wizard asks you to copy it again. In the advanced mode, you can compare the folders to see which folder is most recent. The wizard does not establish shortcuts to an outdated shared folder. Therefore, to copy shortcuts correctly, you must copy shared folders before copying the folder.

If you copy the folder from the global repository to a local repository in the same domain, local shortcuts become global shortcuts.

For example, you copy a shared folder and a non-shared folder with shortcuts from a global repository to a local repository. First copy the shared folder into the local repository. Then copy the non-shared folder. If you copy the folder in typical mode, you establish global shortcuts to the shared folder in the global repository. If you copy the folder in advanced mode, you can also choose to establish local shortcuts to the shared folder in the local repository.

Global Shortcuts

If you copy the folder or deployment group to another repository in the same repository domain, the wizard can reestablish global shortcuts in the following situations:

- When you copy a folder from one local repository to another within the repository domain, the wizard recreates global shortcuts to the global repository.
- When you copy a folder from a local repository to its global repository, the global shortcuts become local shortcuts.
- When you copy a folder from a global repository to a local repository within the repository domain, local shortcuts become global shortcuts.

If you copy folders that contain global shortcuts between repository domains, copy the shared folders from the local and global repositories in the source domain to either the local or global repository in the target domain. The Copy Folder Wizard will either reestablish global shortcuts as local shortcuts or establish the copied shortcuts as global shortcuts.

Reestablishing Shortcuts When Copying Multiple Folder Versions

When you copy folders containing shortcuts to another repository, the Copy Folder Wizard reestablishes shortcuts to the referenced folder in the target repository. If you copy two versions of the referenced folder to another repository, the wizard reestablishes shortcuts to the folder most recently copied to the target repository by default. For example, you have folder F1 and folder F2 in a development repository. Folder F2 contains a shortcut to folder F1. You copy F1 to the production repository as F1_1. Later, you copy F1 to the production repository as F1_2. When you copy F2 to the production repository, the Copy Wizard reestablishes shortcuts to F1_2. If you modify the shortcut object in F1_1, the shortcut in F2 does not reflect the changes and may be invalidated.

To maintain valid shortcuts, you must verify that you maintain the most recent versions of shortcut objects in the most recently copied folder. Or, select Advanced Options when you copy folders to another repository. Use Advanced Options to select the folder to which you reestablish shortcuts.

Folder Permissions and Owners

When you copy or replace a folder, the wizard copies all permissions for the source folder owner to the target folder. The wizard does not copy permissions for users, groups, or all others in the repository to the target folder. When you replace a folder, the wizard retains the target folder permissions for users, groups, and all others in the repository.

By default when you copy or replace a folder, the wizard assigns the source folder owner to the target folder. The wizard does not assign the source folder owner to the target folder in the following situations:

- You choose to retain the target folder owner in the Copy Wizard.
- You specify a different owner for the target folder in the Copy Wizard. You can select any user in the target PowerCenter domain.
- You copy or replace the folder to a repository in a different PowerCenter domain, and the user name and security domain of the source owner do not exist in the target PowerCenter domain. The wizard assigns the user performing the copy as the target folder owner.

Copying Within a Repository

When you copy a folder within the same repository, the wizard asks you to rename the folder. The wizard reestablishes all shortcuts, and the copied folder continues to use the same connection and service information.

Copying Folders Between Versioned and Non-Versioned Repositories

You can copy folders between versioned and non-versioned repositories. When you copy a folder from a versioned repository to a non-versioned repository, the Copy Wizard copies the latest checked-in version of the objects to the target repository. If you copy a folder from a non-versioned repository to versioned repository, the Copy Wizard copies the objects as version 1. If you copy deleted objects or deleted shortcuts from a versioned repository to another versioned repository, the Copy Wizard copies a deleted version of the objects to the target repository. Later, you can recover the deleted objects. If you want to copy a deleted object from a versioned repository to a non-versioned repository, you must first recover it in the versioned repository.

Copying from Local Repositories

When you copy a folder from a local repository to another repository, the wizard verifies that a folder of the same name exists in the target repository. If it does not exist, the wizard uses the folder name for the copied folder. If it does exist, the wizard asks you to rename the folder.

If you want to copy the folder again, you might want to rename the existing folder in the target repository, using a naming convention that clearly defines the existing folder. If you have edited objects in any local shared folders used by the folder, you must copy those shared folders into the target repository before copying the folder. You might want to rename existing shared folders before performing the copy.

Steps to Copy or Replace a Folder

Before you copy a folder, use the Repository Manager to verify that no users are accessing objects in the folder. You might want to ask all users to exit the repository. Also, copy dependent shared folders to the target repository before copying a folder. If you are replacing a folder, verify that no users are accessing objects in the target repository.

1. In the Repository Manager, connect to the source repository and select the folder you want to copy.
2. Click Edit > Copy.
3. If you are copying to a different repository, connect to the target repository.

   You connect to the target repository with the same user account used to connect to the source repository. To connect to the target repository with a different user account, use the DeployFolder *pmrep* command.
4. In the Navigator, select the target repository, and click Edit > Paste.

   Tip: You can also drag the folder into the target repository after connecting to the repository.

   The Copy Folder Wizard displays the folder name and target repository name.
5. The Copy Folder dialog box prompts you to select a mode:
   - **Typical.** The wizard uses the defaults for shortcuts to local and global shared folders.
   - **Advanced.** You can override the defaults for shortcuts to local and global shared folders. You can choose the shared folder to associate shortcuts. The wizard might have to determine how the folders are related before establishing shortcuts.
6. Click Next.

   The Copy Folder Wizard prompts you for more information based on the content of the folders and the copy mode you select.

   The Next button is disabled if object locks cannot be acquired in the target repository. When the objects in the target repository become available, the Next button is enabled. To stop the replacement, click Cancel. The wizard rolls back all changes.

The following table lists the dialog boxes and prompts that may appear when you copy a folder:

Copy Folder Wizard Dialog Box	Modes	Description
Select Versions	Typical, Advanced	Copies the latest version or all versions of objects in the folder. If you copy the latest object versions in the source folder, the wizard replaces each object in the target folder with the latest version.
Replace Folder	Typical, Advanced	Lists existing copies of the folder or all folders in the repository.
Source/Target Comparison	Typical, Advanced	Specifies if objects in the target folder have been created or modified since the last copy.
Compare Folders - Compare Results	Typical, Advanced	Compares modified folders to determine the similarities and differences using a one-way comparison. You cannot compare the mapping variable values.
Sequence Generators and Normalizers	Typical, Advanced	Retains current values for Sequence Generator and Normalizer transformations.
Mapping Variables	Typical, Advanced	Copies persistent values for mapping variables if they exist.

Copy Folder Wizard Dialog Box	Modes	Description
Retain Mapping Variable Persisted Values	Typical, Advanced	Retains persistent values for mapping variables if you replace a folder.
Dependency Information	Typical, Advanced	Copies dependency information for objects in mappings if it exists. The dependency information exists if you set the general options for the Designer to save MX data. The dependency information is organized in a format that enables reporting tools to skip steps such as fetching the entire mapping and parsing expressions when collecting information for reporting.
Workflow Variables	Typical, Advanced	Copies persistent values for workflow variables.
Retain Workflow Variable Persisted Values	Typical, Advanced	Retains persistent values.
Copy Workflow Run History	Typical, Advanced	Copies workflow logs if they exist.
Retain Workflow Run History	Typical, Advanced	Retains existing workflow logs in the target folder if you choose not to copy workflow logs from the source folder.
Folder Exists	Typical, Advanced	Renames the target folder if a folder of the same name exists in the target repository. Otherwise, it appends the date to the original folder name.
Retain Integration Service Assignment	Typical, Advanced	Retains the assigned Integration Service for workflows.
Database Connections	Typical, Advanced	Lists all database connections in the folder, indicating the connections for which you do not have permission to copy.
Message Queue Connections	Typical, Advanced	Lists all message queue connections in the folder.
FTP Connections	Typical, Advanced	Lists all FTP connections in the folder, indicating the connections for which you do not have permission to copy.
External Loaders	Typical, Advanced	Lists all external loader connections in the folder, indicating the connections for which you do not have permission to copy.
Application Connections	Typical, Advanced	Lists all application connections in the folder, indicating the connections for which you do not have permission to copy.
MMD Plug-in	Typical, Advanced	Lists plug-in applications that the source folder depends on if the plug-in information does not exist in the target repository.
Integration Services	Typical, Advanced	Selects an Integration Service to associate with workflows. If the target repository contains less than two Integration Services, the wizard skips this step.
Local Shared Folders	Typical	Lists local shared folders in the target repository that you need to copy first. To preserve shortcuts and prevent invalid mappings, click Cancel and copy all listed local shared folders before copying the folder.

Copy Folder Wizard Dialog Box	Modes	Description
Global Shared Folders	Typical	Lists global shared folders in the target repository that you need to copy first. To preserve shortcuts and prevent invalid mappings, click Cancel and copy all listed local shared folders before copying the folder.
Outdated Shared Folders	Typical	Lists outdated folders in the target repository that you need to copy first. To preserve shortcuts and prevent invalid mappings, click Cancel and copy all listed local shared folders before copying the folder.
Select Shared Folders	Advanced	Lists the folder that contains shortcuts and the folder to which you can establish shortcuts. You can choose to accept or override the shared folder.
Override Shared Folder	Advanced	Lists folders you can select to establish shortcuts if you choose to override the default folder selection in the Select Shared Folders dialog box.
Compare Folder	Advanced	Compares folders if the folders you choose in the Override Shared Folder dialog box are different. You can compare the folders using a one-way comparison. To compare the folder objects using two-way comparison, use the Compare Folders Wizard before you start the Copy Folder Wizard.
Compare Shared Folder - Compare Results	Advanced	Lists the results from the folder comparison, displays objects that exist in the local shared folder but not in the global shared folder, and displays objects that are older or newer than matching objects in the global shared folder. When you compare the folders using a one-way comparison, you can check the folder objects that excludes the mapping variable values. If there are differences between the folders, a message warns you that shortcuts to missing objects will be removed. The wizard takes you back to Select Shared Folders dialog box.
Owner	Typical, Advanced	Selects the owner for the copied folder in the target repository. You can select any user in the target PowerCenter domain.
Complete Deployment	Typical, Advanced	Copies the folder immediately after completing the wizard or generates a deployment control file to use with the *pmrep* command line program. If you do not specify an absolute path, the Repository Manager generates the deployment control file in the directory where the Repository Manager runs. The Repository Manager generates the deployment control file encoded in UTF-8 format.

Copying a Deployment Group

Copy a deployment group and the deployment group objects to folders in a target repository. When you copy a deployment group, you can choose to replace an existing deployment group or create another deployment group. Use the Copy Deployment Group Wizard to copy objects in a deployment group into multiple folders in the target repository.

At the time of deployment, the wizard copies all objects included in a static deployment group. If you are copying a dynamic deployment group, the wizard runs the query associated with the deployment group and copies the objects from the results of the query. When you copy a dynamic deployment group, the Repository Service converts it to a static deployment group in the target repository.

You can copy parts of composite objects, local and global shortcuts, objects with different or conflicting names or status in a deployment group to folders in a target repository.

Note: Verify that the deployment group contains objects before you copy a deployment group. You cannot copy an empty deployment group.

Copying to Repository Types

You can copy a deployment group between versioned an non-versioned repositories. When you copy a deployment group from a versioned repository to a non-versioned repository, the Copy Deployment Group wizard replaces the objects in the target repository with the objects in the deployment group. When you copy a deployment group from a non-versioned repository to a versioned repository, the wizard creates new versions of the objects in the target repository.

If an object copied to a non-versioned repository exists in the target repository, the wizard deletes the object before copying the object from the deployment group. You cannot roll back a deployment from a non-versioned repository.

The first time you copy an object to a versioned repository, the wizard creates an object in the target repository. The next time you copy the object, the wizard identifies the previously copied object and replaces it, creating a new version of the object in the target repository. After it creates the version, the wizard checks in the object.

Related Topics:

- "Copying Object Types" on page 118

Copying Object Types

Consider the relationships between objects in the deployment group and objects in the target repository when you copy the following types of objects:

- **Parts of composite objects.** When you create a deployment group, you can choose to copy all or part of composite objects. If you choose to deploy part of a composite object, you must ensure that dependent objects exist in the target folder.
- **Local and global shortcuts.** When you copy a deployment group, you can reestablish local shortcuts to objects in shared folders. The wizard does not allow you to reestablish global shortcuts. As a result, you must ensure that the shared folders and global shortcuts exist in the target repository.
- **Objects with different or conflicting names in the deployment group and target repository.** An object in the target repository can be a copy of the object in the deployment group but have a different name. In this situation, the wizard replaces the copy of the object with the object in the deployment group.

  An object in the target repository may also have the same name as an object in the deployment group, but may not be a copy of the deployment group object. If this naming conflict occurs, the wizard cannot copy the deployment group object.
- **Objects with different statuses in the deployment group and target repository.** The status of an object in a deployment group may change after the copy operation, depending on the status of the object before deployment.

Locking and Checkouts

To protect the integrity of repository metadata, the Copy Deployment Group Wizard does not allow you to copy a deployment group when objects targeted for replacement are checked out or locked. Before you copy a deployment group, search for checkouts in the target repository and verify that no deployment target objects are checked out.

You can freeze the target deployment folder to ensure that no target objects are checked out when you copy a deployment group. When you freeze a folder, other users cannot check out objects in the folder, but the wizard can still copy and check in deployment group objects. Change the folder status to Frozen, Allow Deploy.

Note: If the repository stops unexpectedly during the copy operation, the Repository Service rolls back changes. However, the deployment group objects may be copied to the target repository but not checked in. If this happens, the objects will be stored in the repository as checked-out objects. To complete the copy operation, view checkouts and manually check in the objects.

Related Topics:

- "Working with Version Properties" on page 73
- "Checking Out and Checking In Objects" on page 76

Copying Composite Objects

A composite object is one that uses other objects. For example, a mapping may use a reusable source, reusable target, and several non-reusable transformations. Each of these objects is a child dependency of the mapping. You can copy the following composite objects to a deployment group:

- Local shortcuts
- Mappings
- Mapplets
- Sessions
- Worklets
- Workflows

When you create a deployment group, you can choose to include all dependencies, non-reusable dependencies, or no dependencies for composite objects. If you choose to copy no dependencies or non-reusable dependencies for a composite object, the wizard uses existing copies of objects in the target repository for all child dependencies not included in the deployment group. If the wizard cannot locate necessary dependencies in the target repository, it fails the copy operation.

You must ensure that the dependent objects are also included in the deployment group or exist in the target repository. The first time you deploy a group, you must include all dependencies of the composite object. To ensure that necessary dependencies exist in the target repository, you might want to copy the entire folder to the target repository the first time you copy the objects. You can then use deployment groups to update individual objects over time. After you initially deploy a group, you do not need to add all object dependencies to the deployment group.

For example, you edit a mapping variable in a mapping. You want to update the copy of the mapping currently stored in the production repository. You add the mapping to a deployment group with no dependencies because you do not want to update any non-reusable or reusable transformations in the mapping. When you copy the mapping to the production repository, the wizard replaces the current version of the mapping and associates all existing transformations with the new version.

When you deploy composite objects, the Repository Service treats the non-reusable objects in the composite object as part of the parent object. For example, if the parent object is deleted, the Repository Service treats associated non-reusable objects as deleted.

You can also add dependencies to the deployment group. Use one of the following methods to ensure that you include dependencies in the deployment group:

- **Manually add the dependencies to the static deployment group.** The PowerCenter Client prompts you to do this when you manually add an object to a static deployment group. You may want to add all dependencies the first time you copy an object to another repository.
- **Design the query associated with the dynamic deployment group to find dependencies.** You can design the query to search for dependencies of a composite object. You may want to further refine the query for a dynamic deployment group by specifying other parameters.

Copying Shortcuts

The deployment group you want to copy might contain shortcuts to shared folders in the same repository or to shared folders in the global repository. Shortcuts to folders in the same repository are known as local shortcuts. Shortcuts to the global repository are called global shortcuts.

Local Shortcuts

The wizard can reestablish local shortcuts to objects in shared folders if you have a current copy of the object in the target repository. You can keep these shortcuts intact by copying the necessary local shared folders to the target repository before you copy the deployment group or by including the object the shortcut references in the deployment group.

If the referenced object exists in the target repository, the wizard verifies that the copy is current. In typical mode, if you edit the original shared folder after you copy it to the target repository, the wizard asks you to copy it again. If you do not copy the shared folders before copying the deployment group, the wizard deletes all shortcuts and marks all affected mappings invalid.

In advanced mode, you can compare the contents of the folders to see which contains the most recent copies of referenced objects, and then decide to cancel and copy the shared folder again or proceed with the copy operation. When you compare folders, the wizard compares the version of the objects in the deployment group with the latest version of objects in the target folder.

Also, if you copy a local shortcut into the same folder that contains the object the shortcut references, the wizard cannot reestablish the shortcut. The wizard deletes the shortcut and marks all affected mappings invalid.

Global Shortcuts

If the deployment group contains global shortcuts, the wizard does not reestablish them when you copy them to the target repository. If you copy a global shortcut alone, the wizard completes the copy operation but does not copy the shortcut. If the global shortcut is part of a composite object you want to copy, the copy operation fails.

To ensure that global shortcuts are preserved when you copy a composite object, verify that a copy of the object, including the shortcut, exists in the target repository. When you copy the object for the first time, consider copying the entire folder. You can then use a deployment group to copy subsequent versions of the object.

Object Naming

You can create copies of objects with different names. As a result, you can add an object to a deployment group that has an existing copy in the target folder, but the copy has a different name. In this situation, the wizard detects the relationship between the objects and replaces the copy in the target folder with the object in the deployment group.

For example, you add the mapping m_Orders to a deployment group and copy it to the production repository. As you continue development, you change the name of the mapping in the development repository to m_OrdersWeekly. You add this new version of the mapping to a deployment group and copy it to the production repository. If the production repository is versioned, the wizard determines that m_Orders is an older copy of m_OrdersWeekly and replaces it, creating a new version. The latest version of the mapping in the production repository is now m_OrdersWeekly. If the production repository is non-versioned, the wizard determines that m_Orders is a copy of m_OrdersWeekly and replaces it with m_OrdersWeekly.

An object in the target repository might also have the same name as a deployment group object without being a copy of the object. The object may be of a different type. If this happens, the naming conflict causes the copy operation to fail.

For example, a mapping uses relational source src_Records in the development repository. You add the mapping to a deployment group and copy it to the production repository. Later, you delete src_Records from the production repository and create a new XML source, also named src_Records. If you then use a deployment group to copy the relational source src_Records to the target repository, the copy operation fails because the XML source src_Records has the same name, but is a different object.

Object Status

When you copy an object in a deployment group, the status of the source object may change if a copy of the object exists in the target folder.

The following table describes the status an object may take after copying a deployment group, depending on the status of the source and target objects:

Status of Deployment Group Object	Status of Target Repository Object	Deployment Wizard Action	Status of Target Repository Object After Copying
Active Object	Deleted	Deploys the object	Active
Active Object	Active	Deploys the object	Active
Deleted Object	Active	Deploys the object	Deleted
Deleted Object	Deleted	Skips the object	Deleted

Note: Non-reusable objects derive status from their parent composite objects. If a parent composite object has a deleted status, associated non-reusable objects also have a deleted status.

Steps to Copy a Deployment Group

Use the Copy Deployment Group Wizard to copy objects in a deployment group. You can perform all of the tasks listed in "Using the Copy Wizards" on page 108. You can also complete the following tasks:

- **Choose deployment folders.** You can choose the folders in the target repository you want to deploy.
- **Apply labels to source and target objects.** You can apply labels to the deployment group objects in the source and target repositories. For example, you may want to apply a label to the source and target objects that specifies when the source object version was deployed and when the target object version was created.
- **Move labels**. You can move labels from version to version in source and target repositories. For example, you might want to move a label from the last version to the latest version before you deploy an object. Or, you might want to deploy an earlier version of an object and apply the latest label to the object.

- **Clear the static deployment group when you finish copying.** You can remove the copied objects from a static deployment group when you finish copying them into the target repository.

Before you copy a deployment group, verify that existing objects in the target repository are not checked out or locked. Also, copy dependent shared folders for shortcuts in the deployment group.

If objects in the target repository are locked, by default the deployment operation waits until either the locks are acquired or you cancel the deployment. If you use *pmrep* to copy the deployment group, you can specify a deployment timeout period. If *pmrep* does not acquire the object locks within the timeout period, the deployment operation fails.

Note: The default behavior is different if you attempt to replace a folder and the target folder is locked. The deployment operation does not wait for the locks to be released. The deployment fails immediately, and an error message indicates that the target folder is in use.

1. Connect to the source and target repositories.
2. Select the deployment group to copy.
3. Drag or paste the deployment group to the target repository.

   The Copy Deployment Group Wizard appears, displaying the folder name and target repository name.
4. The Copy Deployment Group Wizard prompts you to select a mode:
   - **Typical.** The wizard uses the defaults for shortcuts to local and global shared folders.
   - **Advanced.** You can override the defaults for shortcuts to local and global shared folders. You can choose the shared folders to associate shortcuts. The wizard might have to determine how the folders are related before establishing shortcuts.
5. Click Next. The Copy Deployment Group Wizard prompts you for more information based on the content of the folders and the copy mode you selected.

   The Next button is disabled if object locks cannot be acquired in the target repository. When the objects in the target repository become available, the Next button is enabled. To stop the replacement, click Cancel. The wizard rolls back all changes.

The following table lists the dialog boxes and prompts that may appear when you copy a deployment group:

Copy Deployment Group Wizard Dialog Box	Modes	Description
Select Deployment Folders	Typical, Advanced	Folders you want to deploy objects to.
Override Deployment Folder	Typical, Advanced	Overrides the default selections for deployment folders.
Select Labels	Typical, Advanced	Selects a label in the source repository to apply to the copied object versions, and selects a label in the target repository to apply to the newly created object versions.
Clear Source Deployment Group	Typical, Advanced	Removes objects from the deployment group after the wizard completes the deployment operation.
Source/Target Comparison	Typical, Advanced	Specifies if objects in the target folder have been created or modified since the last copy.
Sequence Generators and Normalizers	Typical, Advanced	Retains current values for Sequence Generator and Normalizer transformations and XML generated keys. XML generated keys include primary and foreign keys in XML transformations.

Copy Deployment Group Wizard Dialog Box	Modes	Description
Mapping Variables	Typical, Advanced	Retains persistent values for mapping variables.
Dependency Information	Typical, Advanced	Copies dependency information for objects in mappings if it exists. The dependency information exists if you set the general options for the Designer to save MX data. The dependency information is organized in a format that enables reporting tools to skip steps such as fetching the entire mapping and parsing expressions when collecting information for reporting.
Retain Workflow Variable Persisted Values	Typical, Advanced	Retains persistent values.
Retain Workflow Run History	Typical, Advanced	Retains existing workflow run history in the target repository or folder. When you copy a deployment group, you cannot copy the workflow run history from the source repository or folder.
Retain Integration Service Assignment	Typical, Advanced	Retains the assigned Integration Service for workflows.
Database Connections	Typical, Advanced	Lists all database connections in the folder, indicating the connections for which you do not have permission to copy.
Message Queue Connections	Typical, Advanced	Lists all message queue connections in the folder.
FTP Connections	Typical, Advanced	Lists all FTP connections in the folder, indicating the connections for which you do not have permission to copy.
External Loaders	Typical, Advanced	Lists all external loader connections in the folder, indicating the connections for which you do not have permission to copy.
Application Connections	Typical, Advanced	Lists all application connections in the folder, indicating the connections for which you do not have permission to copy.
MMD Plug-in	Typical, Advanced	Lists plug-in application information upon which the source folder depends if the plug-in information does not exist in the target repository.
Integration Services	Typical, Advanced	Selects an Integration Service to associate with workflows. If the target repository contains less than two Integration Services, the wizard skips this step.
Local Shared Folders	Typical	Lists local shared folders in the target repository that you need to copy first. To preserve shortcuts and prevent invalid mappings, click Cancel and copy all listed local shared folders before copying the folder.
Outdated Shared Folders	Typical	Lists outdated folders in the target repository that you need to copy first. To preserve shortcuts and prevent invalid mappings, click Cancel and copy all listed local shared folders before copying the folder.

Copy Deployment Group Wizard Dialog Box	Modes	Description
Select Shared Folders	Advanced	Lists the folder that contains shortcuts and the folder to which you can establish shortcuts. You can choose to accept or override the shared folder.
Override Shared Folder	Advanced	Lists folders you can select to establish shortcuts if you choose to override the default folder selection in the Select Shared Folders dialog box.
Compare Folder	Advanced	Compares folders if the folders you choose in the Override Shared Folder dialog box are different. You can compare the folders using a one-way comparison. To compare the folder objects using two-way comparison, use the Compare Folders Wizard before you start the Copy Folder Wizard.
Compare Shared Folder - Compare Results	Advanced	Lists the results from the folder comparison, displays objects that exist in the local shared folder but not in the global shared folder, and displays objects that are older or newer than matching objects in the global shared folder. When you compare the folders using a one-way comparison, you can check the folder objects that excludes the mapping variable values. If there are differences between the folders, a message warns you that shortcuts to missing objects will be removed. The wizard takes you back to Select Shared Folders dialog box.
Copy Definition	Typical, Advanced	Copies the deployment group from the source repository to the target repository.
Deployment Group Exists	Typical, Advanced	Replaces an existing deployment group in the target repository. Shows either the existing copies of the deployment group in the repository or all the deployment groups in the repository.
Replace Conflicting Objects	Typical, Advanced	Replaces conflicting objects in the target non-versioned repository. You can choose to replace the conflicting object in the target repository with the object in the deployment group. Does not appear for versioned repositories.
Owner and Group	Typical, Advanced	Selects the owner for the copied deployment group in the target repository. Default is the current user.
Post-Validation	Typical, Advanced	Validates the objects in the target repository after you copy a deployment group to verify that the objects and dependent objects are valid.
Complete Deployment	Typical, Advanced	Copies the deployment group immediately after you complete the wizard and generates a deployment control file to use with the *pmrep* command line program. Or, both copies the deployment group and creates the deployment control file. You can select to create the deployment control file without copying the deployment group. If you do not specify an absolute path, the Repository Manager generates the deployment control file in the directory where the Repository Manager runs. The Repository Manager generates the deployment control file encoded in UTF-8 format.

Troubleshooting Copying Folders or Deployment Groups

When I try to copy a folder or deployment group to another repository, the operation fails and a database error indicates that insufficient free space exists in the target repository. This error occurs even though the target repository database has enough free space to accommodate the folder or deployment group that I am copying.

The target repository database needs enough free disk space to accommodate approximately twice the amount of space required by the folder or deployment group that you are copying. The target repository database requires the extra free space because the deployment operation first copies the data into temporary tables in the target database and then moves the data from the temporary tables to the target repository tables.

When I migrate objects from more than one source repository to a target repository, the target repository contents become corrupted.

PowerCenter maintains an association between the source object IDs and the target object IDs in the target repository. If you migrate objects of the same type and with the same name from multiple source repositories into a single target repository, the association in the target repository leads to issues, such as inconsistencies and duplications. When you perform migration from multiple source repositories to a single target repository, ensure that objects of the same type have unique names across the source repositories.

You can always perform migration from a single source repository to a target repository. When you do not perform bulk migration or migration from numerous folders, a workaround is to use object import or export to deploy objects across repositories.

CHAPTER 11

Exporting and Importing Objects

This chapter includes the following topics:

Exporting and Importing Objects Overview

In the PowerCenter Client, you can export repository objects to an XML file and then import repository objects from the XML file. Use the following client applications to export and import repository objects:

- **Repository Manager.** You can export and import both Designer and Workflow Manager objects.
- **Designer.** You can export and import Designer objects.
- **Workflow Manager.** You can export and import Workflow Manager objects.
- ***pmrep*.** You can export and import both Designer and Workflow Manager objects. You might use *pmrep* to automate exporting objects on a daily or weekly basis.

Exporting and importing an object is similar to copying an object from one folder or repository to another. For example, when you copy an object between folders or export and import that object, you can resolve object name conflicts. However, when you copy objects between folders or repositories, you must be connected to both repositories. When you export an object from one repository and import the object into another repository, you do not need to be connected to both repositories.

Export and import objects between repositories with the same version. Informatica does not support imported objects from a different release.

You can export and import repository objects to accomplish the following tasks:

- **Deploy metadata into production.** After you test a mapping in a development repository, you can export it to an XML file and then import it from the XML file into a production repository. You might export and import objects to incrementally deploy metadata by exporting and importing part of a composite object.
- **Archive metadata.** You can export objects to an XML file that you no longer need before removing them from the repository.
- **Share metadata.** You can share metadata with a third party. For example, you want to send a mapping to someone else for testing or analysis, but you do not want to disclose repository connection information for security reasons. You can export the mapping to an XML file and edit the repository connection information before sending the XML file. The third party can import the mapping from the XML file and analyze the metadata.
- **Search and replace property names in an entire repository object.** You can search for a property name and replace all occurrences of it with a different name. For example, you have a mapping with an unconnected Lookup transformation. You want to change the name of a port in the unconnected Lookup transformation. Several other transformations call the lookup port through an expression, so you want to make sure you change the port name in all other expressions. You can export the mapping to an XML file and open it in a text editor. Search for the old port name and replace all references to it with the new port name. Then import the mapping into the repository.
- **Copy metadata between repositories.** You can copy objects between repositories that you cannot connect to from the same client. Export the object and transfer the XML file to the target machine. Then import the object from the XML file into the target repository.
- **Create mappings.** You can export an existing mapping and use Mapping Architect for Visio to turn the mapping into a mapping template. Once a mapping template is created in Mapping Architect for Visio you can import multiple mappings into the repository.

You can also export and import relational sources and targets to share metadata with other business intelligence and data modeling tools.

Working with Objects and Object Types

You can export and import the following repository objects:

- Sources
- Targets
- Transformations
- Mapplets
- Mappings
- User-defined functions
- Tasks
- Sessions
- Schedulers
- Session configurations
- Worklets
- Workflows

When you export and import repository objects, you can choose to export and import the following types of objects:

- **Multiple object types.** You can export and import one or more object types. The combination of object types you can export and import depends on the PowerCenter Client you use.
- **Multiple objects.** You can export and import one or more objects.
- **Objects from multiple folders.** Using the Repository Manager or *pmrep*, you can export and import objects from one or more folders in the same repository. Also, you can do this when you access a query result from the Designer, Workflow Manager, or Repository Manager.
- **Dependent objects.** You can export and import an object with or without its dependent objects.

Code Pages

To ensure no data is lost when you import an object, you can export and import objects between repositories with compatible code pages with the PowerCenter Client. The code page of the originating repository must be a subset of the destination repository code page. If the two repository code pages are not compatible, the PowerCenter Client displays an error message and does not import any object.

The XML and DTD Files

When you export repository objects, the PowerCenter Client creates an XML file that contains the metadata of the exported repository objects. Use this same file to import the repository objects into a repository.

The XML file has an associated Document Type Definition (DTD) file called powrmart.dtd. When you export repository objects, the PowerCenter Client creates the XML file based on the structure specified in powrmart.dtd. When you import repository objects, the PowerCenter Client validates the XML file against powrmart.dtd.

When you install PowerCenter, the installation program copies powrmart.dtd into the client installation directory. When you export or import an object, the PowerCenter Client looks for powrmart.dtd in the client installation directory. If powrmart.dtd is not in the client installation directory, you cannot import repository objects.

An XML file is valid if it complies with the constraints expressed in its associated DTD. Therefore, an exported XML file is valid if it complies with the constraints expressed in powrmart.dtd. For example, if powrmart.dtd states that an element must occur once in an XML file, the XML file is invalid if the element occurs more than once or not at all.

For more information about XML, see the W3C specifications for XML at the following location:http://www.w3.org/.

Note: If you modify an exported XML file, you need to make sure that the XML file conforms to the structure of powrmart.dtd. You also need to make sure the metadata in the XML file conforms to Designer and Workflow Manager rules. For example, when you define a shortcut to an object, define the folder in which the referenced object resides as a shared folder. Although PowerCenter validates the XML file before importing repository objects from it, it might not catch all invalid changes. If you import into the repository an object that does not conform to Designer or Workflow Manager rules, you might cause data inconsistencies in the repository.

Do not modify the powrmart.dtd file.

CRCVALUE Codes

Informatica restricts which elements you can modify in the XML file. When you export a Designer object, the PowerCenter Client might include a Cyclic Redundancy Checking Value (CRCVALUE) code in one or more elements in the XML file. The CRCVALUE code is another attribute in an element.

When the PowerCenter Client includes a CRCVALUE code in the exported XML file, you can modify some attributes and elements before importing the object into a repository. For example, VSAM source objects always contain a CRCVALUE code, so you can only modify some attributes in a VSAM source object. If you modify certain attributes in an element that contains a CRCVALUE code, you cannot import the object.

For example, if you modify the OWNERNAME attribute in the source object, you cannot import the source into the Designer.

The following XML shows part of the element for a source object with the CRCVALUE code:

```
<SOURCE NAME ="SALES_FILE" DBDNAME ="SALES.CBL" IBMCOMP ="YES"
CRCVALUE ="3108520154" OWNERNAME ="" DESCRIPTION ="" BUSINESSNAME ="" DATABASETYPE
="VSAM" ...>
</SOURCE>
```

The CRCVALUE attribute for the element SOURCE is 3108520154.

Note: The PowerCenter Client includes CRCVALUE codes in the XML file when you export Designer objects.

Related Topics:

- "Modifying an Exported XML File" on page 135

Exporting and Importing Multiple Objects and Object Types

You can export and import multiple objects and multiple object types at the same time. However, the combination of object types depends on the PowerCenter Client application you use.

The following table lists the multiple objects you can export and import:

PowerCenter Client Application	Options for Exporting	Options for Importing
Repository Manager	- Multiple objects from one folder - Multiple object types from one folder For example, you can export multiple mappings to the same file.	- Multiple objects from multiple folders - Multiple object types from multiple folders When you import objects from multiple folders, you can choose which folders to import into.
pmrep	- Multiple objects from multiple folders - Multiple object types from multiple folders For example, you can export reusable transformations and reusable worklets to the same file.	- Multiple objects from multiple folders - Multiple object types from multiple folders When you import objects from multiple folders, you can choose which folders to import into using the control file.

PowerCenter Client Application	Options for Exporting	Options for Importing
Designer	- Multiple sources, targets, or reusable transformations from one folder For example, you cannot export both sources and targets from the Navigator. You cannot export multiple mappings or mapplets. You cannot export multiple object types.	- Multiple objects from one folder - Multiple object types from one folder You can only import Designer objects.
Workflow Manager	- Multiple reusable Email, Session, and Command tasks from one folder - Multiple worklets from one folder - Multiple workflows from one folder For example, you can export a reusable Email task and a reusable Session task.	- Multiple objects from one folder - Multiple object types from one folder You can only import Workflow Manager objects.

Note: You can export different object types from all PowerCenter Client tools by exporting the results of an object query.

Working with Dependent Objects

When you export an object, the PowerCenter Client exports certain dependent objects by default. The PowerCenter Client does not export all dependent objects. A dependent object is an object that is used by another object. For example, a source definition referenced by a shortcut is a dependent object of that shortcut. A dependent object is a child object to the parent object that uses the dependent object.

The following table lists the dependent objects that the PowerCenter Client includes in the XML file by default:

Parent Object	Dependent Child Objects Exported
Mapping	Sources, targets, reusable and non-reusable transformations, mapplets, and user-defined functions.
Mapplet	Sources and reusable transformations.
Source with foreign key	Source definition containing the primary key.
Target with foreign key	Target definition containing the primary key.
Shortcut	The object the shortcut references.
Any repository object	Any reusable or non-reusable metadata extension associated with the object. [1]
Session	Session configuration and reusable and non-reusable tasks when you export from any client application. Mapping used by the session when you export from the Repository Manager or *pmrep*.

Parent Object	Dependent Child Objects Exported
Transformation	User-defined functions.
User-defined function	User-defined functions.
Worklet	Reusable and non-reusable tasks, sessions, worklets, and user-defined functions.
Workflow	Scheduler and reusable and non-reusable tasks, sessions, worklets, and user-defined functions.

1. The PowerCenter Client always exports metadata extensions. Verify that you register a plug-in in the destination repository before you import an object using a vendor-defined metadata extension associated with the plug-in. If the plug-in is not registered, the PowerCenter Client imports the object without the metadata extension.

When you export and import objects, you can export and import any of the following combination of objects:

- **Parent object with dependent child objects.** The XML file contains metadata for parent and child objects. The PowerCenter Client exports the dependent child objects listed in "Working with Dependent Objects" on page 130 by default.
- **Parent object without dependent child objects.** The XML file contains metadata for the parent object, but not the child object.

Exporting and Importing Parent Objects

You can choose to export a parent object with or without its dependent child objects. You might want to export and import an object without its dependent objects if you change a workflow property, such as a workflow variable, but you did not change any task in the workflow.

You can choose the export options in the Export Options dialog box.

The following table describes the options in the Export Options dialog box:

Export Option	Description
Export Primary Key Tables When Exporting Sources/Targets with Foreign Keys	When you export a source or target containing a foreign key, the PowerCenter Client exports the source or target containing the primary key.
Export Original Object Referred by the Shortcut When Exporting Shortcuts	When you export a shortcut, the PowerCenter Client exports the actual object referenced by the shortcut.
Export Reusable Objects Used by Objects Being Exported	When you export a mapping, mapplet, worklet, or workflow, the PowerCenter Client exports all reusable objects used by the parent object. For example, the PowerCenter Client exports all sources, targets, and reusable transformations when you export a mapping.
Export Non-Reusable Objects Used by Objects Being Exported	When you export a mapping, mapplet, worklet, or workflow, the PowerCenter Client exports all non-reusable objects used by the parent object. For example, the PowerCenter Client exports all non-reusable transformations for a mapping or mapplet, and all non-reusable tasks for a worklet or workflow.

To access the Export Options dialog box, click the Advanced Options link in the Export dialog box when you export objects.

When you export an object with its dependent child objects, the PowerCenter Client exports the metadata for the parent object and the dependent objects. When you export an object without its dependent objects, the PowerCenter Client exports the metadata for the object, but does not export metadata for the dependent objects. However, the object you export still references the dependent objects even though they do not exist in the XML file.

When you import an object that uses dependent objects, the results differ depending on whether the dependent objects exist in the XML file:

- **Dependent objects exist in XML file.** When you import an object, the PowerCenter Client imports all dependent objects. For example, you export a mapping including its dependent objects. When you import the mapping, the PowerCenter Client imports all objects used by the mapping, such as the sources.
- **Dependent objects do not exist in XML file.** When you import an object, the PowerCenter Client looks for an object in the destination folder with the same name. If the PowerCenter Client finds an object with the same name, it uses the object in the destination folder. If the PowerCenter Client does not find an object with the same name, it does not import the object.

  For example, you create a workflow with multiple worklets, sessions, and tasks. You change the link condition between two tasks. You want to update the link condition when you import the workflow into a different folder. Export the workflow and do not export the reusable and non-reusable tasks. When you import the workflow, the PowerCenter Client imports the workflow metadata. The PowerCenter Client uses the worklets, sessions, and tasks that exist in the destination folder.

Working with Sessions

When you export a session, the associated mapping must be valid. However, the session does not need to be valid before you export it. You might want to export an invalid session to send to someone else to troubleshoot.

When you export a session from the Workflow Manager, the PowerCenter Client exports the session, but not the associated mapping. However, when you export a session from the Repository Manager, the PowerCenter Client exports the session and the associated mapping.

You can also create an XML file that contains both the session and mapping objects by using *pmrep* or the query results accessed from the Repository Manager to select both objects and export them. Or, use the Designer to export the mapping and the Workflow Manager to export the session. Then edit one of the XML files to include both objects.

To import a session, the associated mapping must exist in the target folder and be valid. If the mapping does not exist or is invalid, the PowerCenter Client does not import the session. However, when you use the Repository Manager or *pmrep*, you can import the session if the XML file contains the metadata for the associated mapping.

Working with Object Versions

You can export one version of an object at a time. When you export an object from the Navigator or workspace, the PowerCenter Client exports the latest version of the object. If you want to export an earlier version of an object, you can select it from a query result or object history. In the View History or Query Results window, select the objects to export and choose Tools-Export to XML File. You can select multiple object versions to export, but the PowerCenter Client exports only the latest version selected for that object.

For example, the query results contain two mappings that use different versions of the same source. If you export both mappings, the PowerCenter Client exports the latest version of the source.

When you import an object that exists in the target folder, the PowerCenter Client handles object versions differently depending on how you resolve the object conflict. You can resolve an object conflict by replacing, renaming, or reusing the object.

For example, the target folder contains a target called WEEKLY_ORDERS and the latest version is three. You import a target with the same name. When you replace the target, the PowerCenter Client changes the existing target definition to version four. When you reuse the target, the PowerCenter Client does not change the version of the existing target definition. When you rename the target, the PowerCenter Client creates a new target definition and assigns it a version of one, and does not change the version of the existing target definition.

Note: You cannot export deleted objects from a query result or object history.

Working with Shortcuts

You can export and import local and global shortcuts.

When you export a shortcut, you can choose to export the object the shortcut references. You might want to export the referenced object if it does not exist in the destination repository.

When you import a shortcut, you can specify folders for the shortcut and the referenced object. You can import a shortcut if the shared folder in the target repository on which you have read-only permission contains the original object. If you want to import a shortcut and the object concurrently, you need write permission.

When you import a shortcut, the PowerCenter Client creates a shortcut in the folder that you specify. The new shortcut points to the object in the folder that you specify for the referenced object.

You always specify a folder for the referenced object, whether or not you import the referenced object into that folder. The PowerCenter Client searches for the referenced object in the folder you specify to establish the shortcut. The import behavior depends on the target folder permission and the location of the referenced object.

The following table describes the import behavior based on the target folder permission and the location of the referenced object:

Permission	Folder contains the referenced object	Folder does not contain the referenced object
Read	The PowerCenter Client imports the shortcut object into the destination repository.	The PowerCenter Client does not import the shortcut object.
Write	The PowerCenter Client imports the shortcut object into the destination repository.	The PowerCenter Client imports the actual object into the destination repository if the XML file contains the metadata for the referenced object.

When you import a shortcut into a local repository, you can specify a folder from the local repository or the global repository in the domain. When you import a shortcut into a global repository, you can specify a folder from the global repository.

Shortcut Types

The type of shortcut the PowerCenter Client creates in the destination repository depends on the folders you specify for the shortcut and the referenced object. When both the shortcut and referenced object exist in the same repository, the PowerCenter Client creates a local shortcut. When the shortcut exists in a local repository and the referenced object exists in a global repository in the same domain, the PowerCenter Client creates a global shortcut. The type of shortcut the PowerCenter Client creates does not depend on the shortcut type specified in the XML file.

Importing Shortcuts to Sources

When the PowerCenter Client imports the object instead of the shortcut, the imported object does not inherit any changes you make to the original object in the source repository. The XML file defines the metadata for the object.

Use the imported object as you would the original object. However, if the object is a source definition, you might need to rename the source definition.

For example, you export a shortcut named Shortcut_To_Employees and you also export the referenced object. You use the Designer to import the shortcut into a different repository. In the Import Wizard, you choose to import the shortcut, but you do *not* import the referenced object. Also in the Import Wizard, you choose a folder in the destination repository to specify the location of an existing referenced object. However, the folder does not contain an object with the same name as the referenced object specified in the XML file.

The PowerCenter Client does not find an object with the same name in the folder you specified, so it imports the actual object instead. The Designer creates a new source definition in the destination folder named Shortcut_To_Employees. This source definition is a copy of the original object, and is not a shortcut. When you use the source definition in a mapping, the default SQL used to extract data from the source defines the source as Shortcut_To_Employees. If the source table in the source database is named Employees, you must rename the source definition (Employees) or enter an SQL override for the source qualifier connected to the source definition (renaming the source table Employees) for the Integration Service to extract source data.

Exporting Objects

When you export an object, the PowerCenter Client writes the definition of the object to an XML file. The XML file complies with powrmart.dtd and uses the same code page as the repository from which it was exported. After you export objects, you can modify the XML file.

When you export the latest version of an object, the PowerCenter Client exports either the version of the object saved in the repository or the version of the object you have open in the Designer or Workflow Manager:

- **Version saved in the repository.** When you export an object from the Repository Manager, *pmrep*, or the query results accessed from the Repository Manager, the PowerCenter Client exports the version of the object saved in the repository.
- **Version you have open in the Designer or Workflow Manager.** When you export an object from the Designer, Workflow Manager, or query results accessed from the Designer or Workflow Manager, the PowerCenter Client exports the latest version of the object, including any change you made to it since you last saved it to the repository.

However, when you export shortcuts from the query results accessed from the Designer, the Designer exports either the version of the referenced object you have open in the Designer or the version of referenced object saved in the repository, depending on the other objects you export.

For example, you run an object query from the Designer. The query result contains the following objects:

- Shortcut_to_Source1 in the Orders folder. The shortcut references Source1 in the Items folder.
- Source1 in the Items folder
- Mapping1 in the Items folder
- Target1 in the Sales folder

The Designer behavior depends on the other objects you export:

Exported Objects	Designer Export Behavior
- Shortcut_to_Source1 from the Orders folder - Target1 from the Sales folder	Designer exports the saved version of Source1 because you do not export any object from the same folder that contains the referenced object, Source1.
- Shortcut_to_Source1 from the Orders folder - Mapping1 from the Items folder	Designer exports the version of Source1 you have open in the Designer because you export an object, Mapping1, from the same folder that contains the referenced object, Source1. Therefore, the Designer exports the latest versions of all objects in the Items folder, including changes you made to them since you last saved the repository.

Modifying an Exported XML File

After exporting an object, you can modify the XML attributes before importing the object into a repository. For example, suppose you have inconsistent column names across a mapping. You want to globally search and replace Cust_ID and Customers_ID with Customer_ID. You can export the mapping into an XML file, modify the values in the XML file, and then import the mapping with the new values.

Modifying an XML file is an easy way to change the metadata for a repository object. However, Informatica restricts the elements and attributes you can modify in the XML file.

Use the following rules when you modify XML files:

- Define only the metadata that you can create in the Designer or Workflow Manager. For example, do not associate a Source Qualifier transformation with a VSAM source.
- Do not modify powrmart.dtd.
- Verify that the structure of the XML file complies with the constraints in powrmart.dtd. For example, if powrmart.dtd says that an element must include a specified child element, make sure you include the child element.
- You can modify the BUSINESSNAME and DESCRIPTION attributes in any element.
- You can modify all attributes listed in "Modifiable Objects" on page 136, regardless of CRCVALUE codes.
- You cannot modify attributes in an element containing a CRCVALUE unless the attribute is listed in "Modifiable Objects" on page 136.
- You cannot modify attributes in an element if its parent element contains a CRCVALUE code, unless the attributes are listed in "Modifiable Objects" on page 136.

Modifiable Objects

You can modify some attributes and elements in an XML file. In the following table, the Modifiable Attributes column lists the attributes you can modify for an exported object and import. The Create New column indicates which objects you can define directly in the XML file and import.

The following table lists the repository objects that you can modify:

Repository Object	Type	Modifiable Attributes	Create New
Source	Relational	All	Yes
Source	Flat File	All	Yes
Source	VSAM	BUSINESSNAME DESCRIPTION	-
Source	MQ	BUSINESSNAME DESCRIPTION	No
Source	XML	BUSINESSNAME DESCRIPTION	No
Source	PeopleSoft	BUSINESSNAME DESCRIPTION	No
Source	SAP table	BUSINESSNAME DESCRIPTION	No
Source	SAP ALE IDoc	All	Yes
Source	TIBCO	All	Yes
Source	Null source	All	Yes
Target	Relational	All	Yes
Target	SAP BW	BUSINESSNAME DESCRIPTION	No
Target	XML	BUSINESSNAME DESCRIPTION	No
Target	MQ	BUSINESSNAME DESCRIPTION	No
Target	TIBCO	All	Yes
Target	Null target	All	Yes
Reusable Transformation	All except the Java transformation	All	Yes
Mapping	Relational	All	Yes

Repository Object	Type	Modifiable Attributes	Create New
Mapping	Flat File	All	Yes
Mapping	VSAM	All	No
Mapping	MQ	All	No
Mapping	XML	All	No
Mapping	PeopleSoft	BUSINESSNAME DESCRIPTION	No
Mapping	SAP table	BUSINESSNAME DESCRIPTION	No
Mapping	SAP ALE IDoc	All	Yes
Mapping	TIBCO	All	Yes
Mapplet	Relational	All	Yes
Mapplet	Flat File	All	Yes
Mapplet	PeopleSoft	BUSINESSNAME DESCRIPTION	No
Mapplet	Siebel	BUSINESSNAME DESCRIPTION	No
Mapplet	SAP table	BUSINESSNAME DESCRIPTION	No
Mapplet	SAP ALE IDoc	All	Yes
Mapplet	TIBCO	All	Yes
Session	Reusable	All	Yes
Session	Non-reusable	All	Yes
Task	Reusable	All	Yes
Task	Non-reusable	All	Yes
Worklet	Reusable	All	Yes
Worklet	Non-reusable	All	Yes
Workflow	-	All	Yes

Importing Objects

You can import objects from a valid XML file. The XML file must comply with powrmart.dtd. You can import objects that you exported from the same repository or a different repository.

When you import an object, the PowerCenter Client performs the following tasks:

1. Validates the XML file against powrmart.dtd.
2. Parses the XML file.
3. Validates the objects in the XML file.
4. Creates the objects in the repository.

When you import an object in the Designer, Workflow Manager, or Repository Manager, the Import Wizard appears. When you import using *pmrep*, you use a control file to specify the same import options in the Import Wizard.

You can complete the following actions in the Import Wizard:

- **Choose the XML file.**
- **Choose which objects to import.** You can choose all or some objects listed in the XML file. If the XML file contains both Designer and Workflow Manager objects, the Import Wizard shows Designer objects when you use the Designer, and Workflow Manager objects when you use the Workflow Manager. You can import all object types using the Repository Manager.
- **Match folders.** When you use the Repository Manager to import, you can match folders listed in the XML file with folders in the destination repository.
- **Check in the objects and apply a label.** When you use the Repository Manager to import objects into a versioned repository, you can check in the objects after you import them. You can enter check in comments in the Import Wizard. If you check in the objects, you can apply a label to them. Choose an existing label or create a new one.
- **Resolve object conflicts.** When you import an object into a folder that contains an object with the same name, you can choose to rename, replace, or reuse the object.

Validating XML Files Against the DTD

You import objects from a valid XML file. The PowerCenter Client validates the XML file against the DTD and parses the XML file before importing. If the XML file is not valid, the PowerCenter Client displays an error message and does not import the objects. The DTD file, powrmart.dtd, is located in the PowerCenter Client directory.

Related Topics:

- "The XML and DTD Files" on page 128

Validating Objects

You import valid objects into the repository. The PowerCenter Client validates each object in the XML file to ensure that it conforms to the PowerCenter specifications for that object. For example, a mapplet cannot contain a target definition.

In addition, the PowerCenter Client validates objects with CRCVALUE codes to ensure that certain elements and attributes of Designer objects in the XML file have not been modified. The CRCVALUE code determines whether or not you can modify certain elements and attributes.

The PowerCenter Client does not import objects with CRCVALUE codes that have been modified nor objects that do not conform to PowerCenter specifications.

Related Topics:

- "The XML and DTD Files" on page 128

Resolving Object Conflicts

When you import objects, sometimes an object in the XML file has the same name as an object in the destination folder. You can choose to resolve object conflicts in the following ways:

- Create general object resolution rules.
- Resolve specific object conflicts.

Resolving General Object Conflicts

You can resolve some object conflicts by creating rules that apply to a set of objects. Create rules on the Specify Rules for Conflict Resolutions page of the Import Wizard. When you create an object resolution rule, the PowerCenter Client resolves object conflicts for objects to which the rule applies.

You can create multiple rules. Use the buttons in the Import Wizard to move the rules up and down. The PowerCenter Client applies the rules to objects in order. If multiple rules apply to one object, the PowerCenter Client applies the uppermost rule.

The following table describes the different columns you define for each rule:

Column	Description
Select Criteria	Choose the set of objects the rule applies to. You can choose the following sets of objects: - Objects with label. Applies to all objects with the label you choose. You can select this option when you import objects into a versioned repository. - Objects in query. Applies to all objects that result from the object query you choose. You can select this option when you import objects into a versioned repository. - Objects of type. Applies to objects of the type you choose. - All objects. Applies to all objects you import.
Select Value	Choose a value that modifies the first column. For example, if you select Objects with label in the first column, choose the label name in this column.
Select Resolution	Choose how to resolve the object conflicts. You can resolve conflicts using the following methods: - Replace. Replaces the existing object in the destination folder. - Reuse. Uses the existing object in the destination folder. - Rename. Creates a new object in the destination folder with a new name. When you choose Rename, you can specify a different name in the Conflict Resolution Wizard. - Prompt User. You can choose a resolution on an object by object basis. When you choose Prompt User, you can define the specific conflict resolutions in the Conflict Resolution Wizard.

After you create general object resolution rules, you can resolve specific object conflicts using the Import Wizard.

Resolving Specific Object Conflicts

Some object conflicts may still exist after you define rules to resolve conflicts. You can resolve specific object conflicts in the Import Wizard.

The Import Wizard displays all folders listed in the XML file. It also displays the object resolution status for objects in each folder:

- **Unresolved.** Some objects in this folder conflict with objects in the target folder. Click Resolve to resolve the object conflicts. You must resolve all object conflicts before you can import objects.
- **Resolved.** No object in this folder conflicts with objects in the target folder. The Import Wizard is ready to import these objects. However, you can view or edit the object conflict resolutions by clicking View/Edit.

When the Import Wizard detects unresolved conflicts for objects in a folder, it opens the Conflict Resolution Wizard.

The Conflict Resolution Wizard is similar to the Copy Wizard. The user interface is similar and you resolve the same type of object conflicts in both.

After you resolve object conflicts in the Conflict Resolution Wizard, you return to the Import Wizard.

Importing Objects from Informatica Analyst

The PowerCenter Repository Service imports Informatica Analyst objects after a data integration analyst exports the mapping specification logic from the Analyst tool.

If the data integration analyst exports the mapping specification and selects a target during export, the Repository Service imports the following objects:

- Sources
- Target
- Mapplet

The export process creates a mapping that uses the mapplet as a source and the exported target, a non-reusable session that contains the mapping, and a workflow that contains the non-reusable session. Create the connection objects for the sources and target before running the workflow.

If the data integration analyst exports the mapping specification and does not select a target during export, the Repository Service imports the following objects:

- Sources
- Mapplet

Create a mapping that uses the mapplet as a source and use a target. Create the session, workflow, and connection objects for the sources and target.

Importing Objects from Informatica Developer

You can import objects from a PowerCenter import XML file created in Informatica Developer.

The import XML file must be an XML file created for import into PowerCenter, not an XML file created for import into the Developer tool. Use the Import Wizard to import objects from a PowerCenter import XML file.

You can import mappings, mapplets, and mapping and mapplet dependent objects from a PowerCenter import XML file created in the Developer tool. The import XML file does not contain sessions or workflows, and it does not contain source and target connection information. If you import a mapping or mapplet that contains data quality transformations, you must have the Informatica Data Quality Integration for PowerCenter plug-in.

Note: You cannot import a Stored Procedure transformation from the Developer tool.

Developer tool users can export objects from multiple folders into a single XML file. In the Developer tool, objects in different folders can have the same object name. When you import objects into PowerCenter, you must choose a single target folder. If objects have the same name, you must specify different names when you import the objects.

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 from the Developer tool. 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.

Differences in Imported Objects

The Developer tool updates mappings and mapplets in the PowerCenter import XML file to ensure that the objects are valid PowerCenter objects.

The Developer tool makes changes to mapplets, port names, and data quality transformations.

Mapplets

The Developer tool makes the following changes to 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.

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.

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 Client 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.

Steps to Export Objects

You can export objects from the repository using the Designer, Workflow Manager, Repository Manager, query result, or object history.

To export objects from the query result or object history, select the objects to export and choose Tools-Export to XML File.

To export an object from the Designer, Workflow Manager, or Repository Manager:

1. Open the folder that contains the objects you want to export.
2. In the Navigator or workspace, select the objects to export.
3. Click Repository > Export Objects.
4. To choose which dependent objects to export, click Advanced Options.
5. In the Export Options dialog box, choose which dependent objects to export and click OK.
6. In the Export dialog box, navigate to the directory where you want to save the XML file. Enter a name for the XML file and click Save.

   The PowerCenter Client exports the objects to an XML file and displays export status in the Exported Objects dialog box.
7. Click View File to view the XML file that the PowerCenter Client creates.
8. Click Errors and Warnings tabs to view errors that may have occurred.
9. Click Close in the Exported Objects dialog box.

Steps to Import Objects

You can import objects into the repository using the Designer, Workflow Manager, or Repository Manager.

You can compare objects when importing objects with the Import Wizard.

To import an object:

1. Open the folder into which you want to import an object.
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 objects listed under that node. For example, when you select Sources and click Add, the Import Wizard adds all sources for that folder. Or, when you click a particular database definition node, the Import Wizard imports all sources listed under that database definition node. After you add an object to the list of objects to import, the Import Wizard displays a check mark on the icon for objects in the Objects in File pane.

   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.

   Note: When you import objects using the Designer or Workflow Manager, you can select objects from one folder. When you import objects using the Repository Manager, you can select objects from multiple folders from one repository.
6. Click Next.

   The Match Folders step of the Import Wizard appears when you import objects using the Repository Manager, or when you import a shortcut object in the Designer. 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 Choose Options step of the Import Wizard appears when you use the Repository Manager to import objects into a versioned repository. You can check in the objects and apply labels to the them after importing.
10. To check in all objects after importing them, select Check In and enter comments in the comment field.
11. To apply a label to all objects you import, select Apply Label and click Select Label. In the Label Browser dialog box, choose the label and click OK.

    You can only apply a label to the objects if you choose to check them in.
12. 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.
13. To create a new rule, click New Rule. Choose to which objects to apply the rule and select a resolution.
14. 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.

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

    The Diff Tool window appears.

    You can save the comparison as a text or HTML file.

    If the objects in the XML file exist in the target repository, the Targets window appears instead of the Diff Tool window.

16. Resolve object conflicts as they appear in the Conflict Resolution Wizard. Click Next to proceed through the Conflict Resolution Wizard.
17. 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 proceeds 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.

18. Click Import in the Import Wizard to import the objects into the repository. The PowerCenter Client 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.

19. Click Done.

Troubleshooting Exporting and Importing Objects

When I tried to import a shortcut to an object, the Designer imported the actual object instead of the shortcut.

To import a shortcut to an object into a repository, the Designer must be able to connect to the source repository to reestablish the shortcut. When it cannot connect to the source repository, it imports the object the shortcut references using the metadata in the XML file.

I imported a mapping from an XML file I modified, but the Designer displays a message that it is not valid.

Make sure that the 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.

I imported a mapping from the Developer tool with an SAP source, but the Designer displays a message that it is not valid.

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 Client imports the mapping without the source. You must manually create the SAP source in PowerCenter and add it to the mapping.

CHAPTER 12

Exchanging Metadata

This chapter includes the following topics:

Exchanging Metadata Overview

Use the Repository Manager to share source and target metadata with other business intelligence (BI) and data modeling tools, such as Business Objects Designer. PowerCenter uses the Meta Integration® Model Bridge (MIMB) from Meta Integration Technology, Inc. to exchange metadata with other BI and data modeling tools. MIMB uses the specifications in powrmart.dtd to exchange metadata with PowerCenter.

Use metadata exchange to synchronize definitions between PowerCenter and third-party tools. For example, you might export some fact and dimension targets to Business Objects Designer and then add a column to each target using Business Objects Designer. You can then import those targets into a repository to update the target definitions with the changes you made in Business Objects Designer.

The Repository Manager uses a wizard to guide you through the export or import process. The wizard prompts you for different options, depending on the BI or data modeling tool. Use the Conflict Resolution Wizard to resolve conflicts between objects when you import metadata. The Conflict Resolution Wizard is similar to the Copy Wizard.

To exchange metadata, you export the metadata from the source tool and import the metadata into the target tool. PowerCenter can be the source or target tool. To exchange metadata between PowerCenter and another tool, use one of the follow methods:

- **Use PowerCenter as the source tool.** In PowerCenter, export metadata to a file recognized by the target tool, and then use the target BI or data modeling tool to import metadata from the file.
- **Use the BI or data modeling tool as the source tool.** In the source BI or data modeling tool, export metadata to a file recognized by the source tool, and then use PowerCenter to import metadata from the file.

To export metadata, select an object and click Repository > Export Metadata. To import metadata, select a folder and click Repository > Import Metadata.

You can export to and import from the following BI and data modeling tools:

- Adaptive Repository Foundation
- Business Objects Data Integrator
- Business Objects Designer
- CA ERwin Data Modeler 4.x and CA ERwin Data Modeler 7.x
- CA ERwin 3.0 (export only), CA ERwin 3.x (ERX), and CA ERwin 3.5x (export only)
- CA Gen
- Cognos Impromptu (import only)
- Cognos ReportNet Framework Manager
- Embarcadero ER/Studio
- Hyperion Application Builder
- Hyperion Essbase Integration Services
- IBM DB2 Cube Views
- IBM DB2 Warehouse Manager
- IBM Rational Rose
- Microsoft Visio Database
- MicroStrategy
- OMG CWM
- Oracle Designer
- Oracle Warehouse Builder
- Popkin System Architect
- SAS ETL Studio
- Select SE
- Silverrun-RDM
- Sybase PowerDesigner

Note: You can also exchange metadata with BI and data modeling tools by using the Export Objects and Import Objects menu commands. You do not need a PowerCenter Metadata Exchange option license key, but you must be able to export or import XML files that conform to powrmart.dtd.

Related Topics:

- "Exporting and Importing Objects" on page 126
- "Steps to Export Metadata" on page 149
- "Steps to Import Metadata" on page 150

Working with Column Properties

Not all BI and data modeling tools use all column properties that PowerCenter uses, such as precision and scale. Also, not all tools support all datatypes that PowerCenter supports. For example, Business Objects Designer does not support binary datatypes and it does not use precision, scale, and not null information. When you export a binary column to Business Objects Designer, Business Objects Designer changes the datatype to a string and does not preserve the values for the precision, scale, or not null properties.

When you import metadata into PowerCenter from a tool that does not use all column properties, the Metadata Import Wizard uses default values for the column properties. However, you can retain the column

properties if a source or target of the same name exists in the folder. To retain the column properties, enable the Retain Physical Properties attribute in the Metadata Import Wizard. You might want to do this if you export metadata to a tool that does not use all column properties, modify the metadata in the other tool, and then import the modified metadata back into PowerCenter.

Rules and Guidelines for Exchanging Metadata

Consider the following rules and guidelines when you exchange metadata with BI or data modeling tools:

- You can export and import relational source and target definitions.
- You can import multiple source and target definitions at a time.
- You can export multiple source definitions or multiple target definitions at a time. You cannot export source and target definitions at the same time.
- You cannot export cubes and dimensions. However, you can export targets to some tools as dimensions.
- You cannot export shortcuts.
- When you export a source or target with a foreign key, the Repository Service also exports the source or target containing the corresponding primary key.
- When you import metadata into PowerCenter through MIMB, keys and referential integrity constraints are not persisted.
- You must have a Metadata Exchange option license to exchange metadata for a specific tool.

Working with Metadata Extensions

Some third-party tools store user-defined properties. A user-defined property is metadata you define for an object within the tool, such as metadata extensions you define in PowerCenter. MIMB preserves user-defined properties when you exchange metadata with third-party tools that support user-defined properties.

For example, when you export metadata to IBM Rational Rose, MIMB converts user-defined metadata extensions to user-defined properties in Rational Rose. Rational Rose creates a tab labeled Informatica with a user-defined property for each user-defined metadata extension. Also, when you import metadata into PowerCenter from Rational Rose, MIMB converts each user-defined property on the Informatica tab into a user-defined metadata extension.

You might want to create user-defined metadata extensions in the source and target definitions to specify fact and dimension tables in a start schema.

For more information about which third-party tools support user-defined properties, consult the third-party tool documentation.

Working with Star Schemas

Use PowerCenter to create a star schema of fact and dimension tables. You can create these definitions in the Target Designer, or you can use the mapping wizards. When you export relational target definitions in a star schema to a third-party tool, by default, MIMB does not store the dimensional role of each definition, such as fact or dimension.

You can create user-defined metadata extensions in the source and target definitions to define the dimensional role of each definition. When you export source and target definitions with these user-defined metadata extensions, MIMB converts the information in the metadata extensions to dimensional role information in third-party tools that work with dimensional metadata, such as IBM DB2 Cube Views.

Also, when you import metadata into PowerCenter from a third-party tool that works with dimensional metadata, MIMB converts the dimensional information into user-defined metadata extensions in PowerCenter.

The following table lists the metadata extension names and values that MIMB uses when you export and import dimensional metadata:

Metadata Extension Name	Possible Metadata Extension Values
DimensionalRoleType	UNDEFINED FACT DIMENSION OUTRIGGER BRIDGE HIERARCHY_NAVIGATION
DimensionalType	FIXED TYPE_1 TYPE_2 TYPE_3

Note: Not all third-party tools that work with dimensional metadata support all dimensional role and dimensional type values.

For example, you have the following relational target definitions with metadata extensions in PowerCenter:

Target Name	Metadata Extension Name	Metadata Extension Value
Customer	DimensionalRoleType	FACT
CustAddress	DimensionalRoleType	DIMENSION
CustPhone	DimensionalRoleType	DIMENSION

You export the target definitions to IBM DB2 Cube Views. You import the PowerCenter metadata into Cube Views. Cube Views imports the Customer table as a fact table, and the CustAddress and CustPhone tables as dimension tables.

Steps to Export Metadata

Use the following procedure to export metadata from PowerCenter to a file recognized by the target BI or data modeling tool.

To export metadata:

1. In the Repository Manager Navigator, select the object or objects you want to export, and click Repository > Export Metadata.

   The Metadata Export Wizard appears.

2. Choose the target tool you want to export the object to.

   Click More Info to read about the tool requirements.

3. Click Next.

   The Metadata Export Wizard displays options specific to the tool you select.
4. Enter the options specific for the tool to which you are exporting.

   Choose a path and file name for the target file, if needed.
5. Click Export.

   Click Show Details to view detailed messages about the metadata conversion. Click Save Log to save the message log to a text file.
6. Click Finish to close the wizard.

Steps to Import Metadata

You can import source definitions, target definitions, and mappings from a file created by another BI or data modeling tool.

1. In the Repository Manager, select the folder into which you want to import metadata, and click Repository > Import Metadata.

   The Metadata Import Wizard appears.
2. Choose the source tool to import the object.

   Click More Info for information about the tool and the usage.
3. Click Next.

   The Metadata Import Wizard displays options specific to the tool you select.
4. Enter the options specific for the tool from which you are importing.

   Choose a path and file name for the file that contains the metadata, if needed.
5. Click Next.

   The PowerCenter Options page of the wizard appears.
6. Enter the PowerCenter options.

   The following table describes the PowerCenter options you define in the Metadata Import Wizard:

Option	Description
Export Objects	Type of repository object or objects to create. You can create following definitions: source, target, or source, target, and mappings. Default is source.
Database Type	Database type for the source or target definitions the wizard creates. The wizard can define the object definition database type based on the metadata defined in the file, or you can override the database type by choosing a database type here. Default is auto detect.
Database Name	Database name under which you want to group the repository objects in the Navigator. If you do not specify a database name, the Metadata Import Wizard groups all objects based on the source database.

Option	Description
Codepage	Code page of the PowerCenter repository you import the metadata into. Default is MS1252.
Export Metadata Extensions	Export additional descriptions, comments or notes as PowerCenter Metadata Extensions. Default is True.
Path to the Informatica installation	Path to the Informatica PowerCenter client binary files. For example, set the PowerCenter client installation to: `C:\Informatica\PowerCenter <Version number>`. Ensure that the path contains **client** and **java** folders.

7. Click Next. The Metadata Import Wizard converts the metadata in the file to a format recognized by PowerCenter.

   The Metadata Import Wizard displays the import results.

   Click Show Details to view detailed messages about the metadata conversion. Click Save Log to save the message log to a text file.
8. Click Next.

   The Object Selection page of the wizard appears.
9. Select the objects to import into the repository, and click Finish.

   The Metadata Import Wizard adds the objects to the folder in the repository. If the folder contains objects with the same name as those you import, the Metadata Import Wizard opens the Conflict Resolution Wizard. Use the Conflict Resolution Wizard to resolve specific object conflicts.
10. Click Compare Conflict to compare conflicting objects in the import file and target repository.

    Resolve object conflicts as they appear in the Conflict Resolution Wizard.
11. Click Next to proceed through the Conflict Resolution Wizard.
12. Click Close when you resolve all conflicts.

    The Metadata Import Wizard imports all objects.

Exchanging Metadata with Business Objects Designer

You can exchange metadata with Business Objects Designer by exporting metadata from PowerCenter into Business Objects Designer or by importing metadata from Business Objects Designer into PowerCenter. You can exchange source and target definitions, including facts and dimensions, between PowerCenter and Business Objects Designer.

When you exchange metadata between PowerCenter and Business Objects Designer, PowerCenter uses MIMB to convert metadata to or from a Business Objects universe. A Business Objects universe is a mapping of the data structure of database tables, columns, and joins. For PowerCenter metadata, a universe is a representation of the metadata from PowerCenter source and target definitions. When you export metadata, you can choose to update an existing universe or create a new universe. Use a universe to build queries and generate and perform analysis in Business Objects.

Related Topics:

- "Steps to Export Metadata" on page 149
- "Steps to Import Metadata" on page 150

Metadata and Datatype Conversion

Metadata names and datatypes are converted between PowerCenter and Business Objects Designer when you exchange metadata. MIMB converts metadata types to classes and objects in Business Objects Designer.

If Business Objects Designer does not support a specific PowerCenter metadata name or column property, PowerCenter does not export the metadata. For example, a source definition in PowerCenter might use the HIDDEN property for a column, which Business Objects Designer does not support. PowerCenter does not export the column property to Business Objects Designer.

When you export metadata from PowerCenter to Business Objects Designer, MIMB converts PowerCenter metadata names and datatypes to the corresponding values in Business Objects Designer. Likewise, when you import metadata from Business Objects Designer, MIMB converts Business Objects Designer metadata object name and datatypes to the corresponding PowerCenter values. If PowerCenter supports a Business Objects datatype, MIMB does not convert the Business Objects datatype when it imports the metadata.

When you export source or target definitions that contain foreign keys, PowerCenter also exports the metadata referenced by the foreign keys to Business Objects Designer. You can define the types of joins PowerCenter performs when it exports metadata that includes foreign keys when you export the metadata in PowerCenter.

Business Objects Designer does not support all metadata names, column properties, and datatypes supported by PowerCenter.

The following table lists the PowerCenter metadata names and the corresponding Business Objects Designer metadata:

PowerCenter Metadata Name	Business Objects Designer Name
Powermart	Universe
Object Name (source or target)	Class Name, represented in Business Objects Designer by a folder icon
Attribute Name	Object Name, represented by an icon under the Class Name in Business Objects Designer
Business Name	Object Name
Attribute	Dimension
Fact	Measure

The following table lists the PowerCenter datatypes and the corresponding Business Objects Designer datatypes:

PowerCenter Datatype	Business Objects Designer Datatype
LONG	Numeric
NUMBER	Numeric
RAW	Blob
CHAR	Character
VARCHAR	Character
NCHAR	Character
DATE	Date

Exporting Metadata to Business Objects Designer

Export metadata from PowerCenter to Business Objects Designer using the Repository Manager. You must have Business Objects installed and the Metadata Exchange option license key to export metadata to Business Objects Designer. When you export metadata from PowerCenter, PowerCenter uses MIMB to export the metadata to a Business Objects universe, and then opens Business Objects Designer with the exported universe.

Before you export metadata from PowerCenter, you must create a connection in Business Objects Designer to the database you want to export metadata from. For more information about creating a connection, see the documentation for Business Objects Designer.

The following table lists the export options in the Metadata Export Wizard for Business Objects Designer:

Option	Description
Connection Name	Connection to the database from which you want to export metadata. You must define this connection name in Business Objects Designer before you export metadata.
Login User	Login name for a repository installation of Business Objects Designer. Business Objects Designer prompts for a user and password when you export metadata to repository installation of Business Objects Designer. Leave this field blank if you use a standalone version of Business Objects Designer or you want to manually enter the user name and password.
Login Password	Password for a repository installation of Business Objects Designer.
Login Offline	Login offline to a local universe. You can store a Business Objects universe locally or in a repository. You can select the following options: - True. The repository is stored locally. - False. The universe is stored locally or in a repository.
Login Repository Name	Name of the repository that contains the universe.
Close Designer	Closes Business Objects Designer after importing the universe from PowerCenter. Use this option to stop Business Objects Designer when it runs on a remote location.

Option	Description
Schema Export Algorithm	Updates the tables and joins in the exported universe.
Allow Outer Joins	Type of joins performed by PowerCenter when exporting metadata with tables that have foreign key columns. You can select the following options: - True. Use the foreign key relationship defined for the column. - False. Only allow inner joins when exporting tables referenced by a foreign key.
Fact Table	Name of the table to be treated as a fact table by Business Object Designer. Use this option to identify a source or target definition as a fact table when exporting metadata.
Assume Tables are Dimensions	Exports metadata as a dimension. You can select the following options: - True. Exports metadata as a dimension. - False. Does not export metadata as a dimension.
Dimensions Export Algorithm	Exports and updates dimensions in an existing Business Objects Designer universe. You can select the following options: - Don't export dimensions - Replace dimensions - Create new dimensions - Update dimension's description
Export Hierarchies	Exports the OLAP hierarchies in the exported source or targets definitions.
Naming Conventions	Exports the class and object names in the universe. You can leave the names as defined in PowerCenter or change them to all uppercase or all lowercase.

Troubleshooting Exchanging Metadata

Enable Retain Physical Properties when you import PowerCenter metadata from a third-party tool that does not use all column properties.

When you import metadata into PowerCenter from a tool that does not use all column properties, the Metadata Import Wizard uses default values for the column properties. However, you can retain the column properties if a source or target of the same name exists in the folder. To retain the column properties, enable the Retain Physical Properties attribute in the Metadata Import Wizard. You might want to do this if you export metadata to a tool that does not use all column properties, modify the metadata in the other tool, and then import the modified metadata back into PowerCenter.

Choose Enable for the Reverse Engineer BI Properties option when you import metadata from Business Objects Designer that originated in PowerCenter.

When you export metadata to Business Objects Designer and then import that metadata into PowerCenter, you lose business name information if you choose Disabled. However, when you choose one of the Enable values for this property, MIMB preserves the business name information for the metadata.

CHAPTER 13

Copying Objects

This chapter includes the following topics:

Copying Objects Overview

The Workflow Manager, Designer, and Repository Manager provide a Copy Wizard that you use to copy repository objects. You can copy repository objects such as workflows, worklets, tasks, sessions, mappings, mapplets, sources, targets, and transformations. You can also copy segments of workflows or mappings.

You can copy objects within the same folder, to a different folder, or to a different repository. If you want to copy an object to another folder, you must first open the target folder.

Code Pages

To ensure no data is lost when you copy an object from one repository to another, you can copy objects between repositories with the PowerCenter Client. This is done when the code page of the originating repository is identical to or a subset of the destination repository code page.

Copy Wizard

The Copy Wizard checks for conflicts in the target folder and provides choices to resolve the conflicts. For example, if an item exists in the target folder, a description of the conflict appears in the Conflict Message section of the screen. The Copy Wizard displays possible resolutions in the Resolution area of the screen. For a duplicate object you can rename, reuse, replace, or skip copying the object.

The following table describes the areas of the Copy Wizard:

Area	Description
Copy From/Copy To	Displays the original repository and folder name and the target repository and folder name.
Overview Area	Displays the items to copy, existing conflicts, original instance name, target instance name, and action taken to resolve the conflict. It displays a red icon next to each object with a conflict, and a green icon next to each object without a conflict.
Conflict Message	Identifies the current conflict and the name of the object with the conflict, if any. After you choose a resolution, the message describes the resolution.
Resolution	Displays the elected resolution or a list of choices for resolution. Choices might be different, depending on the conflict.
Edit	You can edit the object name if you choose to rename the object.
Apply This Resolution to Other Conflicts	Applies the resolution to all unresolved conflicts or just the conflicts for the same object type.
Compare Conflict	Compares duplicate objects in the target folder to the objects you are copying.
Next Conflict/Option	You can choose additional options for session and workflow resolutions, such as applying default connections or retaining connections during the copy. Next Conflict/Option displays with session or workflow conflicts that you resolve by renaming or replacing the target.
View Dependency	Displays object dependencies for the current object.

You can configure display settings and functions of the Copy Wizard by clicking Tools > Options in the Designer or Workflow Manager.

Related Topics:

- "Comparing Repository Objects" on page 45
- "Copying Dependent Objects" on page 159

Resolving Copy Conflicts

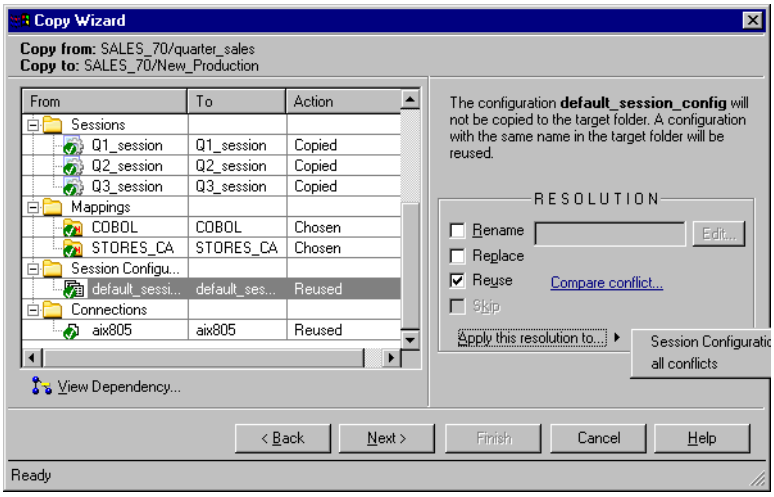
When the Copy Wizard encounters a conflict, it prompts you to resolve the conflict before continuing the copy process. The Copy Wizard provides you with the resolutions depending on the type of conflict.

The following table describes the Copy Wizard resolutions:

Resolution Name	Description	Availability
Copy	Copy a connection object.	When the Copy Wizard cannot find a connection object in the target repository.
Browse	Click Browse to choose a server, connection, or mapping. You must select a valid object in the target folder for the copy process to succeed.	When the Copy Wizard cannot find a server, connection, or mapping in the target folder it displays the Browse button.
Rename	Change the object name when copying it to the target folder.	When an object with the same name exists in the target folder.
Replace	Replace the existing object in the target folder.	When copying to another folder in the same repository or another folder in a different repository and an object with the same name exists in the target folder.
Reuse	Use the existing object in the target folder.	When a reusable object exists in the target folder.
Skip	Skips copying the object.	When an object with the same name exists in the target folder.

If the target folder has duplicate objects, you can compare them to the objects you are copying to determine the differences. Click the Compare Conflict link in the Copy Wizard to display source and target views of the objects.

The following figure shows the conflict when you try to copy an object to a folder that contains an object of the same name:



In the figure, the selected resolution reuses the object.

The wizard prompts you to select a resolution for each unresolved object in the copy. Optionally, you can apply the resolution to all unresolved objects of the same type, or to all conflicts in the copy. To apply the resolution to more objects, click Apply This Resolution To and choose either All Conflicts or conflicts for just the specified object type.

Related Topics:

- "Comparing Repository Objects" on page 45

Steps to Copy Objects

Use the following procedure to copy an object using the Copy Wizard. To cancel the copy operation, click the Cancel button or press the Esc key.

To copy an object using the Copy Wizard:

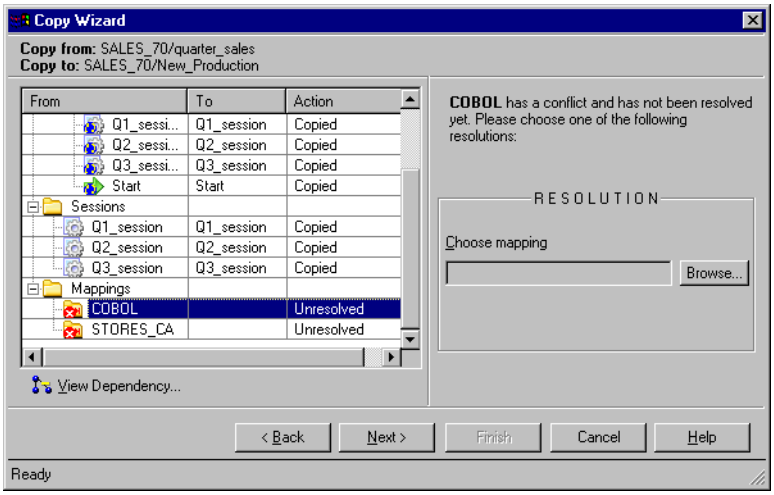
1. Open the target folder.
2. In the Navigator, select the object you want to copy.
3. Drag or copy the object into the target folder.
4. Click Yes in the Copy Confirmation dialog box.

   The Copy Wizard appears. The Copy Wizard displays objects by type. For example, the sessions display under the Sessions node, and mappings display under the Mappings node.

   The Copy Wizard displays a red icon on objects with conflicts. It displays conflicts one object type at a time.
5. Click Next if you do not encounter a conflict.

   If you encounter a conflict, choose a resolution from the Resolution options.

   The following figure shows the first of two unresolved mapping conflicts to resolve and the resolution option requires you to browse for an appropriate mapping:



6. Click Next to view the next conflict.

   If you work with session or workflow conflicts, you can click Next Conflict/Option to configure additional options for sessions or workflows with conflicts. For example, use Next Conflict/Option if you want to apply default connections in the target during the copy. Next Conflict/Option appears when you have session and workflow conflicts, and you choose to Rename or Replace the target objects.
7. Repeat steps 5 to 6 until you resolve all conflicts.

   The Copy Summary information appears.
8. Click Finish to complete the copy process.

Copying Dependent Objects

When you copy an object, the Copy Wizard copies all dependent objects. While you set up a copy, you might want to view the objects or instances that depend on the object you are copying. For example, if you are going to copy a session and an associated connection object in a workflow, you can select the connection object in the Copy Wizard and see which sessions in the workflow use the connection.

The Dependency dialog box displays the objects that use a selected object. The objects display upward in a hierarchy. For example, if you view the object dependencies of a connection object when you copy a workflow, the Dependency dialog box shows the session that uses the source object and the workflow that uses the session.

If there are no object dependencies, the View Object Dependencies dialog box displays the following message:

```
<<No dependencies found for this object.>>
```

To view dependencies for an object:

1. Select the object from the Overview area of the Copy Wizard.
2. Click the View Object Dependencies button.

   The Dependency dialog box appears.

Copying Workflow Manager Objects

In the Workflow Manager, you can copy workflows, worklets, workflow segments, and sessions using the Copy Wizard. You can also use the Copy Wizard to copy segments of a workflow. To copy these objects, you must resolve all conflicts that occur in the target folder.

Note: The Workflow Manager provides an Import Wizard in which you can import objects from an XML file. The Import Wizard also provides options to resolve conflicts.

Copying Workflows and Worklets

When you copy a workflow or a worklet, the Copy Wizard copies all of the worklets, sessions, and tasks included in the original workflow or worklet.

When you copy a workflow or worklet, you might encounter the following copy conflicts:

- **Duplicate name**. A workflow, worklet, or workflow component with the same name exists in the target folder.
- **Cannot find connection.** Connections from the original workflow do not exist for the target. If a connection object does not exist, you can select a connection object or skip the conflict and choose a connection object after you copy the workflow. You cannot copy connection objects.
- **Session conflicts.** When you copy a workflow, the Copy Wizard checks session components for possible conflicts. This includes the associated mapping and the database connection.

Copying Sessions

When you copy a Session task, the Copy Wizard looks for the database connections and associated mappings in the target folder. If the mapping or connection does not exist, you can select a new mapping or

connection. If the target folder has no mappings, you must first copy a mapping to the target folder in the Designer before you can copy the session.

When you copy a session, you might encounter the following copy conflicts:

- **Duplicate name.** A session with the same name exists in the target folder. You can rename the existing session, reuse the session in the target folder or replace it. If you rename or replace the session use the default mappings and connections. Otherwise, you may need to choose the connections and mappings after you copy the session.
- **Cannot find connection.** The connection object for this session does not exist in the target.
- **Cannot find mapping.** The associated mapping is not in the target folder. You can select an available mapping in the target folder. If you have no mappings in the target, you must cancel the session copy.
- **Cannot find database connections.** A database connection object does not exist in the target repository. Select connections from the target repository.

Mapping Conflicts

When you copy a session, the Copy Wizard verifies that the associated mapping exists in the target folder. If the mapping does not exist, you can choose a different mapping from the target folder.

To find available mappings in the target folder, click Browse. If the target folder does not have any mapping in it, the Copy Wizard prompts you to create one:

```
There are no mappings in this folder. Please create a mapping in the Mapping Designer.
```

You must cancel the copy process to create a mapping. When you cancel, the Copy Wizard does not copy any object. To avoid this problem you can copy the mapping to the target folder in the Designer before copying the session. If you replace a mapping with an invalid mapping, the associated sessions become invalid.

Database Connection Conflicts

When you copy a session to a different repository, the session uses the same database connection name and type as it has in the source folder. If a connection of the same name does not exist in the target, you can do one of the following:

- Select any connection of the same type in the target folder.
- Copy the connection to the target repository.
- Skip the connection conflict.

If you override the lookup or stored procedure database connection in the session properties, the Copy Wizard prompts you to either copy the connection information or choose another connection. Otherwise it uses the connection name in the session properties, even if the connection does not exist in the target folder. After you copy the session, you must verify that the lookup or stored procedure database connection exists in the target folder to validate the session.

Note: You cannot copy connection objects when you copy workflows.

Mapping Variables

When you copy a session that uses mapping variable values, the Copy Wizard either copies the variables to the target folder or retains the saved variable values in the folder.

The Workflow Manager copies the variable values to the target folder under the following conditions:

- **You copy a session into a folder to create a new session.** The new session contains a mapping that has an associated mapping variable from the copied session.

- **You copy a session into a folder to replace an existing session.** The replaced session in the target folder does not have saved variable values.

The Workflow Manager retains the saved variable values in the target folder if you replace a session that has saved variable values.

Copying Workflow Segments

You can copy segments of workflows and worklets when you want to reuse a portion of workflow logic. A segment consists of one or more tasks, the links between the tasks, and any condition in the links. You can copy reusable and non-reusable objects in segments. You can copy segments into workflows and worklets within the same folder, within another folder, or within a folder in a different repository. You can also paste segments of workflows or worklets into an empty Workflow or Worklet Designer workspace.

Note: You can copy individual non-reusable tasks by selecting the task and following the instructions for copying segments.

When you copy a segment, you might encounter the following copy conflicts:

- **Duplicate name**. You paste a segment into another workflow or worklet containing a task instance with the same name as the one you are copying. For example, if you copy a segment from Workflow_A containing s_Session1 into Workflow_B containing a session named s_Session1.

  For reusable objects, you can resolve this conflict by replacing the task instance or renaming the task instance with a unique name. If you replace the task instance, the Copy Wizard overwrites the task instance in the target workspace. When you overwrite the segment, the conditions in the copied link overwrite the link conditions in the target workflow or worklet. If you copy and paste a task within the same workflow or worklet, you cannot overwrite the original task with the copied task. The Copy Wizard creates a copy of the task and assigns it a new name. To avoid overwriting an object instance in a target workflow or worklet, choose to create a copy of the instance instead of replace it. Each time the Copy Wizard locates a duplicate instance in the target workflow or worklet, it creates a new copy of the object you are pasting and renames it so that it does not overwrite any objects.
- **Cannot find mapping.** You paste a segment to another folder without the associated mappings in the target folder. You must select a new mapping. If the target folder does not contain a mapping, you must copy a mapping to the target folder before you can copy the segment.
- **Cannot find database connection**. You paste a segment to another folder, but the target folder does not contain the associated database connection. If you copy to a folder in a different repository, and a connection of the same name does not exist, select any connection of the same type in the target folder, copy the connection to the target repository, or skip the connection conflict.
- **Segment contains user-defined workflow variable.** If you paste a segment containing a user-defined workflow or worklet variable, expressions using the variable become invalid. User-defined workflow variables are valid in the workflow or worklet in which they were created.

To copy a segment from a workflow or worklet:

1. Open the workflow or worklet.
2. Select a segment by highlighting each task you want to copy. You can select multiple reusable or non-reusable objects. You can also select segments by dragging the pointer in a rectangle around objects in the workspace.
3. Copy the segment to the clipboard.
4. Open the workflow or worklet into which you want to paste the segment. You can also copy the object into the Workflow or Worklet Designer workspace.
5. Click Edit > Paste or press Ctrl+V.

   The Copy Wizard opens and notifies you if it finds copy conflicts.

Copying Designer Objects

You can copy Designer objects within the same folder, to a different folder, or to a different repository. You can copy any of the Designer objects such as sources, targets, mappings, mapplets, transformations, and dimensions. You must open the target folder before you can copy to it. To copy these objects, you must resolve all conflicts that occur in the target folder.

When you copy Designer objects, you might have the following copy conflicts or options:

- **Duplicate item name.** When you copy objects you might have duplicate objects in the target folder. When you copy a mapping or mapplet, the wizard attempts to copy all the components of the mapping to the target. You might have some duplicate components in the target folder. You can resolve these conflicts individually, or you select resolutions all at once.
- **Copy a source included in a primary key-foreign key relationship that is not included in the mapping.** When you copy a mapping with a source object that has a primary key-foreign key relationship with another object not included in the mapping, the Copy Wizard asks you if you want to copy the referenced object.
- **Retain current values in reusable Sequence Generator or Normalizer transformations.** If you copy Sequence Generator transformations, select the Sequence Generator and Normalizer Current Value to retain the current value of the sequence number. When copying a Normalizer transformation, select this option to retain the current value of the generated key sequence. This option appears when you copy Sequence Generator or Normalizer transformations.
- **Copy SAP Program information.** If you copy an SAP mapping, you can choose to copy the associated installed ABAP program. Choose the Copy SAP Program Information check box. This option appears when you copy an SAP mapping across repositories.

Copying Mapping and Mapplets Segments

You can copy segments of mappings and mapplets when you want to reuse a portion of the mapping logic. A segment consists of one or more objects in a mapping or mapplet. A segment can include a source, target, transformation, mapplet, or shortcut. To copy mapping segments, select and copy the segments from the Mapping Designer and paste them into a target mapping or an empty mapping or mapplet workspace. You can copy segments across folders or repositories.

To copy a segment of a mapping or mapplet:

1. Open a mapping or mapplet.
2. Select a segment by highlighting each object you want to copy. You can select multiple objects. You can also select segments by dragging the pointer in a rectangle around objects in the workplace.
3. Copy the segment to the clipboard by pressing Ctrl+C or clicking Edit > Copy.
4. Open a target mapping or mapplet. You can also paste the segment into an empty workspace.
5. Click Edit > Paste or press Ctrl+V.

   If you are creating duplicate objects in a folder, the Designer assigns a unique name to the new object.

CHAPTER 14

Metadata Extensions

This chapter includes the following topics:

Metadata Extensions Overview

PowerCenter allows end users and partners to extend the metadata stored in the repository by associating information with individual objects in the repository. For example, when you create a mapping, you can store the contact information with the mapping. You associate information with repository metadata using metadata extensions.

PowerCenter Client applications can contain the following types of metadata extensions:

- **Vendor-defined.** Third-party application vendors create vendor-defined metadata extensions. You can view and change the values of vendor-defined metadata extensions, but you cannot create, delete, or redefine them.
- **User-defined.** You create user-defined metadata extensions using PowerCenter. You can create, edit, delete, and view user-defined metadata extensions. You can also change the values of user-defined extensions.

All metadata extensions exist within a domain. You see the domains when you create, edit, or view metadata extensions. Vendor-defined metadata extensions exist within a particular vendor domain. If you use third-party applications or other Informatica products, you may see domains such as Ariba. You cannot edit vendor-defined domains or change the metadata extensions in them.

User-defined metadata extensions exist within the User Defined Metadata Domain. When you create metadata extensions for repository objects, you add them to this domain.

Both vendor and user-defined metadata extensions can exist for the following repository objects:

- Source definitions
- Target definitions
- Transformations
- Mappings

- Mapplets
- Sessions
- Tasks
- Workflows
- Worklets

Working with Metadata Extensions

You can create reusable or non-reusable metadata extensions. You associate reusable metadata extensions with *all* repository objects of a certain type. So, when you create a reusable extension for a mapping, it is available for all mappings. Vendor-defined metadata extensions are always reusable.

Non-reusable extensions are associated with a single repository object. Therefore, if you edit a target and create a non-reusable extension for it, that extension is available for the target you edit. It is not available for other targets.

You can promote a non-reusable metadata extension to reusable, but you cannot change a reusable metadata extension to non-reusable.

You can create, edit, and delete user-defined metadata extensions using the following tools:

- **Designer.** Create, edit, and delete non-reusable metadata extensions for sources, targets, transformations, mappings, and mapplets. You can also promote non-reusable metadata extensions to reusable extensions.
- **Workflow Manager.** Create, edit, and delete non-reusable metadata extensions for sessions, workflows, and worklets. You can also promote non-reusable metadata extensions to reusable extensions.
- **Repository Manager.** Create, edit, and delete reusable metadata extensions for all types of repository objects. If you want to create, edit, and delete metadata extensions for multiple objects at one time, use the Repository Manager.

Creating Reusable Metadata Extensions

You can create reusable metadata extensions for repository objects using the Repository Manager.

When you create a reusable metadata extension for any type of repository object, the metadata extension becomes part of the properties of that type of object. For example, you create a reusable metadata extension for source definitions called SourceCreator. When you create or edit any source definition in the Designer, the SourceCreator extension appears on the Metadata Extensions tab. Anyone who creates or edits a source can enter the name of the person that created the source into this field.

To create a reusable metadata extension:

1. In the Repository Manager, connect to the appropriate repository.
2. Choose Edit > Metadata Extensions.

   The Edit Metadata Extensions dialog box opens.

This dialog box lists the existing user-defined and vendor-defined metadata extensions. User-defined metadata extensions appear in the User Defined Metadata Domain. If vendor-defined metadata extensions exist, they appear in their own domains.

3. Open the User Defined Metadata Domain.
4. Click Add.

   The Add Metadata Extensions dialog box opens.
5. Enter the metadata extension information.

   The following table describes the options available in the Add Metadata Extension dialog box:

Field	Description
Extension Name	Name of the metadata extension. Metadata extension names must be unique for each type of object in a domain. Metadata extension names cannot contain any special character except underscore, and they cannot begin with a number.
Object Type	Type of repository object to which the metadata extension is associated. This can be a source definition, target definition, transformation, mapping, mapplet, session, workflow, worklet, or all of these objects. You associate metadata extensions with specific types of transformations. For example, if you create a metadata extension for Expression transformations, it is available for Expression transformations.
Database Type	Database type. The database type is required for source and target definition objects. You can select a single database type or all database types. Required for source and target definition objects.
Datatype	Datatype: numeric (integer), string, or boolean.
Default Value	An optional default value. For a numeric metadata extension, the value must be an integer between -2,147,483,647 and 2,147,483,647. For a boolean metadata extension, choose true or false. For a string metadata extension, you can enter a default value of more than one line, up to 2,147,483,647 bytes.
Maximum Length	Maximum length for string metadata extensions. Required for string objects.
Client Visible	Specifies whether the metadata extension is visible in PowerCenter.
Client Editable	Specifies whether the value of the metadata extension is editable in PowerCenter. If you select this option, the Repository Manager grants Client Visible permission as well.
Share Read	Specifies whether the metadata extension is visible in vendor domains.
Share Write	Specifies whether the value of the metadata extension is editable across vendor domains. If you enable Share Write permission, the Repository Manager grants Share Read permission as well.

Field	Description
Private	Specifies whether the metadata extension is private to the domain in which it is created. The Repository Manager enables this option when third-party application vendors create vendor-defined metadata extensions.
Description	Optional description of the metadata extension.

6. Click Create.

Editing Reusable Metadata Extensions

You can edit user-defined, reusable metadata extensions for repository objects using the Repository Manager. When you edit a reusable metadata extension, you change the properties of the metadata extension. To change the value of a metadata extension, edit the repository object using the Designer or Workflow Manager.

Note: You cannot edit vendor-defined metadata extensions.

To edit a reusable metadata extension, select the appropriate metadata extension in the Metadata Extensions dialog box, and then click Edit.

You can modify the following fields:

- Default Value
- Permissions
- Description

Deleting Reusable Metadata Extensions

You can delete user-defined, reusable metadata extensions for repository objects using the Repository Manager. When you delete a reusable metadata extension for a repository object, you remove the metadata extension and its values from the properties of all objects of that type.

Note: You cannot delete vendor-defined metadata extensions.

To delete a reusable metadata extension, select the appropriate extension in the Metadata Extensions dialog box and click Delete.

APPENDIX A

MX Views Reference

This appendix includes the following topics:

MX Views Overview

PowerCenter Metadata Exchange (MX) provides a set of relational views that allow easy SQL access to the PowerCenter metadata repository. The Repository Manager generates these views when you create or upgrade a repository.

Warning: The PowerCenter repository tables have an open architecture. Although you can view the repository tables, Informatica strongly advises against altering the tables or data within the tables. Informatica is not responsible for corrupted data that is caused by customer alteration of the repository tables or data within those tables. Therefore, do not directly access the actual repository tables. Instead, use MX to access the repository.

MX View Categories

MX views provide information to help you analyze metadata stored in the repository.

The following table lists the available MX views by category:

Category	Description
Database	Provides a list of database definitions in the repository.
Sources	Provides a list of source definitions by folder.
Targets	Provides a list of target definitions by folder.
Mappings and Mapplets	Provides a list of sources, targets, and transformations used in mappings and mapplets by folder.
Metadata Extensions	Provides details of metadata extensions defined for objects.
Transformations	Provides details of transformation instances by folder.
Workflows, Worklets, and Tasks	Provides static and run time details for workflows and worklets by folder.
Security	Provides user information.
Deployment	Provides deployment details such as deployment groups and objects that were deployed from one repository to another.
Repository	Provides repository details such as repository name and connection information.
Integration Service	Provides details such as server name and host name.
Change Management	Provide version history of object and label details.
Folders	Provides details such as folder name and description.

For example, if a source table changes, and you need to re-import the source definition into the repository, you could use the REP_SRC_MAPPING view to see how many mappings include this source. Likewise, if you want to view source and target dependencies for a mapping, you could use REP_TBL_MAPPING.

Almost all views support access to comment information. You can add comments to any object within PowerCenter through the Designer and Workflow Manager. You can access comments about individual tables, table relationships, data fields, and data transformations.

Use these views to create reports using third-party reporting tools, such as Crystal Reports.

MX facilitates the integration of decision support metadata between the PowerCenter repository and popular Decision Support System (DSS) tools, data modeling tools, and any other metadata resources. With MX, you can drill down to the operational metadata level and expose information needed to support decisions. MX also helps you make precise information requests that draw from data models, mappings, and transformation data. For IS professionals, the MX architecture provides the following benefits:

- Improves warehouse maintenance and management capability.
- Reduces time and resources required to support end-user requests.
- Expands the ability to provide information resources in a controlled manner.

Note: The Designer includes options to save MX data.

Using PowerCenter Repository Reports

You can browse and analyze PowerCenter metadata with PowerCenter Repository Reports.

PowerCenter Repository Reports prepackages a set of reports and dashboards, which can be easily customized to meet business needs. The prepackaged dashboards and reports enable you to analyze the following types of metadata stored in a PowerCenter repository:

- Source and target metadata
- Transformation metadata
- Mapping and mapplet metadata
- Workflow and worklet metadata
- Session metadata
- Change management metadata
- User metadata
- Operational metadata

SQL Definition of Views

PowerCenter provides two sets of SQL scripts: one to create the MX views and one to drop MX views.

Creating MX Views

Each time you create or upgrade a repository, the Repository Service runs SQL scripts that creates the MX views.

The following table lists the SQL scripts to create MX views:

Repository Database	SQL Script
IBM DB2	db2mxbld.sq_ and db2mxbl2.sq_
Microsoft SQL Server	sqlmxbld.sq_ and sqlmxbl2.sq_
Oracle	oramxbld.sq_ and oramxbl2.sq_
Sybase	sybmxbld.sq_ and sybmxbl2.sq_

These SQL scripts are stored in the Repository Service installation directory.

Dropping MX Views

If you delete a repository, the Repository Service runs SQL scripts that drops the MX views. You can run the scripts from the Designer.

The following table lists the SQL scripts to drop MX views:

Repository Database	SQL Script
IBM DB2	db2mxdrp.sq_ and db2mxdr2.sq_
Microsoft SQL Server	sqlmxdrp.sq_ and sqlmxdr2.sq_
Oracle	oramxdrp.sq_ and oramxdr2.sq_
Sybase	sybmxdrp.sq_ and sybmxdr2.sq_

These SQL scripts are stored in the Repository Service installation directory.

Integrating MX Views with Third-Party Software

With MX software and support from Informatica, vendors of popular query and reporting tools can quickly create a metadata link between their products and the PowerCenter repository.

Software vendors can integrate PowerCenter metadata with their products through different methods, from pulling the PowerCenter metadata into product or user repositories to providing dynamic desktop pass-through access.

The next generation of MX, called Metadata Exchange SDK, provides an object-based application programming interface (API) to read and write metadata in Informatica repositories.

Database Definition View

The database definition view provides a list of all database definitions in the repository. A database definition includes the source database names, flat file or RDBMS, and the folder where the database definition resides.

MX provides the REP_DATABASE_DEFS view to help you analyze database definitions.

REP_DATABASE_DEFS

The following table lists database definition details:

Column Name	Datatype	Description
DATABASE_NAME	VARCHAR2 (240)	Database definition name.
DEF_SOURCE	VARCHAR2 (240)	Source of the definition.
SUBJECT_AREA	VARCHAR2 (240)	Folder name.
VERSION_ID	INTEGER	Version ID of the source.
DATABASE_VERSION_NUMBER	NUMBER	Version number of the database.

Source Views

Source views provide a list of the latest version of all source definitions defined by folder of any PowerCenter repository. Source definitions can be defined for both relational and non-relational sources. These views also show source properties such as shortcuts, creation date, version, description, and business name. They also provide information such as source columns, column properties, source metadata extensions, and mappings and mapplets where these sources are used.

The following table lists the different views that help you analyze source metadata:

View	Description
REP_ALL_SOURCES	This view provides a list of the latest version of sources defined in each folder of a repository.
REP_ALL_SOURCE_FLDS	This view provides all the fields and field properties for all sources defined in REP_ALL_SOURCES MX View.
REP_SRC_FILES	This view provides a list of all file definitions in the repository.
REP_SRC_TBLS	This view provides a list of relational database table sources that have been analyzed through the Source Analyzer tool or imported from a DDL (Data Definition Language) file.
REP_SRC_FILE_FLDS REP_SEG_FLDS	These views provide access to the fields in a non-relational source.
REP_SRC_TBL_FLDS	This view provides access to the fields in relational sources. Use the source name to retrieve all the fields belonging to the source.

REP_ALL_SOURCES

This view provides a list of the latest version of sources defined in each folder of a repository. Sources include both relational sources and non-relational sources such as XML files and flat files.

The following table lists source information in the REP_ALL_SOURCES view:

Column Name	Datatype	Description
PARENT_SUBJECT_AREA	VARCHAR2 (240)	Folder name.
PARENT_SUBJECT_ID*	NUMBER	Folder ID.
PARENT_SOURCE_NAME	VARCHAR2 (240)	Name of the parent source.
PARENT_SOURCE_BUSINESS_NAME	VARCHAR2 (240)	Business name of the parent source.
PARENT_SOURCE_ID*	NUMBER	ID of the parent source.
PARENT_SOURCE_DESCRIPTION	VARCHAR2 (2000)	Description of the parent source.
PARENT_SOURCE_VERSION_NUMBER*	NUMBER	Source version number.
PARENT_SOURCE_VERSION_STATUS	NUMBER	Parent source version status.

Column Name	Datatype	Description
PARENT_SOURCE_UTC_CHECKIN	NUMBER	UTC time (Coordinated Universal Time) when the parent source was checked in.
PARENT_SOURCE_UTC_LAST_SAVED	NUMBER	UTC time when the parent source was last modified.
PARENT_SOURCE_LAST_SAVED	VARCHAR2 (30)	Time when the parent source was last saved.
PARENT_SOURCE_SCHEMA_NAME	VARCHAR2 (240)	Name of the source schema.
PARENT_SOURCE_FIRST_FIELD_ID*	NUMBER	ID of the first field in the source.
PARENT_SOURCE_SELECT_INFO_ID*	NUMBER	File organization information.
PARENT_SOURCE_DISPLAY_SIZE	NUMBER	Parent source display size (uncompressed binary).
PARENT_SOURCE_PHYSICAL_SIZE	NUMBER	Parent source physical size (compressed binary).
PARENT_SRC_MIN_PHYSICAL_SIZE	NUMBER	Physical size (compressed binary).
PARENT_SOURCE_DATABASE_NAME	VARCHAR2 (240)	Database name of the parent source.
PARENT_SOURCE_TYPE	NUMBER	Specifies whether the source is a relational or a non-relational source.
PARENT_SOURCE_DATABASE_TYPE	VARCHAR2 (50)	Name of the database type of the parent source.
SUBJECT_AREA	VARCHAR2 (240)	Folder name.
SUBJECT_ID*	NUMBER	Folder ID.
SOURCE_NAME	VARCHAR2 (240)	Source name.
SOURCE_ID*	NUMBER	Source ID.
SOURCE_DESCRIPTION	VARCHAR2 (2000)	Source description.
SOURCE_VERSION_NUMBER*	NUMBER	Source version number.
SOURCE_VERSION_STATUS	NUMBER	Specifies whether the source version is active or has been deleted.
SOURCE_UTC_CHECKIN	NUMBER	UTC time for source checkin.
SOURCE_UTC_LAST_SAVED	NUMBER	UTC time when the source display was last saved.
SOURCE_LAST_SAVED	VARCHAR2 (30)	Time when the source was last saved.
SOURCE_DATABASE_NAME	VARCHAR2 (240)	Source database name.
REPOSITORY_NAME	VARCHAR2 (240)	The repository name.
IS_SHORTCUT	NUMBER	Specifies whether the source is a shortcut. 1 = shortcut; 0 = not a shortcut.

Column Name	Datatype	Description
IS_GLOBAL_SHORTCUT	NUMBER	Specifies whether the source is a global shortcut. 1 = shortcut; 0 = not a shortcut.
**Indicates that the column is a key column.*		

REP_ALL_SOURCE_FLDS

This view provides all the fields and field properties for all sources defined in REP_ALL_SOURCES MX View. For global shortcuts, the name of the shortcut appears. For local shortcuts, the names of shortcut and the parent targets display.

The following table lists source field information in the REP_ALL_SOURCE_FLDS view:

Column Name	Datatype	Description
PARENT_SUBJECT_AREA	VARCHAR2 (240)	Parent folder name.
PARENT_SUBJECT_ID*	NUMBER	Parent folder ID.
PARENT_SOURCE_NAME	VARCHAR2 (240)	Parent source name.
PARENT_SOURCE_BUSINESS_NAME	VARCHAR2 (240)	Business name of the parent source.
PARENT_SOURCE_ID*	NUMBER	Parent source ID.
PARENT_SOURCE_DESCRIPTION	VARCHAR2 (2000)	Description of the parent source.
PARENT_SOURCE_VERSION_NUMBER*	NUMBER	Version number of the parent source.
PARENT_SOURCE_VERSION_STATUS	NUMBER	Status of the parent source version.
PARENT_SOURCE_UTC_CHECKIN	NUMBER	UTC time when the parent source was checked in.
PARENT_SOURCE_UTC_LAST_SAVED	NUMBER	UTC time when the parent source was last saved.
PARENT_SOURCE_LAST_SAVED	VARCHAR2 (30)	Time when the parent source was last saved.
PARENT_SOURCE_TYPE	NUMBER	Source type such as relational database or flat file.
PARENT_SOURCE_DATABASE_NAME	VARCHAR2 (240)	Database name of the parent source.
PARENT_SOURCE_DATABASE_TYPE	VARCHAR2 (50)	Database type of the parent source.
SUBJECT_AREA	VARCHAR2 (240)	Folder name.
SUBJECT_ID*	NUMBER	Folder ID.
SOURCE_NAME	VARCHAR2 (240)	Source name.
SOURCE_ID*	NUMBER	Source ID.
SOURCE_DESCRIPTION	VARCHAR2 (2000)	Source description.

Column Name	Datatype	Description
SOURCE_VERSION_NUMBER*	NUMBER	Source version number.
SOURCE_VERSION_STATUS	NUMBER	Specifies whether the source version is active or has been deleted.
SOURCE_UTC_CHECKIN	NUMBER	UTC time when the source was last checked in.
SOURCT_UTC_LAST_SAVED	NUMBER	UTC time when the source was last saved.
SOURCE_LAST_SAVED	VARCHAR2 (30)	Time when the source was last saved.
SOURCE_DATABASE_NAME	VARCHAR2 (240)	Name of the database for the source.
SOURCE_FIELD_NAME	VARCHAR2 (240)	Source field name.
SOURCE_FIELD_BUSINESS_NAME	VARCHAR2 (240)	Business name of the source field.
SOURCE_FIELD_ID*	NUMBER	ID of the source field (primary key).
SOURCE_FIELD_DESCRIPTION	VARCHAR2 (2000)	Description of the source field.
SOURCE_FIELD_NUMBER	NUMBER	Source field number.
SOURCE_FIELD_NEXT_FIELD_ID*	NUMBER	ID of the field that follows the current field.
SOURCE_FIELD_LEVEL	NUMBER	Field level number for non-relational sources.
SOURCE_FIELD_PICTURE_TEXT	VARCHAR2 (240)	Picture text that a COBOL source uses. Null for relational sources.
SOURCE_FIELD_OCCURS_TIME	NUMBER	Number of time that the field (or record) occurs in the source.
SOURCE_FIELD_REDEFINES_FIELD	VARCHAR2 (240)	Identifies the field/record that this field/record redefines.
SOURCE_FIELD_DISPLAY_OFFSET	NUMBER	Offset of this field within the source.
SOURCE_FIELD_DISPLAY_LENGTH	NUMBER	Display field length.
SOURCE_FIELD_PHYSICAL_OFFSET	NUMBER	Offset of this field within this FD.
SOURCE_FIELD_PHYSICAL_LENGTH	NUMBER	Physical field length.
SOURCE_FIELD_CHILD_FIELD_ID*	NUMBER	The next child, if any, for a non-relational COBOL source.
SOURCE_FIELD_KEY_TYPE	VARCHAR2 (50)	Specifies whether the source field key is a primary key or a foreign key.
SOURCE_FIELD_DATATYPE	VARCHAR2 (40)	Field datatype.
SOURCE_FIELD_PRECISION	NUMBER	Length or precision for the field.

Column Name	Datatype	Description
SOURCE_FIELD_SCALE	NUMBER	Scale for the field.
SOURCE_FIELD_PIC_USAGE_NAME	NUMBER	Source field picture usage name.
SOURCE_FIELD_NULLTYPE	NUMBER	Specifies whether nulls are allowed. 0= nulls allowed; 1 = nulls not allowed.
REPOSITORY_NAME	VARCHAR2 (240)	Repository name.
IS_SHORTCUT	NUMBER	Specifies whether the source is a shortcut. 1 = shortcut; 0 = not a shortcut.
IS_GLOBAL_SHORTCUT	NUMBER	Specifies whether the source is a global shortcut. 1 = shortcut; 0 = not a shortcut.
**Indicates that the column is a key column.*		

REP_SRC_FILES

This view provides a list of all file definitions in the repository. Use FIRST_FIELD_ID to retrieve the fields belonging to a non-relational source by following the links in the REP_SRC_FILE_FLDS view. Any flat file imported through the Source Analyzer has an entry.

The following table lists file information in the REP_SRC_FILES view:

Column Name	Datatype	Description
FILE_ID*	INTEGER	Source ID (primary key).
SUBJECT_AREA*	VARCHAR2 (240)	Folder name.
DATABASE_TYPE	VARCHAR2 (240)	Type of database extracted from.
DATABASE_NAME	VARCHAR2 (240)	Name of database extracted from (DSN).
FILE_NAME	VARCHAR2 (240)	Name of file definitions.
SCHEMA_FILE_NAME	VARCHAR2 (240)	File from which schema was extracted.
SELECT_INFO_ID	INTEGER	File organization information.
DISPLAY_SIZE	INTEGER	Display size (uncompressed).
PHYSICAL_SIZE	INTEGER	Physical size (compressed binary).
MIN_PHYSICAL_SIZE	INTEGER	Minimum physical size (varying records).
FIRST_FIELD_ID	INTEGER	Link to first field of file definitions.
SOURCE_DESCRIPTION	VARCHAR2 (2000)	Source description.

Column Name	Datatype	Description
VERSION_ID	INTEGER	Version ID.
SOURCE_VERSION_NUMBER	NUMBER	Source version number.
**Indicates that the column is a key column.*		

REP_SRC_TBLS

This view provides a list of relational database table sources that have been analyzed through the Source Analyzer tool or imported from a DDL (Data Definition Language) file.

The following table lists relational database table information in the REP_SRC_TBLS view:

Column Name	Datatype	Description
TABLE_NAME	VARCHAR2 (240)	Table name.
TABLE_BUSNAME	VARCHAR2 (240)	Business name of the table.
TABLE_ID	NUMBER	Unique key.
SUBJECT_AREA	VARCHAR2 (240)	Folder name.
DATABASE_TYPE	VARCHAR2 (240)	Type of database extracted from.
DATABASE_NAME	VARCHAR2 (240)	Name of database extracted from.
SCHEMA_NAME	VARCHAR2 (240)	Name of schema extracted from.
FIRST_FIELD_ID	NUMBER	Link to first field.
SOURCE_DESCRIPTION	VARCHAR2 (2000)	Source description.
VERSION_ID	INTEGER	Folder version ID.
VERSION_NAME	VARCHAR2 (240)	Folder version name.
LAST_SAVED	VARCHAR2 (30)	Last time the source table was saved.
SOURCE_VERSION_NUMBER	NUMBER	Source version number.
SUBJECT_ID	NUMBER	Folder ID.

REP_SRC_FILE_FLDS and REP_SEG_FLDS

These views provide access to the fields in a non-relational source. Each field is contained in the scanned tables listed in the REP_SEG_FLDS view.

The following table lists source field information in the REP_SRC_FILE_FLDS and REP_SEG_FLDS views:

Column Name	Datatype	Description
FIELD_ID	INTEGER	Field ID (primary key)
SUBJECT_AREA	VARCHAR2 (240)	Folder name.
FILE_ID	INTEGER	Source ID (primary key).
FIELD_NAME	VARCHAR2 (240)	Field name.
FIELD_LEVEL	NUMBER	Field level (for example, 01 and 02).
FIELD_NUMBER	NUMBER	Order number of the field.
FIELD_DESCRIPTION	VARCHAR2 (2000)	Comments for this field.
PICTURE_TEXT	VARCHAR2 (240)	PIC clause.
OCCURS	NUMBER	Number of OCCURS.
REDEFINES_FIELD	VARCHAR2 (240)	Redefines this field.
KEY_TYPE	NUMBER	Key type. 1 = primary key; 0 = not a key.
DISPLAY_OFFSET	NUMBER	Offset using display length.
DISPLAY_LENGTH	NUMBER	Display length.
PHYSICAL_OFFSET	NUMBER	Physical offset.
PHYSICAL_LENGTH	NUMBER	Physical length.
USAGE_TYPE	VARCHAR2 (240)	COMP type (binary compressed fields).
DATA_PRECISION	NUMBER	Decimal precision for numeric fields or field length for CHAR fields.
DATA_SCALE	NUMBER	Decimal scale for numeric fields.
CHILD_ID	NUMBER	Link to child field if this is a group item.
SIBLING_ID	NUMBER	Link to next field at this level.
VERSION_ID	NUMBER	Link to next field at this level.
SRC_VERSION_NUMBER	NUMBER	Version number of the source.

REP_SRC_TBL_FLDS

This view provides access to the fields in relational sources. Use the source name to retrieve all the fields belonging to the source. The columns in this view are part of the tables listed in the REP_SRC_TBLS views.

The following table lists relational source fields in the REP_SRC_TBL_FLDS view:

Column Name	Datatype	Description
COLUMN_NAME	VARCHAR2 (240)	Field name.
COLUMN_BUSNAME	VARCHAR2 (240)	Business name of the field.
COLUMN_ID*	INTEGER	Field ID (primary key).
SUBJECT_AREA*	VARCHAR2 (240)	Folder name.
TABLE_ID	INTEGER	Source table ID.
TABLE_NAME	VARCHAR2 (240)	Table name.
TABLE_BUSNAME	VARCHAR2 (240)	Business name of the table.
COLUMN_NUMBER	INTEGER	Order number of the column.
COLUMN_DESCRIPTION	VARCHAR2 (2000)	Description of the column.
KEY_TYPE	VARCHAR2 (50)	Key type for this column.
SOURCE_TYPE	VARCHAR2 (40)	Data type for this column.
DATA_PRECISION	INTEGER	Decimal precision for numeric fields or field length for CHAR fields.
DATA_SCALE	INTEGER	Decimal scale for numeric fields.
NEXT_COLUMN_ID	INTEGER	Link to next field in source table.
VERSION_ID	INTEGER	Folder version ID.
VERSION_NAME	VARCHAR2 (240)	Folder version name.
SOURCE_VERSION_NUMBER	NUMBER	Source version number.
**Indicates that the column is a key column.*		

Target Views

Target views provide a list of the latest version of all target definitions defined by folder of a PowerCenter repository for both relational and non-relational sources. These views also show target properties such as shortcuts, creation date, version, description, and business name. They also provide information such as target columns, column properties, target metadata extensions, and mappings and mapplets where these target are used.

The following table lists the different views that help you analyze target metadata:

View	Description
REP_ALL_TARGETS	This view provides a list of the latest version of all targets defined in each folder of a repository.
REP_ALL_TARGET_FLDS	This view provides all the fields and field properties for targets defined in REP_ALL_TARGETS view.
REP_TARG_TBLS	This view provides a list of targets in the repository. FIRST_COLUMN_ID is a link to the set of columns for this table.
REP_TARG_TBL_COLS	This view provides the properties of columns defined for the target tables in a data warehouse or data mart.

REP_ALL_TARGETS

This view provides a list of the latest version of all targets defined in each folder of a repository. Targets include both relational and non-relational targets such as XML files and flat files. For global shortcuts, the name of the shortcut appears. For local shortcuts, the names of shortcut and the parent targets display.

The following table lists target details in the REP_ALL_TARGETS view:

Column Name	Datatype	Description
PARENT_SUBJECT_AREA	VARCHAR2 (240)	Parent folder name.
PARENT_SUBJECT_ID*	NUMBER	Folder ID.
PARENT_TARGET_NAME	VARCHAR2 (240)	Target name.
PARENT_TARGET_BUSINESS_NAME	VARCHAR2 (240)	Business name for the target.
PARENT_TARGET_ID*	NUMBER	Target ID (primary key).
PARENT_TARGET_DESCRIPTION	VARCHAR2 (2000)	Target description.
PARENT_TARGET_VERSION_NUMBER*	NUMBER	Target version number.
PARENT_TARGET_VERSION_STATUS	NUMBER	Status of the parent target version.
PARENT_TARGET_UTC_CHECKIN	NUMBER	UTC time (Coordinated Universal Time) when the parent target was checked in.
PARENT_TARGET_UTC_LAST_SAVED	NUMBER	UTC time when the target was last saved.
PARENT_TARGET_LAST_SAVED	VARCHAR2 (30)	Time when the target was last saved.
PARENT_TARGET_FIRST_FIELD_ID*	VARCHAR2	Link to first field of this table.
PARENT_TARGET_CONSTRAINT	VARCHAR2 (2000)	User-specified constraint string used when the DDL is generated.
PARENT_TARGET_CREATE_OPTIONS	VARCHAR2 (2000)	Options for use when generating DDL.

Column Name	Datatype	Description
PARENT_TARGET_FIRST_INDEX_ID*	NUMBER	Link to first field of this table.
PARENT_TARGET_FILE_ID	NUMBER	ID for the parent target file.
PARENT_TARGET_DATABASE_TYPE	VARCHAR2 (50)	Database type for the parent target.
SUBJECT_AREA	VARCHAR2 (240)	Folder name.
SUBJECT_ID*	NUMBER	Folder ID.
TARGET_NAME	VARCHAR2 (240)	Target name.
TARGET_ID*	NUMBER	Target ID.
TARGET_DESCRIPTION	VARCHAR2 (2000)	Target description.
TARGET_VERSION_NUMBER*	NUMBER	Target version number.
TARGET_VERSION_STATUS	NUMBER	Status of the target version.
TARGET_UTC_CHECKIN	NUMBER	UTC time (Coordinated Universal Time) when the target was last checked in.
TARGET_UTC_LAST_SAVED	NUMBER	UTC time when the target was last saved.
TARGET_LAST_SAVED	VARCHAR2 (30)	Time when the target was last saved.
REPOSITORY_NAME	VARCHAR2 (240)	Repository name.
IS_SHORTCUT	NUMBER	Specifies whether the target is a shortcut. 1 = shortcut; 0 = not a shortcut.
IS_GLOBAL_SHORTCUT	NUMBER	Specifies whether the target is a global shortcut. 1 = shortcut; 0 = not a shortcut.
**Indicates that the column is a key column.*		

REP_ALL_TARGET_FLDS

This view provides all the fields and field properties for targets defined in REP_ALL_TARGETS view. For global shortcuts, the shortcut name appears. For local shortcuts, the names of the shortcut and the parent targets display.

The following table lists target field data in the REP_ALL_TARGET_FLDS view:

Column Name	Datatype	Description
PARENT_SUBJECT_AREA	VARCHAR2 (240)	Folder name.
PARENT_SUBJECT_ID*	NUMBER	Folder ID.
PARENT_TARGET_NAME	VARCHAR2 (240)	Name of parent target.

Column Name	Datatype	Description
PARENT_TARGET_BUSINESS_NAME	VARCHAR2 (2000)	Business name of the parent target.
PARENT_TARGET_ID*	NUMBER	Parent target ID.
PARENT_TARGET_DESCRIPTION	VARCHAR2 (2000)	Description of parent target.
PARENT_TARGET_VERSION_NUMBER*	NUMBER	Target version number.
PARENT_TARGET_VERSION_STATUS	NUMBER	Status of the parent target version.
PARENT_TARGET_UTC_CHECKIN	NUMBER	UTC time (Coordinated Universal Time) when the parent target was last checked in.
PARENT_TARGET_UTC_LAST_SAVED	NUMBER	UTC time when the parent target was last saved.
PARENT_TARGET_LAST_SAVED	VARCHAR2 (30)	Time when the parent target was last modified.
PARENT_TARGET_FILE_ID*	NUMBER	ID of parent target file.
PARENT_TARGET_DATABASE_TYPE	VARCHAR2 (50)	Database type of parent target.
SUBJECT_AREA	VARCHAR2 (240)	Folder name.
SUBJECT_ID*	NUMBER	Folder ID.
TARGET_NAME	VARCHAR2 (240)	Target name.
TARGET_ID*	NUMBER	Target ID.
TARGET_DESCRIPTION	VARCHAR2 (2000)	Target description.
TARGET_VERSION_NUMBER*	NUMBER	Target version number.
TARGET_VERSION_STATUS	NUMBER	Status of the target version.
TARGET_UTC_CHECKIN	NUMBER	UTC time when the target was last checked in.
TARGET_UTC_LAST_SAVED	NUMBER	UTC time when the target was last saved.
TARGET_LAST_SAVED	VARCHAR2 (30)	Time when target was last saved.
TARGET_FIELD_NAME	VARCHAR2 (240)	Target field name.
TARGET_FIELD_BUSINESS_NAME	VARCHAR2 (240)	Business name of target field.
TARGET_FIELD_ID*	NUMBER	Target field ID.
TARGET_FIELD_DESCRIPTION	VARCHAR2 (2000)	Description of target field.
TARGET_FIELD_NUMBER	VARCHAR2 (240)	Target field number.
TARGET_FIELD_NEXT_FIELD_ID*	NUMBER	ID of the next field in target.
TARGET_FIELD_PICTURE_TEXT	VARCHAR2 (240)	Picture text that COBOL sources use.

Column Name	Datatype	Description
TARGET_FIELD_IS_NULLABLE	NUMBER	Specifies whether target field is null. 0 = Null; 1 = Not Null.
TARGET_FIELD_SOURCE_FIELD_ID*	NUMBER	Link to source from which this field was created.
TARGET_FIELD_KEY_TYPE	NUMBER	Key type of target field.
TARGET_FIELD_DATATYPE	VARCHAR2 (240)	Datatype of target field.
TARGET_FIELD_DATATYPE_GROUP	CHAR (1)	Datatype group codes. B = Binary and Bit C = Character, String, Text, and Byte D = Date N = Numeric and Money
TARGET_FIELD_PRECISION	NUMBER	Precision for target field.
TARGET_FIELD_SCALE	NUMBER	Scale for target field.
REPOSITORY_NAME	VARCHAR2 (240)	Repository name.
IS_SHORTCUT	NUMBER	Specifies whether the target is a shortcut. 1 = shortcut; 0 = not a shortcut.
IS_GLOBAL_SHORTCUT	NUMBER	Specifies whether the target is a global shortcut. 1 = shortcut; 0 = not a shortcut.
**Indicates that the column is a key column.*		

REP_TARG_TBLS

This view provides a list of targets in the repository. FIRST_COLUMN_ID is a link to the set of columns for this table. All tables contained in the target table model are part of this view. It is the primary table list used to delineate a PowerCenter data model. The tables are *virtual,* not physically created. Therefore, verify that the table exists before you use this view.

The following table lists the columns in the REP_TARG_TBLS view:

Column Name	Datatype	Description
SUBJECT_AREA*	VARCHAR2 (240)	Folder name.
TABLE_NAME*	VARCHAR2 (240)	Table name.
BUSNAME	VARCHAR2 (240)	Table business name.
VERSION_ID	INTEGER	Folder version ID.
VERSION_NAME	VARCHAR2 (240)	Folder version name.
DESCRIPTION	VARCHAR2 (2000)	Description of the table.

Column Name	Datatype	Description
FIRST_COLUMN_ID	INTEGER	Link to first field of this table.
TABLE_CONSTRAINT	VARCHAR2 (2000)	Table constraint specified in the Target Designer.
CREATE_OPTIONS	VARCHAR2 (2000)	Table creation options specified in the Target Designer.
FIRST_INDEX_ID	INTEGER	Link to first index.
LAST_SAVED	VARCHAR2 (30)	Time target table was last saved.
TARGET_VERSION_NUMBER*	NUMBER	Target version number.
SUBJECT_ID*	NUMBER	Folder ID.
TABLE_ID*	NUMBER	Table ID.
**Indicates that the column is a key column.*		

REP_TARG_TBL_COLS

This view provides the properties of columns defined for the target tables in a data warehouse or data mart.

The following table lists target table column properties for the REP_TARG_TBL_COLS view:

Column Name	Datatype	Description
SUBJECT_AREA*	VARCHAR2 (240)	Folder name.
TABLE_NAME*	VARCHAR2 (240)	Table this column belongs to.
TABLE_BUSNAME	VARCHAR2 (240)	Business name of the table.
COLUMN_NAME	VARCHAR2 (240)	Column name.
COLUMN_BUSNAME	VARCHAR2 (240)	Business name of this column.
COLUMN_NUMBER	INTEGER	Order number of the column.
COLUMN_ID*	INTEGER	Column ID (primary key).
VERSION_ID	INTEGER	Folder version ID.
VERSION_NAME*	VARCHAR2 (240)	Folder version name.
DESCRIPTION	VARCHAR2 (2000)	Column description.
COLUMN_KEYTYPE	VARCHAR2 (50)	Primary Key, Not a Key, Foreign Key, Primary and Foreign Key.
DATA_TYPE	VARCHAR2 (40)	Native database datatype.

Column Name	Datatype	Description
DATA_TYPE_GROUP	CHAR (1)	Datatype group. C = Character D = Date N = Numeric
DATA_PRECISION	INTEGER	Decimal precision for numeric fields or field length for CHAR fields.
DATA_SCALE	INTEGER	Decimal scale for numeric columns.
NEXT_COLUMN_ID	INTEGER	Link to next column.
IS_NULLABLE	INTEGER	Whether NULLs are accepted.
SOURCE_COLUMN_ID	INTEGER	Link to source this column was created from.
TARGET_VERSION_NUMBER	NUMBER	Target version number.
**Indicates that the column is a key column.*		

Mapping and Mapplet Views

Mapping and mapplet views allow you to see the sources, targets, and transformations used in a mapping or a mapplet by folder in a PowerCenter repository. These views also display properties of mappings and mapplets such as description, version and creation date, validity of the mapping or mapplet, and whether the mapping or mapplet is a shortcut.

When you save MX data for mappings, PowerCenter creates a field expression for each target field in the mappings. The field expression describes the source definition and transformation corresponding to the target field. In addition to viewing the MX data in MX views, you can view the field expressions in the Main window of the Repository Manager when you analyze source-target dependencies.

Note: MX views do not provide field expressions for all transformations. MX views provide field expressions for Expression, Aggregator, Rank, Lookup, Stored Procedure, External Procedure, Router, Custom, and Normalizer transformations. All other transformations produce NULL values in the TRANS_EXPRESSION column for views such as the REP_TARG_FLD_MAP view.

The following table lists the different views that help you analyze mapping and mapplet metadata:

View	Description
REP_ALL_MAPPINGS	This view provides a list of the latest version of all mappings defined in each folder of a repository.
REP_ALL_MAPPLETS	This view provides a list of the latest version of all mapplets defined in each folder of a repository.
REP_TARG_MAPPING	This view provides access to the compound table-level transformation expressions for each target table.

View	Description
REP_TARG_FLD_MAP	This view shows compound field-level transformation expressions associated with a target.
REP_FLD_MAPPING	This view shows the source fields used by the target fields in a mapping. This is the companion view for the REP_TBL_MAPPING view.
REP_SRC_MAPPING	This view shows all sources used in a mapping.
REP_SRC_FLD_MAP	This view shows all of the source fields used in a mapping.
REP_TBL_MAPPING	This view shows all of the target tables used in a mapping and provides source to target mapping information.
REP_TARG_TBL_JOINS	This view contains join information between target tables.
REP_MAPPING_CONN_PORTS	This view displays the port-level connections between the objects of a mapping.
REP_MAPPING_UNCONN_PORTS	This view displays the unconnected ports in sources, targets, and transformations in a mapping.

REP_ALL_MAPPINGS

This view provides a list of the latest version of all mappings defined in each folder of a repository. For local shortcuts, the names of the shortcut and the parent mappings display. For global shortcuts, the name of the shortcut appears.

The following table lists mapping information in the REP_ALL_MAPPINGS view:

Column Name	Datatype	Description
PARENT_SUBJECT_AREA*	VARCHAR2 (240)	Parent folder name.
PARENT_SUBJECT_ID*	NUMBER	Parent folder ID.
SUBJECT_AREA	VARCHAR2 (240)	Folder name.
SUBJECT_ID*	NUMBER	Folder ID.
PARENT_MAPPING_NAME	VARCHAR2 (240)	Name of the parent mapping.
PARENT_MAPPING_ID*	NUMBER	Sequence ID of the parent mapping.
PARENT_MAPPING_VERSION_NUMBER*	INTEGER	Parent mapping version number.
PARENT_MAPPING_VERSION_STATUS	NUMBER	Parent mapping version status.
PARENT_MAPPING_UTC_CHECKIN	NUMBER	UTC time (Coordinated Universal Time) when the parent mapping was checked in.
PARENT_MAPPING_UTC_LAST_SAVED	NUMBER	UTC time when mapping was last saved.
PARENT_MAPPING_LAST_SAVED	NUMBER	Date and time when parent mapping was last saved.

Column Name	Datatype	Description
PARENT_MAPPING_IS_VALID	NUMBER	Specifies whether the parent mapping is valid.
PARENT_MAPPING_DESCRIPTION	VARCHAR2 (2000)	Parent mapping description.
MAPPING_NAME	VARCHAR2 (240)	Name of mapping.
MAPPING_ID*	NUMBER	Sequence ID for mapping.
MAPPING_VERSION_NUMBER*	NUMBER	Mapping version number.
MAPPING_VERSION_STATUS	NUMBER	Status of the mapping version.
MAPPING_UTC_CHECKIN	NUMBER	UTC time when the mapping was checked in.
MAPPING_UTC_LAST_SAVED	NUMBER	UTC time when the mapping was last saved.
MAPPING_LAST_SAVED	NUMBER	Time when the mapping was last saved.
MAPPING_DESCRIPTION	VARCHAR2 (2000)	Mapping description.
REPOSITORY_NAME	VARCHAR2 (240)	Repository name.
IS_GLOBAL_SHORTCUT	NUMBER	Specifies whether the mapping is a global shortcut. 1 = shortcut; 0 = not a shortcut.
IS_SHORTCUT	NUMBER	Specifies whether the mapping is a shortcut. 1 = shortcut; 0 = not a shortcut.
**Indicates that the column is a key column.*		

REP_ALL_MAPPLETS

This view provides a list of the latest version of all mapplets defined in each folder of a repository. For local shortcuts, the names of the shortcut and the parent mapplets display. For global shortcuts, the name of the shortcut appears.

The following table lists mapplet metadata in the REP_ALL_MAPPLETS view:

Column Name	Datatype	Description
PARENT_SUBJECT_AREA	VARCHAR2 (240)	Parent folder name.
PARENT_SUBJECT_ID*	NUMBER	Parent folder ID.
SUBJECT_AREA	VARCHAR2 (240)	Folder name.
SUBJECT_ID*	NUMBER	Folder ID.
PARENT_MAPPLET_NAME	VARCHAR2 (240)	Name of parent mapplet.
PARENT_MAPPLET_ID*	NUMBER	Sequence ID of parent mapplet.

Column Name	Datatype	Description
PARENT_MAPPLET_VERSION_NUMBER*	INTEGER	Field ID (primary key).
PARENT_MAPPLET_VERSION_STATUS	NUMBER	Parent mapplet version status.
PARENT_MAPPLET_UTC_CHECKIN	NUMBER	UTC time (Coordinated Universal Time) when the parent mapplet was checked in.
PARENT_MAPPLET_UTC_LAST_SAVED	NUMBER	The UTC time when mapplet was last saved.
PARENT_MAPPLET_LAST_SAVED	NUMBER	The date and time when parent mapplet was last saved.
PARENT_MAPPLET_IS_VALID	NUMBER	Specifies whether the parent mapplet is valid.
PARENT_MAPPLET_DESCRIPTION	VARCHAR2 (2000)	Parent mapplet description.
MAPPLET_NAME	VARCHAR2 (240)	Name of mapplet.
MAPPLET_ID*	NUMBER	Mapplet ID.
MAPPLET_VERSION_NUMBER*	NUMBER	Mapplet version number.
MAPPLET_VERSION_STATUS	NUMBER	Status of the mapplet version.
MAPPLET_UTC_CHECKIN	NUMBER	UTC time when the mapplet was checked in.
MAPPLET_UTC_LAST_SAVED	NUMBER	UTC time when the mapplet was last saved.
MAPPLET_LAST_SAVED	NUMBER	Time when the mapplet was last saved.
MAPPLET_DESCRIPTION	VARCHAR2 (2000)	Mapplet description.
REF_WIDGET_ID*	NUMBER	Foreign key that points to generated mapplet transformation.
REPOSITORY_NAME	VARCHAR2 (240)	Repository name.
IS_GLOBAL_SHORTCUT	NUMBER	Specifies whether the mapplet is a global shortcut. 1 = shortcut; 0 = not a shortcut.
IS_SHORTCUT	NUMBER	Specifies whether the mapplet is a shortcut. 1 = shortcut; 0 = not a shortcut.
**Indicates that the column is a key column.*		

REP_TARG_MAPPING

This view provides access to the compound table-level transformation expressions for each target table. This view pulls information from all the transformation objects that contribute to the target table in a valid mapping. This view contains information about mappings defined for target tables. It does not contain information about sources.

Note: Use the REP_TBL_MAPPING view to analyze source and target relationships. Join REP_TBL_MAPPING and REP_TARG_MAPPING by MAPPING_NAME to include column-level mapping information in the REP_FLD_MAPPING view.

The following table lists expression information in the REP_TARG_MAPPING view:

Column Name	Datatype	Description
TARGET_NAME*	VARCHAR2 (240)	Target name.
TARG_BUSNAME	VARCHAR2 (240)	Target business name.
SUBJECT_AREA*	VARCHAR2 (240)	Folder name.
MAPPING_NAME*	VARCHAR2 (240)	Mapping name.
VERSION_ID	INTEGER	Folder version ID.
VERSION_NAME	VARCHAR2 (240)	Folder version name.
SOURCE_FILTER	VARCHAR2 (2000)	Compound source filter condition.
CONDITIONAL_LOAD	VARCHAR2 (2000)	Compound conditional load.
GROUP_BY_CLAUSE	VARCHAR2 (2000)	Compound group by expression.
SQL_OVERRIDE	VARCHAR2 (2000)	Compound SQL override expression.
DESCRIPTION	VARCHAR2 (2000)	Description of transformation expression.
MAPPING_COMMENT	VARCHAR2 (2000)	Description of mapping.
MAPPING_LAST_SAVED	VARCHAR2 (30)	Time the mapping was saved last.
MAPPING_VERSION_NUMBER	NUMBER	Mapping version number.
TARGET_VERSION_NUMBER	NUMBER	Target version number.
**Indicates that the column is a key column.*		

REP_TARG_FLD_MAP

This view shows compound field-level transformation expressions associated with a target. This view pulls information from all transformation objects that contribute to the target table in a valid mapping. There might be many mappings for a set of fields, each distinguished by the MAPPING_NAME field. The field-level expression contains all the source fields (both file definition/non-relational source) that determine the value of the target field.

The following table lists expression metadata that you find in the REP_TARG_FLD_MAP view:

Column Name	Datatype	Description
TARGET_COLUMN_NAME	VARCHAR2 (240)	Name of target field (table field).
TARG_COL_BUSNAME	VARCHAR2 (240)	Business name of target field.
TARGET_NAME	VARCHAR2 (240)	Name of target (table).
TARG_BUSNAME	VARCHAR2 (240)	Business name of target table.
SUBJECT_AREA	VARCHAR2 (240)	Folder name.
MAPPING_NAME	VARCHAR2 (240)	Mapping name.
VERSION_ID	INTEGER	Folder version ID.
VERSION_NAME	VARCHAR2 (240)	Folder version name.
TRANS_EXPRESSION	VARCHAR2 (2000)	Compound transformation expression.
USER_COMMENT	VARCHAR2 (2000)	End user comment.
DBA_COMMENT	VARCHAR2 (2000)	Administrator comment.
MAPPING_COMMENT	VARCHAR2 (2000)	Mapping comment.
MAPPING_LAST_SAVED	VARCHAR2 (30)	Time the mapping was saved last.
MAPPING_VERSION_NUMBER	NUMBER	Mapping version number.
TARGET_VERSION_NUMBER	NUMBER	Target version number.

REP_FLD_MAPPING

This view shows the source fields used by the target fields in a mapping. This is the companion view for the REP_TBL_MAPPING view. It contains both source and target column names and details.

The following table lists the source and target field metadata in the REP_FLD_MAPPING view:

Column Name	Datatype	Descriptions
SOURCE_FIELD_NAME*	VARCHAR2 (240)	Name of the source field.
SRC_FLD_BUSNAME	VARCHAR2 (240)	Business name of the source field.
SOURCE_NAME	VARCHAR2 (240)	Name of the source table.
SRC_BUSNAME	VARCHAR2 (240)	Business name of the source table.
TARGET_COLUMN_NAME*	VARCHAR2 (240)	Name of the target field.
TARG_COL_BUSNAME	VARCHAR2 (240)	Business name of the target column.

Column Name	Datatype	Descriptions
TARGET_NAME	VARCHAR2 (240)	Target name.
TARG_BUSNAME	VARCHAR2 (240)	Business name of the target.
SUBJECT_AREA*	VARCHAR2 (240)	Folder name.
SUBJECT_ID*	NUMBER	Folder ID.
MAPPING_NAME	VARCHAR2 (240)	Name of the mapping.
VERSION_ID	NUMBER	Folder version ID.
VERSION_NAME	VARCHAR2 (240)	Folder version name.
TRANS_EXPRESSION	VARCHAR2 (2000)	Target field transformation expression.
USER_COMMENT	VARCHAR2 (2000)	End user comment.
DBA_COMMENT	VARCHAR2 (2000)	Administrator comment.
MAPPING_COMMENT	VARCHAR2 (2000)	Mapping comment.
MAPPING_LAST_SAVED	VARCHAR2 (240)	Time the mapping was saved last.
MAPPING_ VERSION_NUMBER*	NUMBER	Mapping version number.
SOURCE_VERSION_NUMBER*	NUMBER	Source version number.
TARGET_VERSION_NUMBER*	NUMBER	Target version number.
SOURCE_ID*	NUMBER	Source table ID.
TARGET_ID*	NUMBER	Target table ID.
MAPPING_ID*	NUMBER	Mapping ID.
**Indicates that the column is a key column.*		

REP_SRC_MAPPING

This view shows all sources used in a mapping. Query this view by MAPPING_NAME and VERSION_NAME. A mapping might contain several sources. This view contains the mapping names defined for an individual source table. It does not contain information about the targets involved in a mapping. The REP_TBL_MAPPING view contains the entire source and target mapping relationship.

The following table lists mapping source metadata in the REP_SRC_MAPPING view:

Column Name	Datatype	Description
SOURCE_NAME	VARCHAR2 (240)	Name of the source.
SOURCE_ID	NUMBER	Source ID.
SRC_BUSNAME	VARCHAR2 (240)	Business name of source table.
SUBJECT_AREA*	VARCHAR2 (240)	Folder name.
SUBJECT_ID*	NUMBER	Folder ID.
MAPPING_NAME	VARCHAR2 (240)	Mapping name.
MAPPING_ID*	NUMBER	Mapping ID.
VERSION_ID	INTEGER	Folder version ID.
VERSION_NAME	VARCHAR2 (240)	Folder version name.
MAPPING_COMMENT	VARCHAR2 (2000)	Mapping comment.
MAPPING_LAST_SAVED	VARCHAR2 (30)	Time the mapping was last saved.
MAPPING_VERSION*_NUMBER	NUMBER	Mapping version number.
SOURCE_VERSION_NUMBER*	NUMBER	Source version number.
**Indicates that the column is a key column.*		

REP_SRC_FLD_MAP

This view shows all of the source fields used in a mapping. The transformation expression corresponds to the target fields that get data from a particular source field. This view creates these expressions by pulling information from all transformation objects that contribute to the target table in a valid mapping.

The following table lists mapping source fields:

Column Name	Datatype	Description
SOURCE_FIELD_NAME*	VARCHAR2 (240)	Source field name.
SRC_FLD_BUSNAME	VARCHAR2 (240)	Business name of the field.
SOURCE_NAME*	VARCHAR2 (240)	Name of the source.
SRC_BUSNAME	VARCHAR2 (240)	Business name of the source table.
SUBJECT_AREA*	VARCHAR2 (240)	Folder name.
MAPPING_NAME*	VARCHAR2 (240)	Name of the mapping.
VERSION_ID	INTEGER	Folder version ID.

Column Name	Datatype	Description
VERSION_NAME	VARCHAR2 (240)	Folder version name.
TRANS_EXPRESSION	VARCHAR2 (2000)	Compound target. Field transformation expression.
USER_COMMENT	VARCHAR2 (2000)	End user comment.
DBA_COMMENT	VARCHAR2 (2000)	Administrator comment.
MAPPING_COMMENT	VARCHAR2 (2000)	Mapping comment.
MAPPING_LAST_SAVED	VARCHAR2 (30)	Time the mapping was saved last.
SOURCE_VERSION_NUMBER	NUMBER	Source version number.
TARGET_VERSION_NUMBER	NUMBER	Target version number.
**Indicates that the column is a key column.*		

REP_TBL_MAPPING

This view shows all of the target tables used in a mapping and provides source to target mapping information. This view pulls information from all transformation objects that contribute to the target table in a valid mapping to provide the table-level expressions.

The following table lists mapping target metadata in the REP_TBL_MAPPING view:

Column Name	Datatype	Description
SOURCE_NAME*	VARCHAR2 (240)	Name of the source object.
SOURCE_ID*	NUMBER	Source ID.
SRC_BUSNAME	VARCHAR2 (240)	Business name of the source.
TARGET_NAME	VARCHAR2 (240)	Target name.
TARGET_ID*	NUMBER	Target ID.
TARG_BUSNAME	VARCHAR2 (240)	Business name of the target.
SUBJECT_AREA*	VARCHAR2 (240)	Folder name.
SUBJECT_ID	NUMBER	Folder ID.
MAPPING_NAME	VARCHAR2 (240)	Name of the mapping.
MAPPING_ID*	NUMBER	Mapping ID.
VERSION_ID	INTEGER	Folder version ID.
VERSION_NAME	VARCHAR2 (240)	Folder version name.

Column Name	Datatype	Description
SOURCE_FILTER	VARCHAR2 (2000)	Compound source filter condition.
CONDITIONAL_LOAD	VARCHAR2 (2000)	Compound conditional load.
GROUP_BY_CLAUSE	VARCHAR2 (2000)	Compound group by clause.
SQL_OVERRIDE	VARCHAR2 (2000)	Compound SQL override expression.
DESCRIPTION	VARCHAR2 (2000)	Description of transformation.
MAPPING_COMMENT	VARCHAR2 (2000)	Mapping comment.
MAPPING_LAST_SAVED	VARCHAR2 (240)	Time the mapping was saved last.
MAPPING_VERSION_NUMBER*	NUMBER	Mapping version number.
SOURCE_VERSION_NUMBER*	NUMBER	Source version number.
TARGET_VERSION_NUMBER*	NUMBER	Target version number.
**Indicates that the column is a key column.*		

REP_TARG_TBL_JOINS

This view contains join information between target tables. Use this view to query the PowerCenter defined joins for a target table model. It is populated when you link fields in the Target Designer or through primary key-foreign key relationships.

The following table lists target table join metadata in the REP_TARG_TBL_JOINS view:

Column Name	Datatype	Description
SUBJECT_AREA*	VARCHAR2 (240)	Folder name.
VERSION_ID	NUMBER	Folder version ID.
VERSION_NAME	VARCHAR2 (240)	Folder version name.
TABLE1_NAME	VARCHAR2 (240)	Name of first table in the join.
TABLE1_BUSNAME	VARCHAR2 (240)	Business name of first table.
TABLE1_ID*	NUMBER	ID of first table in the join.
COLUMN1_NAME	VARCHAR2 (240)	Name of column in first table.
COLUMN1_BUSNAME	VARCHAR2 (240)	Business name of column in first table.
COLUMN1_NUMBER	NUMBER	Number of column in first table.
COLUMN1_ID*	NUMBER	ID of column in first table.

Column Name	Datatype	Description
TABLE2_NAME	VARCHAR2 (240)	Name of second table in the join.
TABLE2_BUSNAME	VARCHAR2 (240)	Business name of second table.
TABLE2_ID*	NUMBER	ID of second table in the join.
COLUMN2_NAME	VARCHAR2 (240)	Name of column in second table.
COLUMN2_BUSNAME	VARCHAR2 (240)	Business name of column in second table.
COLUMN2_NUMBER	VARCHAR2 (240)	Number of column in second table.
COLUMN2_ID	NUMBER	ID of column in second table.
TABLE1_VERSION_NUMBER	NUMBER	Table1 version number.
TABLE2_VERSION_NUMBER	NUMBER	Table2 version number.
**Indicates that the column is a key column.*		

REP_MAPPING_CONN_PORTS

This view displays the port-level connections between the objects of a mapping. Objects include sources, targets, transformations, and mapplets. Unconnected transformations are not included.

The following table lists port-level connection metadata in the REP_MAPPING_CONN_PORTS view:

Column Name	Datatype	Description
SUBJECT_ID*	NUMBER	Folder ID.
SUBJECT_AREA	VARCHAR2 (240)	Folder name.
MAPPING_ID*	NUMBER	Sequence ID for the mapping (primary key).
MAPPING_NAME	VARCHAR2 (240)	Mapping name.
MAPPING_VERSION_NUMBER*	NUMBER	Mapping version number.
FROM_OBJECT_ID*	NUMBER	Source object ID.
FROM_OBJECT_TYPE	NUMBER	Source object type.
FROM_OBJECT_TYPE_NAME	VARCHAR2 (240)	Name of the source object type.
FROM_OBJECT_NAME	VARCHAR2 (240)	Source object name.
FROM_OBJECT_INSTANCE_ID*	NUMBER	Source object instance ID.
FROM_OBJECT_FIELD_NAME	VARCHAR2 (240)	Source object field name.
FROM_OBJECT_VERSION_NUMBER	NUMBER	Source object version number.

Column Name	Datatype	Description
TO_OBJECT_ID*	NUMBER	Target object ID.
TO_OBJECT_TYPE	NUMBER	Target object type such as port, target, mapplet, and transformation.
TO_OBJECT_TYPE_NAME	VARCHAR2 (240)	Target object type name.
TO_OBJECT_NAME	VARCHAR2 (240)	Target object name.
TO_OBJECT_INSTANCE_ID*	NUMBER	Target object instance ID.
TO_OBJECT_FIELD_NAME	VARCHAR2 (240)	Target object field name.
TO_OBJECT_VERSION_NUMBER*	NUMBER	Target object version number.
**Indicates that the column is a key column.*		

REP_MAPPING_UNCONN_PORTS

This view displays the unconnected ports in sources, targets, and transformations in a mapping.

The following table lists unconnected port metadata in the REP_MAPPING_UNCONN_PORTS view:

Column Name	Datatype	Description
SUBJECT_AREA	VARCHAR2 (240)	Folder name.
SUBJECT_ID*	NUMBER	Folder ID (primary key).
MAPPING_NAME	VARCHAR2 (240)	Name of the mapping.
MAPPING_ID*	NUMBER	Sequence ID for the mapping (primary key).
MAPPING_ VERSION_NUMBER*	NUMBER	Mapping version number.
OBJECT_INSTANCE_NAME	VARCHAR2 (240)	Name of the instance.
OBJECT_INSTANCE_ID*	NUMBER	Unique ID for the instance in a mapping.
OBJECT_TYPE	NUMBER	Object type.
OBJECT_TYPE_NAME	VARCHAR2 (240)	Object type name.
FIELD_ID*	NUMBER	Source field ID (primary key).
FIELD_NAME	VARCHAR2 (240)	Source field name.
OBJECT_VERSION_NUMBER*	NUMBER	Version number of the source, target, or transformation.
**Indicates that the column is a key column.*		

Metadata Extension Views

Metadata views allow you to see metadata extension details including reusable metadata extensions defined for objects in metadata extension domains.

The following table lists the different views that help you analyze metadata extensions metadata:

View	Description
REP_METADATA_EXTNS	This view displays the details of all metadata extensions in the repository.
REP_METADATA_EXTN_DEFINES	This view displays reusable metadata extensions defined for objects in metadata extension domains.

REP_METADATA_EXTNS

This view displays the details of all metadata extensions in the repository.

The following table lists metadata extension information in the REP_METADATA_EXTNS view:

Column Name	Datatype	Description
SUBJECT_ID*	NUMBER	Folder ID.
METADATA_EXTN_NAME	VARCHAR2 (240)	Metadata extension name.
METADATA_EXTN_OBJECT_TYPE	NUMBER	Object type the metadata is associated with.
METADATA_EXTN_OBJECT_ID*	NUMBER	Object the metadata value is associated with.
METADATA_EXTN_DATA_TYPE	NUMBER	Datatype of the metadata extension value.
PERMISSIONS	NUMBER	Permissions type.
METADATA_EXTN_VALUE	VARCHAR2 (2000)	Metadata extension value.
LINE_NO	NUMBER	Line number of the text when there are multiple lines of text.
METADATA_EXTN_DESCRIPTION	VARCHAR2 (2000)	Description of the metadata extension.
VERSION_NUMBER*	NUMBER	Object version number.
OBJECT_TYPE_NAME	NUMBER	Name of the object type.
DOMAIN_ID*	NUMBER	Globally unique domain identifier.
DOMAIN_NAME	VARCHAR2 (240)	Unique name for a user-defined metadata domain.
DOMAIN_KEY	VARCHAR2 (240)	Domain key.
DOMAIN_USAGE	NUMBER	Specifies domain usage. 1= Domain is visible through client tool. 2= Domain is editable through client tool. 4 = Domain has full access without a key.

Column Name	Datatype	Description
DOMAIN_DESCRIPTION	VARCHAR2 (2000)	Domain description.
*Indicates that the column is a key column.		

REP_METADATA_EXTN_DEFINES

This view displays reusable metadata extensions defined for objects in metadata extension domains.

The following table lists information in the REP_METADATA_EXTN_DEFINES view:

Column Name	Datatype	Description
DOMAIN_NAME	VARCHAR2 (240)	Unique name for a user-defined domain.
DOMAIN_ID*	NUMBER	Globally unique domain identifier.
METAEXT_NAME	VARCHAR2 (240)	Unique name for metadata within a domain.
OBJECT_TYPE_NAME	VARCHAR2 (240)	Object type name.
DATABASE_TYPE	VARCHAR2 (240)	Name of the database type.
METADATA_EXTN_DESCRIPTION	VARCHAR2 (2000)	Description of the metadata extension.
VENDOR_NAME	VARCHAR2 (240)	Name of the vendor.
*Indicates that the column is a key column.		

Transformation Views

Transformation views display details of all reusable and non-reusable transformation instances by folder in a PowerCenter repository. These views also display properties such as attributes, dependencies, port-level connections, and field level details of transformations.

The following table lists the different views that help you analyze transformation metadata:

View	Description
REP_ALL_TRANSFORMS	This view provides a list of the latest version of all transformations and their properties in each folder of a repository.
REP_WIDGET_INST	This view displays the details of all transformation instances.
REP_WIDGET_DEP	This view displays the details of dependencies between transformation instances in a mapping.

View	Description
REP_WIDGET_ATTR	This view displays attribute details for transformations, instances, and sessions.
REP_WIDGET_FIELD	This view displays field level details for transformations.

REP_ALL_TRANSFORMS

This view provides a list of the latest version of all transformations and their properties in each folder of a repository. This view displays both reusable transformations defined in the Transformation Designer and transformation instances defined in mapping and mapplets. It also shows all shortcut transformations in a folder. For local shortcuts, the names of the shortcut and the parent transformation display. For global shortcuts, the name of the shortcut appears.

The following table lists transformation metadata in the REP_ALL_TRANFORMS view:

Column Name	Datatype	Description
PARENT_SUBJECT_AREA	VARCHAR2 (240)	Parent folder name.
PARENT_SUBJECT_ID*	NUMBER	Parent folder ID.
SUBJECT_AREA	VARCHAR2 (240)	Folder name.
SUBJECT_ID*	NUMBER	Folder ID.
PARENT_WIDGET_NAME	VARCHAR2 (240)	Name of the parent transformation.
PARENT_WIDGET_ID*	NUMBER	Parent transformation ID (primary key).
PARENT_WIDGET_VERSION_NUMBER*	NUMBER	Parent transformation ID.
PARENT_WIDGET_VERSION_STATUS	NUMBER	Status of the parent transformation version.
PARENT_WIDGET_UTC_CHECKIN	NUMBER	UTC time (Coordinated Universal Time) when the parent transformation was last checked in.
PARENT_WIDGET_UTC_LAST_SAVED	NUMBER	UTC time when the parent transformation was last saved.
PARENT_WIDGET_LAST_SAVED	VARCHAR2 (30)	Date and time when transformation was last saved.
PARENT_WIDGET_IS_REUSABLE	NUMBER	Specifies whether the transformation is reusable. 1= reusable; 0 = not reusable.
PARENT_WIDGET_DESCRIPTION	VARCHAR2 (2000)	Parent transformation description.
WIDGET_NAME	VARCHAR2 (240)	Name of the transformation.
WIDGET_ID*	NUMBER	Transformation ID.
WIDGET_VERSION_NUMBER*	NUMBER	Version number of the transformation.
WIDGET_VERSION_STATUS	NUMBER	Status of the transformation version.

Column Name	Datatype	Description
WIDGET_UTC_CHECKIN	NUMBER	UTC time when the transformation was checked in.
WIDGET_UTC_LAST_SAVED	NUMBER	UTC time when the transformation was last saved.
WIDGET_LAST_SAVED	VARCHAR2 (30)	Time when the transformation was last saved.
WIDGET_TYPE_ID*	NUMBER	Transformation type ID.
WIDGET_TYPE_NAME	VARCHAR2 (240)	Transformation type name.
WIDGET_DESCRIPTION	VARCHAR2 (2000)	Transformation description.
REPOSITORY_NAME	VARCHAR2 (240)	Repository name.
IS_GLOBAL_SHORTCUT	NUMBER	Specifies whether the transformation is a global shortcut. 1 = shortcut; 0 = not a shortcut.
IS_SHORTCUT	NUMBER	Specifies whether the transformation is a shortcut. 1 = shortcut; 0 = not a shortcut.
**Indicates that the column is a key column.*		

REP_WIDGET_INST

This view displays the details of all transformation instances.

The following table list transformation metadata in the REP_WIDGET_INST view:

Column Name	Datatype	Description
MAPPING_ID*	NUMBER	Mapping ID.
WIDGET_ID*	NUMBER	Transformation ID.
WIDGET_TYPE	NUMBER	Transformation type.
WIDGET_TYPE_NAME	VARCHAR2 (240)	Transformation name.
INSTANCE_ID*	NUMBER	ID of the transformation instance.
INSTANCE_NAME	VARCHAR2 (240)	Name of the transformation instance.
DESCRIPTION	VARCHAR2 (2000)	Description of the transformation instance.
VERSION_NUMBER*	NUMBER	Version number of the transformation.
REF_WIDGET_ID	NUMBER	0 for mappings. For mapplets, contains a foreign key which points to a table that has the generated mapplet widget.

Column Name	Datatype	Description
SUBJECT_ID	NUMBER	Folder ID.
**Indicates that the column is a key column.*		

REP_WIDGET_DEP

This view displays the details of dependencies between transformation instances in a mapping.

The following table lists transformation dependency information in the REP_WIDGET_DEP view:

Column Name	Datatype	Description
MAPPING_ID*	NUMBER	Mapping ID.
FROM_INSTANCE_ID*	NUMBER	Source transformation instance ID.
FROM_FIELD_ID*	NUMBER	Field ID of the source transformation instance.
TO_INSTANCE_ID*	NUMBER	Field ID of the target transformation instance.
TO_FIELD_ID*	NUMBER	Target field ID.
VERSION_ NUMBER*	NUMBER	Version number of the mapping.
**Indicates that the column is a key column.*		

REP_WIDGET_ATTR

This view displays attribute details for transformations, instances, and sessions.

The following table lists attribute details for transformations, instances, and sessions:

Column Name	Datatype	Description
WIDGET_ID*	NUMBER	Transformation ID.
WIDGET_TYPE	NUMBER	Transformation type.
MAPPING_ID*	NUMBER	Mapping ID.
INSTANCE_ID*	NUMBER	Instance ID.
ATTR_ID*	NUMBER	Attribute ID.
ATTR_DESCRIPTION	VARCHAR2 (2000)	Description of the attribute.
ATTR_DATATYPE	NUMBER	Attribute data type.
ATTR_NAME	VARCHAR2 (240)	Attribute name.

Column Name	Datatype	Description
ATTR_TYPE	NUMBER	Attribute type.
LINE_NO	NUMBER	Used to break up long strings into multiple rows.
ATTR_VALUE	VARCHAR2 (2000)	Attribute value.
PARTITION_ID*	NUMBER	Partition ID.
SESSION_TASK_ID*	NUMBER	Session task ID.
VERSION_NUMBER*	NUMBER	Object (session, mapping, or transformation) version number.
**Indicates that the column is a key column.*		

REP_WIDGET_FIELD

This view displays field level details for transformations.

The following table lists transformation field information in the REP_WIDGET_FIELD view:

Column Name	Datatype	Description
WIDGET_ID*	NUMBER	Transformation ID.
FIELD_NAME	VARCHAR2 (240)	Transformation field name.
FIELD_ID*	NUMBER	Transformation field ID.
WGT_PREC	NUMBER	Transformation field precision.
WGT_SCALE	NUMBER	Transformation field scale.
WGT_DATATYPE	NUMBER	Transformation field data type.
PORTTYPE	NUMBER	Transformation port type.
FIELD_ORDER	NUMBER	Transformation order.
DESCRIPTION	VARCHAR2 (2000)	Comments on the field.
PROPERTY	NUMBER	Field-level property used by transformations.
DEFAULT_VALUE	VARCHAR2 (2000)	Default value of the transformation field.
SRC_FIELD_ID*	NUMBER	Source field ID for normalizer transformation.
GROUP_ID*	NUMBER	ID of the corresponding instance in the mapplet's mapping.
VERSION_NUMBER*	NUMBER	Transformation version number.
DATATYPE_NUM	NUMBER	Datatype number.

Column Name	Datatype	Description
DATATYPE	VARCHAR2 (40)	Transformation datatype of the port.
DATATYPE_GROUP_CODE	CHAR (1)	Datatype group code.
DATABASE_TYPE	VARCHAR2 (20)	External database type.
EXPRESSION	VARCHAR2 (2000)	Expression name.
EXPR_DESCRIPTION	VARCHAR2 (2000)	Comments on the expression.
EXPR_TYPE	NUMBER	Expression type.
**Indicates that the column is a key column.*		

Workflow, Worklet, and Task Views

Workflow, worklet, and task views provide both static and run time details about all workflows and worklets created in each folder in a PowerCenter repository.

These views provide information about worklets and sessions inside a workflow. The views also provide information about events, schedules, tasks, connections, and metadata extensions associated with a workflow or a worklet; workflow and worklet execution details such as start time, end time, and the Integration Service on which a workflow or worklet runs and its run status.

Task views provide both static and run time details about tasks such as sessions created in each folder of a PowerCenter repository. These views provide information such as the validity of a session, creation date, sources and targets defined in a session, session connections, and metadata extensions associated with a session. These views also give information about session runtime details like start time, end time, and run status.

The following table lists the different views that help you analyze workflow, worklet, and task metadata:

View	Description
REP_WORKFLOWS	This view contains information about individual workflows and workflow scheduling.
REP_ALL_TASKS	This view provides a list of all reusable and non-reusable tasks that can be used by a workflow or a worklet.
REP_ALL_SCHEDULERS	This view displays a list of schedulers by folder.
REP_WFLOW_VAR	This view displays a list of all variables declared within a workflow or worklet.
REP_EVENT	This view displays the details of events created at the workflow or worklet level.
REP_TASK_INST	This view displays all task instances within workflows and worklets.

View	Description
REP_WORKFLOW_DEP	This view shows how individual tasks and worklets are connected within a worklet or a workflow.
REP_TASK_INST_RUN	This view displays the run statistics and folder reference for tasks within a workflow or worklet.
REP_WFLOW_RUN	This view displays the run statistics for all workflows by folder.
REP_LOAD_SESSIONS	This view provides information about sessions in the repository.
REP_SESSION_CNXS	This view contains information about connections associated with reusable sessions.
REP_SESSION_INSTANCES	This view contains connection information for session instances.
REP_SESSION_FILES	This view contains file connections associated with reusable sessions.
REP_SESSION_INST_FILES	This view contains file connection information for session instances associated with workflows.
REP_SESS_WIDGET_CNXS	This view contains information about the sources and targets used in a session.
REP_COMPONENT	This view displays the list of tasks such as a command or an email for each session.
REP_SESS_PARTITION_DEF	This view provides partition details of the sources, targets, and transformations in a session.
REP_SESS_CONFIG_PARM	This view displays session configuration parameter details. If the session overrides a parameter in the configured object, the view displays two rows.
REP_SESS_INST_CONFIG_PARM	This view displays the attributes that are overriden at the session instance.
REP_TASK_ATTR	This view displays the attribute values and overridden values for session and workflow tasks.
REP_SESS_LOG	This view provides log information about sessions.
REP_SESS_TBL_LOG	This view contains information about the status of an individual session run against a target.

REP_WORKFLOWS

This view contains information about individual workflows and workflow scheduling.

The following table lists workflow and scheduling information in the REP_WORKFLOWS view:

Column Name	Datatype	Description
SUBJECT_AREA*	VARCHAR2 (240)	Folder name.
WORKFLOW_NAME*	VARCHAR2 (240)	Workflow name.

Column Name	Datatype	Description
SCHEDULER_NAME*	VARCHAR2 (240)	Scheduler associated with the workflow.
START_TIME	TIMESTAMP	Start time configured for the scheduler.
END_TIME	TIMESTAMP	End time configured for the scheduler.
IS_RUN_ON_LIMIT	NUMBER	Object ID used for cyclic redundancy check (CRC).
RUN_OPTIONS	INTEGER	The workflow schedule type. Records the following values for each schedule type: 1 = Run on demand. 2 = Run once. 4 = Run every DELTA_VALUE seconds. 8 = Customized repeat. 16 = Run on Integration Service initialization. 18 = Run on Integration Service initialization and run once. 20 = Run on Integration Service initialization and every DELTA_VALUE seconds. 24 = Run on Integration Service initialization and customized repeat. 32 = Run continuously.
END_OPTIONS	INTEGER	The stop condition option for the workflow schedule type. Records the following values for each stop condition option: 0 = End on a date. 1 = End after the number of runs stored in RUN_COUNT. 2 = Run forever.
DELTA_VALUE	NUMBER	Number of seconds the Integration Service waits between successive workflow runs.
RUN_COUNT	INTEGER	Number of times the Integration Service runs the workflow before stopping the workflow.
SCHEDULER_ID*	NUMBER	Scheduler ID.
SCHEDULER_IS_REUSABLE	NUMBER	Specifies if scheduler is reusable.
SCHEDULER_COMMENTS	VARCHAR2 (2000)	Scheduler description.
SCHEDULER_VERSION_NUMBER*	NUMBER	Version number of the scheduler.
WORKFLOW_VERSION_NUMBER*	NUMBER	Workflow version number.
WORKFLOW_ID*	NUMBER	Workflow ID.
WORKFLOW_IS_VALID	NUMBER	Specifies whether the workflow is valid or not. 1 = valid; 0 = invalid.

Column Name	Datatype	Description
WORFLOW_IS_SERVICE	NUMBER	Specifies whether the workflow is a service. 1 = service; 0 = not a service.
WORKFLOW_IS_RUNNABLE_SERVICE	NUMBER	Specifies whether the workflow is a runnable service. 1 = runnable service; 0 = not a runnable service.
WORKFLOW_LAST_SAVED	VARCHAR2 (30)	Date and time when the workflow was last saved.
WORKFLOW_COMMENTS	VARCHAR2 (2000)	Description of the workflow.
SUBJECT_ID*	NUMBER	Folder ID.
SERVER_NAME	VARCHAR2 (240)	Name of the Integration Service registered with the repository.
SERVER_ID	NUMBER	Integration Service ID.
WORKFLOW_IS_IMPACTED	NUMBER	Specifies whether the workflow is impacted by a change to dependent objects that may require the workflow to be revalidated. 0 = not impacted; 1 = impacted.
**Indicates that the column is a key column.*		

REP_ALL_TASKS

This view provides a list of all reusable and non-reusable tasks that can be used by a workflow or a worklet.

The following table lists reusable and non-reusable task information in the REP_ALL_TASKS view:

Column Name	Datatype	Description
SUBJECT_AREA	VARCHAR2 (240)	Folder name.
SUBJECT_ID*	NUMBER	Folder ID.
TASK_NAME	VARCHAR2 (240)	Task name.
TASK_ID*	NUMBER	Task ID.
IS_VALID	NUMBER	Specifies whether a workflow, worklet, or session is valid. 1 = valid; 0 = invalid.
LAST_SAVED	VARCHAR2 (30)	Time when task was last saved.
DESCRIPTION	VARCHAR2 (2000)	Description of the task.
VERSION_NUMBER*	NUMBER	Version number of the task.
IS_ENABLED	NUMBER	Specifies whether the task is enabled or not. 1 = enabled; 0 = disabled.

Column Name	Datatype	Description
UTC_CHECKIN	NUMBER	UTC checkin time.
UTC_LAST_SAVED	VARCHAR2 (30)	UTC time when task was last saved.
IS_REUSABLE	NUMBER	Specifies whether the task is reusable or not. Values are: 1 = reusable; 0 = not reusable.
TASK_TYPE	NUMBER	Task type.
TASK_TYPE_NAME	VARCHAR2 (240)	Task type name.
**Indicates that the column is a key column.*		

REP_ALL_SCHEDULERS

This view displays a list of schedulers by folder.

The following table lists information in the REP_ALL_SCHEDULERS view:

Column Name	Datatype	Description
SUBJECT_ID*	NUMBER	Folder ID.
SCHEDULER_ID*	NUMBER	Scheduler ID (primary key).
SCHEDULER_NAME	VARCHAR2 (240)	Name of the scheduler.
START_TIME	VARCHAR2 (30)	Start time configured for the object associated with the scheduler.
END_TIME	VARCHAR2 (30)	End time configured for the object associated with the scheduler.
RUN_OPTIONS	NUMBER	The scheduler type. Records the following values for each schedule type: 1 = Run on demand. 2 = Run once schedule. 3 = Run on demand and Run once schedule. 5 = Run on demand and Delta schedule. 9 = Run on demand and Custom repeat. 18 = Run on server init and Run once schedule. 20 = Run on server init and Delta schedule. 24 = Run on server init and Custom repeat. 34 = Run continuously and Run once schedule. 36 = Run continuously and Delta schedule. 40 = Run continuously and Custom repeat.
END_OPTIONS	NUMBER	Specifies when the task must stop running.
DELTA_VALUE	NUMBER	Delta between successive runs (stored as seconds).
RUN_COUNT	NUMBER	Number of workflow runs. Used by END_OPTIONS column.

Column Name	Datatype	Description
DESCRIPTION	VARCHAR2 (2000)	Description of the scheduler.
IS_REUSABLE	NUMBER	Specifies whether the scheduler is reusable or not.
LAST_SAVED	NUMBER	Date and time when this task was last saved.
VERSION_NUMBER*	NUMBER	Version number of the scheduler.
UTC_LAST_SAVED	NUMBER	UTC time (Coordinated Universal Time) when the scheduler was last saved.
UTC_CHECKIN	NUMBER	UTC checkin time.
**Indicates that the column is a key column.*		

REP_WFLOW_VAR

This view displays a list of all variables declared within a workflow or worklet.

The following table lists variable information in the REP_WFLOW_VAR view:

Column Name	Datatype	Description
SUBJECT_ID*	NUMBER	Folder ID.
WORKFLOW_ID*	NUMBER	Workflow ID (primary key).
VARIABLE_ID*	NUMBER	Unique ID for a variable within a workflow (primary key).
VARIABLE_NAME	VARCHAR2 (240)	Name of the variable.
VARIABLE_TYPE	NUMBER	Variable type. 0 = built in; 1 = user-defined.
VARIABLE_DESCRIPTION	VARCHAR2 (2000)	Comments on the variable.
VARIABLE_DATATYPE	NUMBER	Datatype of a workflow variable. 3 = decimal 4 = integer 5 = small integer 7 = real 8 = double 11 = date/time 12 = string
VARIABLE_DEFAULT_VALUE	VARCHAR2 (2000)	Default value of a variable.
LAST_SAVED	VARCHAR2 (30)	Date and time that this task was last saved.
TASK_INST_ID*	NUMBER	ID of the instance where the variable is defined.
TASK_INST_NAME	VARCHAR2 (240)	Name of the task instance.

Column Name	Datatype	Description
BIT_OPTIONS	NUMBER	Specifies whether the workflow variable is null or persistent. 1 = workflow variable is persistent; 2 = workflow variable is NULL.
VERSION_NUMBER*	NUMBER	Workflow version number.
**Indicates that the column is a key column.*		

REP_EVENT

This view displays the details of events created at the workflow or worklet level.

The following table lists event information in the REP_EVENT view:

Column Name	Datatype	Description
SUBJECT_ID*	NUMBER	Folder ID.
WORKFLOW_ID*	NUMBER	Workflow ID (primary key).
EVENT_ID*	NUMBER	Event ID (primary key).
EVENT_NAME	VARCHAR2 (30)	Name of the event.
EVENT_TYPE	NUMBER	Event type. 0 = built in; 1 = user-defined.
EVENT_SCOPE	NUMBER	Event scope.
EVENT_DESCRIPTION	VARCHAR2 (2000)	Event description.
LAST_SAVED	VARCHAR2 (30)	Date and time that this event was last saved.
VERSION_NUMBER*	NUMBER	Workflow version number.
**Indicates that the column is a key column.*		

REP_TASK_INST

This view displays all task instances within workflows and worklets.

The following table lists task instance information in the REP_TASK_INST view:

Column Name	Datatype	Description
WORKFLOW_ID*	NUMBER	Workflow ID (primary key).
INSTANCE_ID*	NUMBER	Instance ID (primary key).
TASK_ID*	NUMBER	Task ID.

Column Name	Datatype	Description
TASK_TYPE	NUMBER	Task type.
TASK_TYPE_NAME	VARCHAR2 (240)	Name of the object.
INSTANCE_NAME	VARCHAR2 (240)	Name of the instance.
IS_ENABLED	NUMBER	Specifies whether the task instance is enabled.
DESCRIPTION	VARCHAR2 (2000)	Description of the task.
IS_VALID	NUMBER	Specifies whether the task is valid. 0 = invalid; 1 = valid.
VERSION_NUMBER*	NUMBER	Workflow version number.
SERVER_ID*	NUMBER	Server ID associated with the workflow.
**Indicates that the column is a key column.*		

REP_WORKFLOW_DEP

This view shows how individual tasks and worklets are connected within a worklet or a workflow.

The following table lists task and worklet connection information in the REP_WORKFLOW_DEP view:

Column Name	Datatype	Description
WORKFLOW_ID*	NUMBER	Workflow ID.
FROM_INSTANCE_ID*	NUMBER	ID of the source task instance.
TO_INSTANCE_ID*	NUMBER	ID of the target task instance.
CONDITION_ID*	NUMBER	Condition ID.
VERSION_NUMBER*	NUMBER	Version number.
CONDITION	VARCHAR2 (2000)	The value that identifies the condition associated with the link.
**Indicates that the column is a key column.*		

REP_TASK_INST_RUN

This view displays the run statistics and folder reference for tasks within a workflow or worklet.

The following table lists run statistics and folder reference information in the REP_TASK_INST_RUN view:

Column Name	Datatype	Description
SUBJECT_AREA	VARCHAR2 (240)	Folder name.
WORKFLOW_NAME	VARCHAR2 (240)	Workflow name.
VERSION_NUMBER*	NUMBER	Version number.
SUBJECT_ID*	NUMBER	Folder ID.
WORKFLOW_ID*	NUMBER	ID of the parent workflow.
WORKFLOW_RUN_ID*	NUMBER	Run ID of the parent workflow.
WORKLET_RUN_ID*	NUMBER	Run ID of a worklet in a workflow.
CHILD_RUN_ID*	NUMBER	Run ID of a child task in a worklet.
INSTANCE_ID*	NUMBER	ID of an instance within a workflow or a worklet.
INSTANCE_NAME	VARCHAR2 (240)	Name of the task instance.
TASK_ID*	NUMBER	Task ID.
TASK_TYPE_NAME	VARCHAR2 (240)	Object name.
TASK_TYPE	NUMBER	Task type.
START_TIME	DATE	Start time configured for task execution.
END_TIME	DATE	End time configured for task execution.
RUN_ERR_CODE	NUMBER	Task error code.
RUN_ERR_MSG	VARCHAR2 (2000)	Task error message.
RUN_STATUS_CODE	NUMBER	Status code of the task. 1 = Succeeded 2 = Disabled 3 = Failed 4 = Stopped 5 = Aborted 6 = Running 15 = Terminated **Note:** MX views do not provide information about transient session and workflow status, such as suspending, stopping, scheduling, and aborting. You can view all statuses, including transient status, using *pmcmd* getservicedetails.
TASK_NAME	VARCHAR2 (240)	Task name.
TASK_VERSION_NUMBER*	NUMBER	Task version number.
SERVER_ID	NUMBER	ID of the Integration Service.

Column Name	Datatype	Description
SERVER_NAME	VARCHAR2 (240)	Name of the server.
**Indicates that the column is a key column.*		

REP_WFLOW_RUN

This view displays the run statistics for all workflows by folder.

The following table lists workflow run statistic information in the REP_WFLOW_RUN view:

Column Name	Datatype	Description
SUBJECT_ID*	NUMBER	Folder ID.
WORKFLOW_ID*	NUMBER	Workflow ID.
WORFLOW_RUN_ID*	NUMBER	Workflow run ID.
WORKFLOW_NAME	VARCHAR2 (240)	Workflow name.
SERVER_ID*	NUMBER	Integration Service ID.
SERVER_NAME	VARCHAR2 (240)	Integration Service name.
START_TIME	DATE	Start time configured for the workflow.
END_TIME	DATE	End time configured for the workflow.
LOG_FILE	VARCHAR2 (2000)	Full path and name of the log file.
RUN_ERR_CODE	NUMBER	Error message code.
RUN_ERR_MSG	VARCHAR2 (2000)	Error message.
RUN_STATUS_CODE	NUMBER	Status code of the task. 1 = Succeeded 2 = Disabled 3 = Failed 4 = Stopped 5 = Aborted 6 = Running 15 = Terminated **Note:** MX views do not provide information about transient session and workflow status, such as suspending, stopping, scheduling, and aborting. You can view all statuses, including transient status, using *pmcmd* getservicedetails.
USER_NAME	VARCHAR2 (240)	Name of the user who ran the workflow.

Column Name	Datatype	Description
RUN_TYPE	NUMBER	Specifies how the workflow was run. 1 = Scheduler 2 = User request 3 = Debug session 4 = Server initialization 5 = Remote task 6 = Remote debug session
VERSION_NUMBER*	NUMBER	Workflow version number.
SUBJECT_AREA	VARCHAR2 (240)	Folder name.
**Indicates that the column is a key column.*		

REP_LOAD_SESSIONS

This view provides information about sessions in the repository.

The following table lists session information in the REP_LOAD_SESSIONS view:

Column Name	Datatype	Description
SUBJECT_AREA*	VARCHAR2 (240)	Folder name.
SESSION_NAME	VARCHAR2 (240)	Name of the session.
LAST_SAVED	VARCHAR2 (240)	Time the session was last saved.
SESSION_ID*	NUMBER	Session ID.
MAPPING_NAME*	VARCHAR2 (240)	Name of the mapping this session uses.
VERSION_ID	NUMBER	Folder version ID.
VERSION_NAME	VARCHAR2 (240)	Folder version name.
IS_ACTIVE	NUMBER	Specifies whether the session is active.
STARTTIME	VARCHAR2 (240)	Session start time.
SESS_INTERVAL	NUMBER	Session interval.
REPEAT_COUNT	NUMBER	Repeat count.
SESSION_LOG_FILE	VARCHAR2 (240)	Session log file name.
BAD_FILE_LOCATION	VARCHAR2 (240)	Location of the reject file.
TARGET_ID	NUMBER	Target ID.
SOURCE_ID	NUMBER	Source ID.

Column Name	Datatype	Description
SESSION_VERSION_NUMBER	NUMBER	Version number of the session.
MAPPING_VERSION_NUMBER	NUMBER	Version number of the mapping.
SUBJECT_ID*	NUMBER	Folder ID.
IS_VALID	NUMBER	Specifies whether the session is valid or not. 0 = invalid; 1 = valid.
IS_REUSABLE	NUMBER	Specifies whether the session is reusable or not. 0 = not reusable; 1= reusable.
COMMENTS	VARCHAR2 (2000)	Description of the session.
MAPPING_ID	NUMBER	Sequence ID for the mapping associated with the session.
IS_IMPACTED	NUMBER	Specifies whether the session is impacted by a change to dependent objects that may require the session to be revalidated. 0 = not impacted; 1 = impacted.
**Indicates that the column is a key column.*		

REP_SESSION_CNXS

This view contains information about connections associated with reusable sessions.

The following table lists reusable session connection information in the REP_SESSION_CNXS view:

Column Name	Datatype	Description
SUBJECT_AREA*	VARCHAR2 (240)	Folder name in which the session is stored.
SUBJECT_ID*	NUMBER	Folder ID.
SESSION_NAME*	VARCHAR2 (240)	Name of the session.
SESSION_ID*	NUMBER	Session ID.
IS_TARGET	INTEGER	Specifies whether the connection is the target or the source. 0 = source connection; 1 = target connection; 22 = multi-group external procedure template extension; 25 = flat file lookup extension.
CONNECTION_NAME	VARCHAR2 (240)	Name of the connection.
CONNECTION_ID*	INTEGER	Connection ID.
SESSION_VERSION_NUMBER*	NUMBER	Version number of the session.
**Indicates that the column is a key column.*		

REP_SESSION_INSTANCES

This view contains connection information for session instances. If a session instance overrides the connection information in a reusable session, this view shows the connection in the session instance and the connection information in the reusable session. This view does not show connection information for reusable sessions that are not associated with any workflows.

The following table lists session instance connection information in the REP_SESSION_INSTANCES view:

Column Name	Datatype	Description
SUBJECT_AREA*	VARCHAR2 (240)	Folder name.
SUBJECT_ID*	NUMBER	Folder ID.
WORKFLOW_NAME*	VARCHAR2 (240)	Name of the workflow the session instance belongs to.
WORKFLOW_ID*	NUMBER	Workflow ID.
SESSION_INSTANCE_NAME*	VARCHAR2 (240)	Session instance name.
SESSION_INSTANCE_ID*	NUMBER	Session instance ID.
SESSION_ID*	NUMBER	Session ID.
IS_TARGET	INTEGER	Specifies the connection type. You can enter the following value for the applicable connection type: - **0.** Source connection - **1.** Target connection - **22.** Custom transformation - **25.** Lookup connection
CONNECTION_NAME	VARCHAR2 (240)	Name of the connection associated with the session instance.
CONNECTION_ID*	INTEGER	Connection ID associated with the session instance.
WORKFLOW_VERSION_NUMBER*	NUMBER	Workflow version number.
SESSION_VERSION_NUMBER*	NUMBER	Version number of the session.
**Indicates that the column is a key column.*		

REP_SESSION_FILES

This view contains file connections associated with reusable sessions.

The following table lists reusable session file connection information in the REP_SESSION_FILES view:

Column Name	Datatype	Description
SUBJECT_AREA*	VARCHAR2 (240)	Name of the folder containing the session.
SESSION_NAME*	VARCHAR2 (240)	Name of the session.

Column Name	Datatype	Description
IS_TARGET	INTEGER	Specifies the connection type. 1 = target file connection; 0 =source file connection.
FILE_NAME	VARCHAR2 (240)	Name of the source or target file.
DIR_NAME	VARCHAR2 (240)	Directory where the source or target file is stored.
CODE_PAGE	NUMBER	Code page associated with the source or target file. Values correspond to the code page IDs listed in the *Informatica Administrator Guide*.
SESSION_VERSION_NUMBER*	NUMBER	Session version number.
**Indicates that the column is a key column.*		

REP_SESSION_INST_FILES

This view contains file connection information for session instances associated with workflows. If a reusable session is not associated with a workflow, this view does not show file connection information for the session.

The following table lists session instance file connection information in the REP_SESSION_INST_FILES view:

Column Name	Datatype	Description
SUBJECT_AREA*	VARCHAR2 (240)	Name of the folder containing the session.
WORKFLOW_NAME*	VARCHAR2 (240)	Name of the workflow to which the session instance belongs.
WORKFLOW_VERSION_NUMBER*	NUMBER	Workflow version number.
SESSION_INSTANCE_NAME*	VARCHAR2 (240)	Name of the session instance.
IS_TARGET	INTEGER	Specifies the connection type. 1 = target file connection; 0 = source file connection.
FILE_NAME	VARCHAR2 (2000)	Name of the source or target file.
DIR_NAME	VARCHAR2 (2000)	Directory where the source or target file is stored.
CODE_PAGE	NUMBER	Code page associated with the source or target file. Values correspond to the code page IDs listed in the *Informatica Administrator Guide*.
**Indicates that the column is a key column.*		

REP_SESS_WIDGET_CNXS

This view contains information about the sources and targets used in a session. The reader and writer types and the connection name also display.

The following table lists connection information in the REP_SESS_WIDGET_CNXS view:

Column Name	Datatype	Description
WIDGET_INSTANCE_ID*	NUMBER	Instance ID of a source, target, or transformation.
WIDGET_TYPE	NUMBER	Identifies a source, target, or transformation.
INSTANCE_NAME	VARCHAR2 (240)	Instance name.
READER_WRITER_TYPE	VARCHAR2 (240)	Type of reader or writer used.
CNX_NAME	VARCHAR2 (240)	Connection name.
SESSION_ID*	NUMBER	Session ID.
SESSION_WIDG_INST_ID*	NUMBER	Transformation instance ID referenced by a session (primary key).
SESS_EXTN_OBJECT_TYPE	NUMBER	Indicates whether the object is a reader or a writer. 78 = reader; 79 = writer.
SESS_EXTN_OBJECT_SUBTYPE	NUMBER	Indicates a specific reader or writer.
SESS_CNX_REFS_OBJECT_TYPE	NUMBER	Type of referenced object.
SESS_CNX_REFS_OBJECT_SUBTYPE	NUMBER	Indicates a specific object.
SESS_CNX_REFS_OBJECT_ID*	NUMBER	ID of the referenced object.
WORKFLOW_ID*	NUMBER	Workflow ID.
SESSION_INSTANCE_ID*	NUMBER	Session instance ID.
SESSION_VERSION_NUMBER*	NUMBER	Session version number.
**Indicates that the column is a key column.*		

REP_COMPONENT

This view displays the list of tasks such as a command or an email for each session.

The following table lists session component information in the REP_COMPONENT view:

Column Name	Datatype	Description
WORKFLOW_ID*	NUMBER	ID of the workflow to which the session belongs.
TASK_ID*	NUMBER	Session ID.
TASK_INST_ID*	NUMBER	Session instance ID.
REF_OBJ_ID	NUMBER	ID of the referred object within a session.
REF_OBJ_TYPE	NUMBER	Referred object type.

Column Name	Datatype	Description
OBJECT_TYPE	NUMBER	Object type.
OBJECT_SEQ_TYPE	NUMBER	Identifies the referred object's sequence type.
VERSION_NUMBER*	NUMBER	Object version number.
PM_VALUE	VARCHAR2 (2000)	Component value.
VAL_NAME	VARCHAR2 (240)	Name of the value.
DESCRIPTION	VARCHAR2 (2000)	Description of the value.
**Indicates that the column is a key column.*		

REP_SESS_PARTITION_DEF

This view provides partition details of the sources, targets, and transformations in a session.

The following table lists partition information in the REP_SESS_PARTITION_DEF view:

Column Name	Datatype	Description
SESSION_ID*	NUMBER	Session ID.
SESS_WIDG_INST_ID*	NUMBER	Session instance ID.
PARTITION_ID*	NUMBER	Partition ID.
PARTITION_NAME	VARCHAR2 (240)	Partition name.
DESCRIPTION	VARCHAR2 (2000)	Description of the partition.
LAST_SAVED	VARCHAR2 (30)	Time when the partition was last modified.
VERSION_NUMBER	NUMBER	Session version number.
MAPPING_ID*	NUMBER	ID of the mapping used by the session.
WIDGET_ID*	NUMBER	ID of a source, target, or transformation in a session.
WIDGET_TYPE	NUMBER	Identifies a source, target, or transformation.
INSTANCE_ID*	NUMBER	Instance ID of a source, target, or transformation.
INSTANCE_NAME	VARCHAR2 (240)	Instance name.
TYPE_NAME	VARCHAR2 (240)	Object type name.
**Indicates that the column is a key column.*		

REP_SESS_CONFIG_PARM

This view displays session configuration parameter details. If the session overrides a parameter in the configured object, the view displays two rows. Select the row which contains the session ID reference.

The following table lists session configuration information in the REP_SESS_CONFIG_PARM view:

Column Name	Datatype	Description
SESSION_ID*	NUMBER	Session ID.
SESSION_VERSION_NUMBER*	NUMBER	Session version number.
CONFIG_ID*	NUMBER	Session configuration ID.
ATTR_ID*	NUMBER	Session configuration attribute ID.
ATTR_TYPE	NUMBER	Session configuration attribute type.
ATTR_NAME	VARCHAR2 (240)	Session configuration attribute name.
ATTR_VALUE	VARCHAR2 (2000)	Attribute value.
**Indicates that the column is a key column.*		

REP_SESS_INST_CONFIG_PARM

This view displays the attributes that are overwritten at the session instance.

The following table lists session instance configuration information in the REP_SESS_INST_CONFIG_PARM view:

Column Name	Datatype	Description
WORKFLOW_ID*	NUMBER	Workflow ID.
SESSION_ID*	NUMBER	Session ID.
SESSION_INST_ID*	NUMBER	Session instance ID.
WORKFLOW_VERSION_NUMBER*	NUMBER	Workflow version number.
CONFIG_ID*	NUMBER	Session configuration ID.
ATTR_ID*	NUMBER	Session configuration attribute ID.
ATTR_TYPE	NUMBER	Session configuration attribute type.
ATTR_NAME	VARCHAR2 (240)	Session configuration attribute name.
ATTR_VALUE	VARCHAR2 (2000)	Attribute value.
**Indicates that the column is a key column.*		

REP_TASK_ATTR

This view displays the attribute values and overridden values for session and workflow tasks.

The following table lists attribute information in the REP_TASK_ATTR view:

Column Name	Datatype	Description
WORKFLOW_ID*	NUMBER	Workflow ID.
INSTANCE_ID*	NUMBER	Task instance ID.
TASK_ID*	NUMBER	Task ID.
TASK_TYPE	NUMBER	Task type.
REF_SESSION_ID*	NUMBER	Session ID.
TASK_TYPE_NAME	VARCHAR2 (240)	Task type name.
ATTR_ID	NUMBER	Task attribute ID.
ATTR_NAME	VARCHAR2 (240)	Task attribute name.
ATTR_VALUE	VARCHAR2 (2000)	Attribute value.
LINE_NO	NUMBER	Line number of attribute values. Used for storing multiple lines of attribute values.
GROUP_ID*	NUMBER	Group ID.
VERSION_NUMBER*	NUMBER	Workflow version number if task attribute is overridden at workflow level. Session version number if task attribute is overridden at session level.
**Indicates that the column is a key column.*		

REP_SESS_LOG

This view provides log information about sessions. This view supplies the status of the last session, which might contain one or many target tables.

The following table lists session log information in the REP_SESS_LOG view:

Column Name	Datatype	Description
SUBJECT_AREA*	VARCHAR2 (240)	Folder name.
SUBJECT_ID*	NUMBER	Folder ID.
SESSION_NAME	VARCHAR2 (240)	Session name.
SESSION_ID*	NUMBER	Session ID.
SESSION_INSTANCE_NAME*	VARCHAR2 (240)	Session instance name.

Column Name	Datatype	Description
SUCCESSFUL_ROWS	NUMBER	Number of successfully loaded target rows.
FAILED_ROWS	NUMBER	Number of failed target rows.
SUCCESSFUL_SOURCE_ROWS	NUMBER	Number of successfully read source rows.
FAILED_SOURCE_ROWS	NUMBER	Number of failed source rows.
FIRST_ERROR_CODE	NUMBER	First error code.
FIRST_ERROR_MSG	VARCHAR2 (2000)	First error message.
LAST_ERROR_CODE	NUMBER	Last error code.
LAST_ERROR	VARCHAR2 (2000)	Last error message.
RUN_STATUS_CODE	NUMBER	Run status code. 1 = Suceeded 2 = Disabled 3 = Failed 4 = Stopped 5 = Aborted 6 = Running 7 = Suspending 8 = Suspended 9 = Stopping 10 = Aborting 11 = Waiting 12 = Scheduled 13 = Unscheduled 14 = Unknown 15 = Terminated **Note:** MX views may not provide up-to-the-minute information about transient session and workflow status, such as suspending, stopping, and aborting.
ACTUAL_START	DATE (DB SPECIFIC)	Actual time session started.
SESSION_TIMESTAMP	DATE (DB SPECIFIC)	Time completed.
SESSION_LOG_FILE	VARCHAR2 (2000)	Session log file name.
BAD_FILE_LOCATION	VARCHAR2 (4000)	Location of the reject file.
TASK_VERSION_NUMBER*	NUMBER	Version number of the task.
WORKFLOW_VERSION_NUMBER*	NUMBER	Workflow version number.
WORKFLOW_NAME	VARCHAR2 (240)	Name of the workflow that contains the session instance.
MAPPING_NAME	VARCHAR2 (240)	Mapping name.

Column Name	Datatype	Description
TOTAL_ERR	NUMBER	Total error code.
WORKFLOW_ID*	NUMBER	Workflow ID.
WORKFLOW_RUN_ID*	NUMBER	Workflow run ID.
WORKLET_RUN_ID	NUMBER	Run ID of a worklet in a workflow.
INSTANCE_ID	NUMBER	Instance ID.
**Indicates that a column is a key column.*		

REP_SESS_TBL_LOG

This view contains information about the status of an individual session run against a target. It provides the last update time, row counts, and error status based on a last update timestamp on a per target basis.

The following table lists individual session information in the REP_SESS_TBL_LOG view:

Column Name	Datatype	Description
SUBJECT_AREA*	VARCHAR2 (240)	Folder name.
SUBJECT_ID*	NUMBER	Folder ID.
SESSION_NAME*	VARCHAR2 (240)	Session name.
SESSION_ID*	NUMBER	Session ID.
SESSION_INSTANCE_NAME	VARCHAR2 (240)	Name of the session instance.
SESSION_INSTANCE_ID*	NUMBER	Session instance ID.
WORKFLOW_ID*	NUMBER	Workflow ID.
WORKFLOW_VERSION_NUMBER*	NUMBER	Workflow version number.
TABLE_NAME*	VARCHAR2 (240)	Name of the table for this log.
TABLE_ID*	NUMBER	Target table ID.
TABLE_VERSION_NUMBER*	NUMBER	Version number of the target.
TABLE_BUSNAME	VARCHAR2 (240)	Business name of the target.
TABLE_INSTANCE_NAME	VARCHAR2 (240)	Target instance name for the session.
SUCCESSFUL_ROWS	NUMBER	Number of successfully loaded target rows.
SUCCESSFUL_AFFECTED_ROWS	NUMBER	Number of affected target rows
FAILED_ROWS	NUMBER	Number of failed target rows.

Column Name	Datatype	Description
LAST_ERROR	VARCHAR2 (2000)	Last error message.
LAST_ERROR_CODE	NUMBER	Last error code.
START_TIME	DATE	Time the target load started.
END_TIME	DATE	Time the target load ended.
SESSION_TIMESTAMP	NUMBER	Session timestamp.
BAD_FILE_LOCATION	VARCHAR2 (4000)	Location of the reject file.
SESSION_VERSION_NUMBER*	NUMBER	Version number of the session.
PARTITION_NAME	VARCHAR2 (240)	Name of the partition.
MAPPLET_INSTANCE_NAME	VARCHAR2 (240)	Mapplet instance name.
WIDGET_NAME	VARCHAR2 (240)	Transformation name.
TYPE_NAME	VARCHAR2 (240)	Object name.
GROUP_NAME	VARCHAR2 (240)	Group name.
THROUGHPUT	NUMBER	Performance numbers for the target.
TYPE_ID	NUMBER	Object unique type ID.
**Indicates that the column is a key column.*		

Security Views

Security views allow you to see user information. The REP_USERS view provides a list of all PowerCenter users.

The following table lists user information in the REP_USERS view:

Column Name	Datatype	Description
USER_ID*	NUMBER	User ID (primary key).
NAME_SPACE	VARCHAR2 (240)	Security domain the user belongs to.
USER_NAME	VARCHAR2 (240)	User name.
STATUS	NUMBER	Not applicable. Reserved for future use.
**Indicates that the column is a key column.*		

Deployment Views

Deployment views allow you to see deployment information such as deployment groups, deployment date, source and target repository names associated with deployment, and objects which were deployed from one repository to another.

The following table lists the different views that help you analyze deployment metadata:

View	Description
REP_DEPLOY_GROUP	This view provides information about deployment groups in Change Management.
REP_DEPLOY_GROUP_DETAIL	This view provides Change Management deployment details.

REP_DEPLOY_GROUP

This view provides information about deployment groups.

The following table lists deployment group information in the REP_DEPLOY_GROUP view:

Column Name	Datatype	Description
DEP_GROUP_ID*	NUMBER	Deployment group ID.
DEP_GROUP_NAME	VARCHAR2 (240)	Deployment group name.
DESCRIPTION	VARCHAR2 (2000)	Description of the group.
CREATED_BY	VARCHAR2 (240)	Name of user who created the deployment group.
OWNER_ID*	NUMBER	User ID.
GROUP_ID*	NUMBER	Group ID.
CREATION_TIME	VARCHAR2 (30)	Creation time.
LAST_SAVED	VARCHAR2 (30)	Last saved time.
GROUP_TYPE	NUMBER	Deployment group type. 0 = static; 1 = dynamic.
QUERY_ID*	NUMBER	Query ID associated with a dynamic group.
QUERY_NAME	VARCHAR2 (240)	Query name associated with a dynamic group.
QUERY_DESCRIPTION	VARCHAR2 (2000)	Query description.
QUERY_CREATED_BY	VARCHAR2 (240)	Name of user who created the query.
QUERY_OWNER_ID	NUMBER	Query user.
QUERY_GROUP_ID	NUMBER	Query group ID.
QUERY_CREATION_TIME	VARCHAR2 (30)	Query creation time.

Column Name	Datatype	Description
QUERY_LAST_SAVED	VARCHAR2 (30)	Query last saved time.
QUERY_TYPE	NUMBER	Query type. 1 = public; 2 = personal.
**Indicates that the column is a key column.*		

REP_DEPLOY_GROUP_DETAIL

This view provides deployment details.

The following table lists deployment information in the REP_DEPLOY_GROUP_DETAIL view:

Column Name	Datatype	Description
DEP_RUN_ID	NUMBER	Unique deployment run ID.
OBJECT_ID*	NUMBER	Object ID.
OBJECT_NAME	VARCHAR2 (240)	Name of the object.
OBJECT_TYPE	NUMBER	Object type.
OBJECT_TYPE_NAME	VARCHAR2 (240)	Object type name.
SRC_VERSION_NUMBER	NUMBER	Object version number in the source repository.
TARG_VERSION_NUMBER	NUMBER	Object version number in the target repository.
SRC_SUBJECT_ID	NUMBER	Folder ID in the source repository.
TARG_SUBJECT_ID	NUMBER	Folder ID in the target repository.
SRC_SUBJECT_AREA	VARCHAR2 (240)	Folder name in the source repository.
TARG_SUBJECT_AREA	VARCHAR2 (240)	Folder name in the target repository.
IS_SHORTCUT	NUMBER	Specifies whether the object is a shortcut. 1 = shortcut; 0 = not a shortcut.
DEP_GROUP_ID	NUMBER	Deployment group ID.
DEP_GROUP_NAME	VARCHAR2 (240)	Deployment group name.
DEPLOY_TIME	NUMBER	Deployment start time.
DEPLOY_TYPE	NUMBER	Deployment type. 0 = invalid. 1 = deploy to. 2 = deploy from.
TARGET_REP_NAME	VARCHAR2 (240)	Target repository name.

Column Name	Datatype	Description
REP_GID	VARCHAR2 (240)	Global ID of the repository.
USER_ID	NUMBER	Deployment user ID.
GROUP_ID	NUMBER	Group ID.
USER_NAME	VARCHAR2 (240)	Deployment user name.
UTC_DEPLOY_TIME	NUMBER	UTC deployment time.
DEPLOY_STATUS	NUMBER	Deployment status. 0 = deployed. 1 = rollback. 2 = rollback failed.
ROLLBACK_TIME	VARCHAR2 (30)	Deployment rollback time.
**Indicates that the column is a key column.*		

Repository View

In the repository view you can see repository name, database type, connection information on which the repository is created, and whether the repository is local or global.

MX provides the REP_REPOSIT_INFO view to help you analyze repository metadata.

REP_REPOSIT_INFO

This view provides repository information such as repository name and type, domain name, and database type.

The following table lists repository information in the REP_REPOSIT_INFO view:

Column Name	Datatype	Description
RECID	NUMBER	Repository record ID.
REPOSITORY_NAME	VARCHAR2 (240)	Repository name.
REPOSITORY_DESCRIPTION	VARCHAR2 (2000)	Description of the repository.
REPOSITORY_ID*	NUMBER	Repository ID.
REPOSITORY_TYPE	NUMBER	Repository type. 1 = global. 2 = standalone. 3 = local.

Column Name	Datatype	Description
DOMAIN_NAME	VARCHAR2 (240)	Global domain name.
DATABASE_USER	VARCHAR2 (240)	Database user name used to connect to the repository.
DATABASE_TYPE	NUMBER	Repository type.
HOSTNAME	CHAR(3)	Returns value 'n/a'. The column refers to PowerCenter versions earlier than 8.0.
PORTNUM	CHAR(3)	Returns value 'n/a'. The column refers to PowerCenter versions earlier than 8.0.
**Indicates that the column is a key column.*		

Integration Service Views

The Integration Service views allow you to see information about Integration Service resources, such as the Integration Service name that can be used to run workflows in PowerCenter. The views allow you to see information about the grid, such as service locations, descriptions, and recent activity.

The following table lists the different views that help you analyze server resources and their access rights:

View	Description
REP_SERVER_INFO	This view is not used.
REP_SERVER_NET	This view provides information about Integration Service description, location, and usage.
REP_SERVER_NET_REF	This view provides information about Integration Service identification and usage.

REP_SERVER_NET

This view provides Integration Service grid information and provides description and usage information.

The following table lists Integration Service information in the REP_SERVER_NET view:

Column Name	Datatype	Description
SERVER_NET_ID*	NUMBER	Integration Service ID within the grid (primary key).
SERVER_NET_NAME	VARCHAR2 (240)	Integration Service name.
SERVER_NET_DESCRIPTION	VARCHAR2 (2000)	Description of the Integration Service.
LAST_SAVED	VARCHAR2 (30)	Time when object was last saved.
**Indicates that the column is a key column.*		

REP_SERVER_NET_REF

This view provides Integration Service identification within the grid and usage information.

The following table lists Integration Service identification and usage information within the grid in the REP_SERVER_NET_REF view:

Column Name	Datatype	Description
SERVER_NET_ID*	NUMBER	Integration Service ID within the grid (primary key).
SERVER_ID*	NUMBER	Integration Service ID (primary key).
BIT_OPTIONS	NUMBER	Integration Service job distribution options. 1 = use network; 3 = use network and used by network
**Indicates that the column is a key column.*		

Change Management Views

Change Management views allow you to see the version history of all objects in a PowerCenter repository and label metadata. Objects can be defined as tables, mappings, mapplets, transformations, sessions, workflows, worklets, and tasks. Labels can be defined on all objects.

The following table lists the different views that help you analyze version history of objects and label metadata:

View	Description
REP_VERSION_PROPS	Provides information about the version history of all objects in a PowerCenter repository.
REP_LABEL	Provides information about labels in Change Management.
REP_LABEL_REF	Provides information about label details in Change Management.

REP_VERSION_PROPS

This view provides the version history of all objects in a PowerCenter repository.

The following table lists label information in the REP_VERSION_PROPS view:

Column Name	Datatype	Description
OBJECT_ID*	NUMBER	Object ID.
OBJECT_TYPE*	NUMBER	Object type ID (primary key).
OBJECT_SUBTYPE	NUMBER	Object subtype ID.

Column Name	Datatype	Description
IS_SHORTCUT	NUMBER	Specifies whether the object is a shortcut. 1 = shortcut; 0 = not a shortcut.
VERSION_NUMBER	NUMBER	Object version number.
SUBJECT_ID	NUMBER	Folder ID.
USER_ID	NUMBER	User who last modified this version of the object.
OBJECT_NAME	VARCHAR2 (240)	Name of the object.
GROUP_NAME	VARCHAR2 (240)	Database name used by source objects.
LAST_SAVED	VARCHAR2 (30)	Time when object was last saved.
UTC_LAST_SAVED	NUMBER	UTC time when the object was last modified.
COMMENTS	VARCHAR2 (2000)	Description of the object.
SAVED_FROM	VARCHAR2(240)	Host machine name from which the version of the object is saved.
PURGED_BY_USERID	NUMBER	User ID who purged the object from the repository.
**Indicates that the column is a key column.*		

REP_LABEL

This view provides label information.

The following table lists label information in the REP_LABEL view:

Column Name	Datatype	Description
LABEL_ID*	NUMBER	Label ID.
LABEL_NAME	VARCHAR2 (240)	Label name.
DESCRIPTION	VARCHAR2 (2000)	Label description.
CREATED_BY	VARCHAR2 (240)	Name of user who created the label.
OWNER_ID	NUMBER	User ID.
GROUP_ID	NUMBER	Group ID.
CREATION_TIME	VARCHAR2 (30)	Label creation time.
LAST_SAVED	VARCHAR2 (30)	Time when label was last saved.
LABEL_TYPE	NUMBER	Label type. 2 = Can apply label to one version of the object.

Column Name	Datatype	Description
LABEL_STATUS	NUMBER	Label status. 1 = label unlocked; 2 = label locked.
**Indicates that the column is a key column.*		

REP_LABEL_REF

This view provides information about label details.

The following table lists label information in the REP_LABEL_REF view:

Column Name	Datatype	Description
LABEL_ID*	NUMBER	Label ID.
OBJECT_ID*	NUMBER	Object ID.
OBJECT_TYPE	NUMBER	Object type ID.
VERSION_NUMBER*	NUMBER	Object version number.
SUBJECT_ID*	NUMBER	Folder ID.
USER_ID*	NUMBER	User ID.
DESCRIPTION	VARCHAR2 (2000)	Label description.
APPLY_TIME	VARCHAR2 (30)	Time when label was applied to the object.
**Indicates that the column is a key column.*		

Folder View

In the folder view, you can see all the folders defined in the PowerCenter repository. It describes the name, ID, and description of each folder.

MX provides the REP_SUBJECT view to help you analyze folder metadata:

REP_SUBJECT

This view displays folder information such as folder name and description.

The following table lists folder information in the REP_SUBJECT view:

Column Name	Datatype	Description
SUBJECT_AREA	VARCHAR2 (240)	Folder name.
SUBJECT_ID*	NUMBER	Folder ID.
DESCRIPTION	VARCHAR2 (2000)	Folder description.
**Indicates that the column is a key column.*		

INDEX

D

E

F

G

I

K

L

M

N

O

P

Q

R

S