Informatica

Informatica® Multidomain MDM
10.4 HotFix 1

Repository Manager Guide

Informatica Multidomain MDM Repository Manager Guide
10.4 HotFix 1
September 2020

Publication Date: 2021-02-10

Table of Contents

Preface

Refer to the Informatica® *Multidomain MDM Repository Manager Guide* for information about the Repository Manager tool of the MDM Hub Console. Learn how to use the tool to validate metadata for repositories, copy objects from one repository to another, export repositories, and visualize repository schemas.

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

Introduction

This chapter includes the following topics:

Overview

This chapter introduces the Repository Manager tool and related concepts.

Note: This document assumes that you have read the *Multidomain MDM Overview Guide* and have a basic understanding of the MDM Hub architecture and key concepts.

About the Repository Manager Tool

The Repository Manager is the Hub Console tool that you use to:

- **Validate** metadata for a repository in your Informatica MDM Hub implementation.

  Validation verifies the completeness and integrity of the metadata that describes a repository. The validation process compares the logical model of a repository with its physical schema. If any issues arise, the Repository Manager generates a list of issues requiring attention, categorized by severity. For certain operations, Repository Manager requires a repository that is free of major problems.
- **Copy** design objects from one repository to another, in either of two ways:

Approach	Description
Promote	You can promote new design objects, or changes to existing objects (such as differences in attribute values), to a target repository. Promotion is used to copy incremental changes from one repository to another.
Import	You can import design objects from a design library into an empty target repository.

- **Export** a repository to a portable XML file that can then be imported or promoted into another repository, edited, or saved for archival purposes in a source control system.

You can use the Repository Manager to export an entire repository to a change list XML file, which can then be used to import design objects into a target repository or to save in a source control system for archival purposes.

- **Visualize** the schema using a graphical model view of the repository.

Related Topics:

- "Validating Metadata" on page 18
- "Promoting Changes Between Repositories" on page 28
- "Importing Design Objects" on page 48
- "Exporting Repositories" on page 55
- "Viewing the Schema in the Graphical Model View" on page 17
- "Getting Started with Repository Manager" on page 15

Metadata Management Concepts

This section introduces metadata management concepts that you need to understand in order to use the Repository Manager effectively.

Metadata

Metadata is data that is used to describe other data. In Informatica MDM Hub, metadata is used to describe the schema (data model) that is used in your Informatica MDM Hub implementation, along with related configuration settings.

Design Objects

In Informatica MDM Hub, *design objects* are metadata that are used to define the schema for an implementation. Design objects include base objects and columns, landing and staging tables, columns, indexes, relationships, mappings, cleanse functions, queries and packages, trust settings, validation and match rules, Security Access Manager definitions, Hierarchy Manager definitions, and other settings.

Related Topics:

- "Design Objects Supported in Repository Manager" on page 64

Dependencies

Repository Manager manages dependencies among design objects. For example, when you select a base object for promote or import, Repository Manager automatically selects all associated child objects—columns, match rules, and so on. Similarly, when you select a mapping, Repository Manager automatically selects the associated landing and staging tables. Repository Manager includes both direct and indirect dependencies so that all related design objects are selected. Repository Manager also flags dependency conflicts for design objects that you are trying to promote.

Related Topics:

- "Dependency Conflicts" on page 30

Conflicts When Copying Objects Between Repositories

Conflicts can arise when attempting to promote or import design objects between repositories. A conflict results from differences between two design objects with the same identification, such as two base objects with the same name. Certain conflicts can be resolved automatically, while others require human interpretation and manual intervention.

Related Topics:

- "Conflicts When Promoting Objects" on page 29

System Objects

Certain system objects that come predefined in a newly-created ORS—such as the Admin system, system cleanse functions, and the Hierarchy Manager RBO objects—are outside the set of design objects that are managed by the Repository Manager.

Repositories

The Repository Manager manages data stored in repositories.

Metadata in the Hub Store

Metadata is stored in two places in Informatica MDM Hub:

- The Master Database stores global metadata that is descriptive of the entire Informatica MDM Hub implementation.
- An Operational Reference Store (ORS, also known as a *repository*) contains metadata about its own schema and other configuration settings.

The Repository Manager works with metadata stored in repositories, not in the Master Database. Metadata stored in the Master Database—such as user accounts or message queue settings—is outside the scope of this document.

To learn more about the Hub Store and the schema for a Informatica MDM Hub implementation, see the *Multidomain MDM Configuration Guide* .

Source and Target Repositories

When copying metadata between repositories, there is always a source repository that contains the design object to copy, and the target repository that is destination for the design object.

When copying metadata between repositories, you explicitly identify—in the Repository Manager—which repository is the source and which is the target.

Change Lists

A *change list* is a list of changes to make to a target repository. A *change* is an operation in the change list—such as adding a base object or updating properties in a match rule—that is executed against the target repository.

Types of Change Lists

Repository Manager uses two kinds of change lists:

- A creation change list is the result of exporting the contents of a repository. Creation change lists represent an entire repository and are used as sources for import and promote operations.
- A comparison change list is the result of comparing the contents of two repositories and generating a list of changes to make to the target repository. Comparison change lists are used when promoting design objects.

Related Topics:

- "Exporting Repositories" on page 55

Change List XML Files

Change lists are stored in XML format with a *.change.xml extension. The generated XML file can be subsequently reviewed, edited, and applied to a target repository. Change list XML files can also be archived for configuration backup or stored in a source control system for configuration change management.

Related Topics:

- "Change List Reference" on page 68

Real-Time Metadata Management

External applications can manage metadata using the following Services Integration Framework (SIF) requests:

Task	Method	Description
Validation	validateMetadata	Validates the metadata for the specified repository.
Export	getOrsMetadata	Retrieves the metadata for the specified repository.
Change List Management	applyChangeList	Applies the specified change list to the specified target repository.
	createChangeList	Compares two repositories and creates a comparison change list XML file.
	validateChangeList	Runs a simulation of applying the specified change list to the specified target repository.

For more information, see the *Multidomain MDM Services Integration Framework Guide* and the *Multidomain MDM Javadoc*.

Considerations When Copying Metadata

When importing or promoting design objects, consider the following issues.

Migrating Large Repositories

To enhance performance when migrating large repositories in the Repository Manager, launch the Hub Console on a machine other than the host on which the application server is running. Ideally, both the application server machine and client machine would have 1GB (or more) of memory to handle migration

processing, which can be memory-intensive for large repositories. For Informatica MDM Hub system requirements, see the *Informatica MDM Hub Release Notes*.

Hierarchy Manager Requirements

Both the source and target repositories must be created in a Informatica MDM Hub environment with identical Hierarchy Manager licensing.

- You cannot copy changes between repositories if one repository was created in a Informatica MDM Hub environment *with* a Hierarchy Manager license if the other was created in a Informatica MDM Hub environment *without* a Hierarchy Manager license. The licensing for both environments must be identical —either both had a Hierarchy Manager license, or neither had a hierarchy license.
- To copy Hierarchy Manager metadata between repositories, both repositories must have been created in a Informatica MDM Hub environment with a Hierarchy Manager license.

In addition to the licensing requirement, Hierarchy Manager for the source and target repositories must be enabled. To enable Hierarchy Manager, open the Hierarchies tool in the Hub Console, select the repository, and follow the prompts to create the Repository Base Objects (RBO tables) and their associated queries and HM packages. For more information, see the *Multidomain MDM Configuration Guide* .

Note: Repository Manager does not promote, export, or import Repository Base Objects (RBO tables). Repository Manager displays a conflict if the source and target ORS databases have RBO tables created in different tablespaces. You should ensure that the source and target RBO tables are created with the same index and data space names.

Java Cleanse Adapters

Java cleanse adapters, which are stored on a file system, are not physically copied to the target repository—only the references to the Java cleanse libraries in the metadata are copied. However, cleanse adapters are dynamic libraries, and metadata can change—perhaps as the result of changes to an external configuration file, or because a user has access to a new feature in the cleanse adapter.

The import and promote processes assume that the source and target environments have the same cleanse adapters configured.

User Exits

User exits that are stored in the database, are not physically copied to the target repository.

User Information

User information is not copied. This includes:

- user accounts (Users tool)
- user groups and user account memberships (Users and Groups tool)
- user account assignments to databases (Users and Groups tool)
- user/role assignments (Roles tool)

For more information, see the *Multidomain MDM Configuration Guide* .

Populations for Matching

If your Informatica MDM Hub implementation uses one or more non-US populations for matching, all required populations must be enabled for both the source and target repositories before attempting to copy design objects between them. For more information, see the *Multidomain MDM Configuration Guide* .

Repositories Currently Registered with a Proxy User

If an ORS is currently registered with a proxy user, the ORS schema owner password is required when applying changes in a changelist (either via promotion or import).

In these situations, the Repository Manager prompts you to provide the owner password before proceeding with the requested operation.

Monitoring the Results of Changes

When you import design objects or apply promotion changes to a target repository, Repository Manager stores the results of change list execution in the following log tables in the Hub Store:

Table in Hub Store	Description
C_REPOS_MET_CHANGE_EXEC	Log of each change list execution, including the resulting execution status code and error description, if applicable.
C_REPOS_MET_CHANGE_EXEC_ITEM	Log of each individual action that was executed in the change list, including the item involved and the action that was taken on that item. Child table of C_REPOS_MET_CHANGE_EXEC.

If an error occurs during change list execution, the process terminates and reports the last problem encountered. Fix the problem and apply the change list again.

CHAPTER 2

Getting Started with Repository Manager

This chapter includes the following topics:

Overview

This chapter describes how to start and navigate the Repository Manager tool in the Hub Console.

Starting Repository Manager

To start the Repository Manager:

1. Start the Hub Console according to the instructions in “Getting Started With the Hub Console” as described in the *Multidomain MDM Configuration Guide* .
2. In the Hub Console, connect to the Master Database.
3. In the Hub Console, expand the Configuration workbench, and then click **Repository Manager**.

   The Hub Console displays the Repository Manager tool.

Repository Manager Interface Components

This section describes Repository Manager interface components.

Tabs

The Repository Manager tool has the following tabs:

Tab	Description
Validate	Used to validate metadata for a *repository* in your Informatica MDM Hub implementation. The *metadata validation process* is designed to verify the completeness and integrity of the metadata that describes a repository.
Promote	Used to promote design objects between repositories.
Import	Used to import design objects into an empty repository.
Export	Used to export a repository.

Related Topics:

- "Validating Metadata" on page 18
- "Promoting Changes Between Repositories" on page 28
- "Importing Design Objects" on page 48
- "Exporting Repositories" on page 55

Command Buttons

Command buttons are used to execute applicable operations on the current tab. For a list of buttons on each tab, see:

Tab	Description
Validate	"Command Buttons on the Validation Tab" on page 20
Promote	"Command Buttons on the Visual Tab" on page 33 "Command Buttons on the Change List Tab" on page 41
Import	"Command Buttons on the Import Tab" on page 50
Export	"Command Buttons on the Export Tab" on page 56

Repository Lists

All Repository Manager tabs have lists of the repositories (ORS databases) that are defined in the Master Database for this Informatica MDM Hub implementation. If the validation process was previously run on an ORS, an icon next to the ORS name indicates whether the repository has been validated and, if so, whether the most recent validation succeeded or failed.

Related Topics:

- "Validation Indicator" on page 20

Navigating the Repository Manager

This section describes how to navigate the Repository Manager.

Automatic Exclusive Locking

Before applying changes to a target repository during promote or import, the Repository Manager automatically releases locks that other users have on the target repository, then places an exclusive lock on the target repository while changes are being applied. An *exclusive lock* prevents any other users from making changes to the target repository in the Hub Console while its metadata is being modified.

After the changes are complete, Repository Manager automatically releases the exclusive lock on the target repository. For more information about locks in the Hub Console, see "Getting Started with the Hub Console" in the *Multidomain MDM Configuration Guide* .

Note: Write locks cannot be obtained on an ORS that is in production mode—when the Production Mode check box is selected for this ORS in the database properties tab in the Databases tool. For more information, see "Configuring Operational Record Stores and Datasources" in the *Multidomain MDM Configuration Guide* .

Viewing the Schema in the Graphical Model View

You can use the Schema Viewer tool, accessible from different tabs in the Repository Manager, to display a graphical view of the data model for any repository—whether an ORS or loaded from an XML file. The Schema Viewer is particularly helpful for visualizing a complex schema, graphically showing base objects and foreign-key relationships in the schema. The Schema Viewer gives you the ability to visually examine an existing schema before applying changes to it or exporting its metadata to a change list.

To launch the Schema Viewer:

▶ On the Validate, Promote, or Export tab, click the **Schema Viewer** button.

The Hub Console displays the Schema Viewer.

For detailed instructions on using the tool, see the *Multidomain MDM Configuration Guide* .

CHAPTER 3

Validating Metadata

This chapter includes the following topics:

Overview

This chapter describes how to use the Repository Manager in the Hub Console to validate metadata for a repository in your Informatica MDM Hub implementation.

About the Metadata Validation Process

The *metadata validation process* is designed to verify the completeness and integrity of the metadata that describes a repository.

Logical Model and Physical Schema Should Match

In an ORS, its metadata (logical model) should match its physical schema (tables and columns) exactly. For every metadata definition, there should be a corresponding physical component, and for every physical component there should be a corresponding metadata definition.

Certain events, however, can cause discrepancies between the metadata and physical schema. For example, a database administrator might make changes directly to the database instead of using the Schema Manager in the Hub Console. Similarly, database corruption might result from a power outage or hardware failure. The Repository Manager can help determine whether a discrepancy is inconsequential to Informatica MDM Hub operations, or whether it can cause major disruptions.

Metadata Validation Process

The metadata validation process performs the following tasks:

- determines whether all expected metadata tables are in the schema and that they have the expected signature (column data type, precision, and scale)
- determines whether the physical schema is synchronized with the metadata
- checks just the metadata to determine whether it is internally consistent
- generates a list of issues, if any, that merit attention

Note: For promotion, import, and export operations, Repository Manager requires a repository that has been validated and free of major problems.

Related Topics:

- "Issue Severity Levels" on page 20

Scope of Metadata Validation

The metadata validation process performs the following kinds of checks on the repository:

Type of Check	Description
system	Checks all repository tables (C_REPOS_*), columns, and views at the system level. Includes constraints (primary keys, foreign keys, and indexes), sequences, and triggers (signature only).
physical	Compares repository structure with database's metadata. For example, for base object tables, the repository metadata must match the Oracle physical metadata; tables, constraints, views, and sequences.
repository	Metadata stored in the repository.

Note: The Repository Manager allows you to narrow the scope of validation.

In addition, Repository Manager verifies that the version of the ORS matches the installed Informatica MDM Hub software. Validation cannot proceed if the versions do not match.

Note: To validate design objects associated with the Informatica MDM Hub Services Integration Framework (SIF), use the SIF Manager tool in the Hub Console instead.

Related Topics:

- "Design Objects Supported in Repository Manager" on page 64
- "Validating Metadata" on page 23
- "Issue Severity Levels" on page 20
- "Showing the Validation History" on page 25

Issue Severity Levels

During the validation process, the Repository Manager assigns a severity level to each issue. The issues are displayed in the Issues Found pane, under the issue severity category tabs described in the following table:

Severity	Description
Information	Information-only message that warrants attention. Example: A package does not contain all records from the underlying table.
Warnings	Inconsistency that does not have a harmful effect on Informatica MDM Hub operations.
Errors	Problem that can prevent normal promote, import, and export operations from completing successfully.
FATAL	Repository cannot be loaded or severe error that could lead to others errors if not remedied immediately. Example: unknown data type.

For promotion, import, and export operations, Repository Manager requires a repository that has no Errors or FATAL issues.

Validation Indicator

Throughout the Hub Console, an icon next to an ORS indicates whether it has been validated and, if so, whether the most recent validation resulted in issues.

Image	Meaning
	Unknown. ORS has not been validated since it was initially created, or since the last time the repository was changed. If anything in the ORS metadata is changed, its validation indicator reverts to Unknown.
	ORS has been validated with no issues. No change has been made to the repository since the validation process was made.
	ORS has been validated with warnings.
	ORS has failed validation and errors were found.

Command Buttons on the Validation Tab

The Validation tab has the following command buttons.

Button	Description
	Validates the selected repository.
	Saves the validation results for the selected repository to an HTML file.
	Displays the validation history for the selected repository.

Button	Description
	Start the Schema Viewer for the selected repository.
	Shows reparable metadata errors in the Issues Found pane.
	Stops filtering reparables in the Issues Found pane.
	Expands all nodes in the Issues Found pane.
	Collapses all nodes in the Issues Found pane.
	Repairs the selected reparable metadata error.
	Filters out and hides metadata errors that do not match search criteria specified in the search filter field.
	Shows all metadata errors. The metadata errors that match search criteria are highlighted.
	Clears the search filter field.

Related Topics:

- "Validating Metadata" on page 23
- "Saving the Validation Results" on page 25
- "Viewing the Schema in the Graphical Model View" on page 17

Operational Reference Store Metadata Validation

You can run use the Repository Manager tool in the Hub Console to validate Operational Reference Store (ORS) metadata.

You must validate ORS metadata and resolve all errors and fatal issues before you perform a promotion operation, import operation, or an export operation. You must also validate the metadata and resolve all errors and fatal issues before you upgrade.

By default, all validation checks are enabled, but you can choose to validate specific areas of the repository.

The following table lists the validation checks that you can select:

Validation Check	Description
All System Checks	Performs the following systems checks: - repository tables - system columns - system views - primary key constraints - foreign key constraints - constraints on indexes - sequences - trigger signatures - packages
All Physical Checks	Compares repository structure with database metadata. For example, for base object tables, the repository metadata must match the physical database metadata for tables, constraints, views, and sequences.
All Repository Checks	Validates metadata stored in the repository

The following table describes the individual repository checks you can select:

Validation Check	Description
Repository	Validates the repository
Mappings	Validates mappings between landing and staging tables
Cleanse	Validates Process Servers, cleanse functions, and cleanse lists
Queries	Validates queries
Schema	Validates the following data model design objects: - base object tables - base object columns - landing tables - match configuration - external match tables - validation rules - message queues - relationships - staging tables - custom indexes
Trust	Validates the source system trust configuration and trust columns
Security Access Manager	Validates resource groups, secure resources, and roles
Hierarchy Manager	Validates entity types, relationship types, hierarchies, packages, profiles, sandboxes, and Hierarchy Manager metadata
Miscellaneous	Validates all other supported design objects

Validating Metadata

To validate the metadata of an Operational Reference Store (ORS) repository, use the Repository Manager tool in the Hub Console.

1. From the **Configuration** workbench in the Hub Console, select the **Repository Manager** tool.
2. From the **Repository Manager** tool, select the **Validate** tab.
3. From the **Select the repository to validate** list, select a repository.
4. Click the **Validate** button.
5. From the **Select Validation Checks** dialog box, select the validation checks to perform. Click **OK**.

   The Repository Manager tool validates the repository and displays any issues in the **Issues Found** pane.
6. Click the **Repair** button to fix repairable issues.
7. If the ORS remains in the **Unknown** state, synchronize the system clocks of the application server and the database machine.

Related Topics:

- "Issue Severity Levels" on page 20
- "Starting Repository Manager" on page 15
- "Scope of Metadata Validation" on page 19
- "Design Objects Supported in Repository Manager" on page 64

Information Pane

In the Information pane, Repository Manager displays the following information:

Field	Description
Version	Informatica MDM Hub version. The repository version must match the version of the installed Informatica MDM Hub software.
Date	Date/time when the validation process was started.
Validation Scope	Scope of validation. Determined by whether all options were selected in the Select Validation Checks (Complete) or at least one option was not selected (Partial).
Summary	Summary of validation results.

Properties Pane

In the properties pane, the Repository Manager displays a scrollable report of the issues found. This report contains the following columns:

Column	Description
#	Sequential number of the issue.
Severity	Severity of the issue.
Message	Description of the issue, including: - error code. Example: **SIP-PV-10312** - diagnostic text. Example: **Index 'SVR1_AF9' of table 'C_RBO_HIERARCHY_XREF' is in the metadata but not in the database.** **Note:** If you encounter an SIP-PV-10000 issue, the Repository Manager could not load the metadata (for example, there is corruption in the database), and therefore repository checks were not performed. This error means that a system check failed.

Prior Validation Results for Imported Schemas

If an Oracle export (dump) file is created for an ORS that has undergone the validation process, and if the dump file contains prior validation results, then if the dump file is subsequently imported into a new ORS (when running setup.sql or setup_ors.sql according to the instructions in the *Multidomain MDM Installation Guide*, the new ORS will contain the prior validation results from the exported dump file.

When you validate the new ORS, Repository Manager appends the new results in two system tables (C_REPOS_MET_VALID_MSG and C_REPOS_MET_VALID_RESULT). If you click the History button, you can see all past validation results—including those from the imported dump file.

Related Topics:

- "Showing the Validation History" on page 25

Filtering Issues

To filter metadata validation results (if issues are found):

1. Run the metadata validation process.

   The metadata errors are displayed in the Issues Found pane under the FATAL, Errors, Warnings, and Information issues category tabs.

2. Click an issues tab to select an issue category such as FATAL, Errors, Warnings, or Information.

   The issues found for the selected issue category is displayed.

3. Enter a string filter criterion in the filter field for filtering validation error messages for the selected issue category.

   The filtered error messages are highlighted in the Issues Found pane.

4. Click the **Hide** button to hide the error messages that did not match the filter criterion.

You can use the Hide button and the Stop filtering reparables or Show reparables only buttons at the same time. If the two buttons are clicked at the same time, they act as an AND operation.

Related Topics:

- "Validating Metadata" on page 23

Saving the Validation Results

After you run the validation process, you can save the validation results as an HTML file.

1. From the **Repository Manager** tool in the **Hub Console**, select the **Validate** tab.
2. Click the **Save** button.
3. From the **Save** dialog box, navigate to the directory where you want to save the validation results.
4. Specify a descriptive file name for the HTML file. Click **Save**.

   The Repository Manager saves the validation results as an HTML file in the specified location.

Related Topics:

- "Validating Metadata" on page 23

Showing the Validation History

To show the validation history:

1. Run the metadata validation process.
2. Click the **History** button.

   The Repository Manager displays the Validation History window.

   Each row in the Validation History window represents the results of one execution of the validation process. The Validation History window displays the following columns.

Column	Description
Date	Date and time when the validation process was run.
User	User who initiated the validation process.
# Fatal	Number of issues encountered during validation with a Fatal severity level.
# Error	Number of issues encountered during validation with an Error severity level.
# Warning	Number of issues encountered during validation with a Warning severity level.
# Information	Number of issues encountered during validation with an Information severity level.
Scope	Specifies whether the scope of validation was Complete or Partial.

Related Topics:

- "Scope of Metadata Validation" on page 19
- "Validating Metadata" on page 23
- "Issue Severity Levels" on page 20

Viewing Validation Results

To view validation results from the Validation History window:

1. In the Validation History window, select the validation result that you want to view.
2. Click the **View** button.

   The Repository Manager displays the View Validation Result window.
3. If you want, you can:
   - Filter results.
   - Click the **Save** button to save the results.
   - Select an issue in the list and click the **Recommendations** button to see recommendations for that issue.
4. Click **Close**.

Related Topics:

- "Saving the Validation Results" on page 25

Deleting Entries in the Validation History Log

To delete a validation result from the Validation History window:

1. In the Validation History window, select the validation result that you want to view.
2. Click the **Delete** button.

   The Repository Manager prompts you to confirm deletion.
3. Click **Yes**.
4. Click **Close**.

Metadata Repair Process

This section describes how to repair metadata for a repository (ORS) in your Informatica MDM Hub implementation.

The MDM Hub provides an internal framework for automatic repair of some validation errors reported by the Repository Manager. You can perform a batch or selective repair of reparable metadata validation errors. A metadata error can prevent validation of metadata dependent on it and can result in hidden metadata validation errors. The hidden metadata validation errors are revealed after the metadata errors related to them are repaired.

Note:

- Only metadata validation errors with the repair icon can be repaired.
- The Repair feature is available only to users with the administrator role.

Metadata Repair Results

When you repair metadata for a repository, Repository Manager appends the new results to the C_REPOS_MET_VALID_MSG system table.

The results of the metadata repair are stored in the following columns of the C_REPOS_MET_VALID_MSG system table:

- REPAIR_IND
- REPAIR_FAIL_MESSAGE

Repairing Metadata in a Repository

To repair metadata errors, you must use the following procedure:

1. Validate the metadata.

   If issues are found, then they are displayed in the Issues Found pane. The reparable errors are displayed with a repair icon under the FATAL, Errors, Warnings, and Information issues category tabs.
2. Click an issues tab to select an issue category such as FATAL, Errors, Warnings, or Information.

   The issues found for the selected issue category is displayed.
3. Click the **Show reparables only** button, to view reparable metadata errors.

   For help with metadata validation errors that cannot be repaired, contact Informatica Global Customer Support.
4. Select an error from the Issues Found pane.
   - For a batch repair of reparable validation errors, select the validation error code.
   - For repair of selective reparable validation errors, select one error at a time from the validation errors listed under a validation error code.
5. Click the **Repair** button.

   A repair dialog appears with a warning of the metadata repair operation that will be executed.
6. If you need to perform the prompted metadata repair operation, then click **OK**.

   The metadata is repaired automatically.
   If you do not want to perform the prompted metadata repair operation, then click **Cancel**.

   If the repair is successful, then the repaired validation errors are marked as repaired with a distinctive icon.
7. If the repair fails, then you can click the **Repair** button to try again.

   Note: A repair may fail because it may be blocked by other issues. You need to find and fix the issues blocking the repair and then try again.

Related Topics:

- "Validating Metadata" on page 23

CHAPTER 4

Promoting Changes Between Repositories

This chapter includes the following topics:

Promoting Changes Between Repositories Overview

You can use the Repository Manager in the Hub Console to move incremental changes from one repository to another in your MDM Hub implementation.

In the Repository Manager, the Promote tab allows you promote changes between repositories. Promotion copies incremental changes from one repository to another. Incremental changes can involve the following types of operation:

The insertion of new design objects

To insert new design objects into an empty target repository, you can use the import process according to the instructions in Chapter 5, "Importing Design Objects" on page 48.

The update of metadata for identically named design objects in the source and target repositories

If the target repository already contains design objects, however, then you must promote changes between repositories. Through the Repository Manager, you can promote changes, such as differences in attribute values, to a design object from the source repository into the target repository. For example, you could update a Party base object in a production repository with changes from a development repository. Changes might include differences in base object property settings, column definitions, or mappings.

You can choose whether the Repository Manager performs data integrity validation before or after the promotion of changes. Data integrity validation checks unique constraints and foreign keys.

Note: You cannot promote changes between repositories across heterogeneous databases such as between an Oracle repository and an IBM Db2 repository.

Promotion Scenarios

This section describes common scenarios in which promotion is used.

Synchronize Promotion

Synchronize promotion is used to synchronize design objects from one repository with another, such as between development and test repositories, or between test and production repositories.

Synchronize promotion can also be used to create an identical repository. In this scenario, synchronize promotion uses an intermediary change list XML file. The change list XML file contains a list of changes to apply to the target repository. All changes in the change list XML file are applied to the target repository. This scenario uses the Change List tab in the Repository Manager.

Related Topics:

- "Change Lists" on page 11
- "Promoting Changes Using Change Lists" on page 40

Selective Promotion

In a distributed implementation environment, developers can use the Repository Manager tool to share and re-use selected design objects across separate but parallel implementation repositories. For example, developers can use separate repositories for data modeling, SQL coding, and application integration.

The list of selected changes can be saved in a comparison change list XML file before being applied so that changes can be reviewed, edited, and approved beforehand. Approved changes can be propagated to a central, master repository. This scenario uses the Visual tab in the Repository Manager.

Related Topics:

- "Promoting Changes Visually" on page 32

Differences Between Selective and Synchronize Promotion

Selective promotion is more granular than synchronize promotion, and it allows you to select individual design objects to promote. Synchronize promotion requires that you apply the entire change list XML to the target repository. However, you can edit the change list XML file beforehand to edit or remove changes as needed.

Design Objects That Can Be Promoted

For a complete list of design objects that can be promoted, see "Design Objects Supported in Repository Manager" on page 64.

Note: Custom indexes created on supporting tables are not promoted to the target environment during migration.

Conflicts When Promoting Objects

Conflicts can arise when attempting to promote objects between repositories.

Property Conflicts

Property conflicts arise during the promotion process when the same design object in both the source and target repositories has different property values. For example, suppose you were trying to promote changes in a Customer base object in which the Enable History property differed between the source and target base objects.

In this case, you could have Repository Manager compare the repositories and create a conflicts list. For each property conflict, Repository Manager will highlight the conflict and prompt you to take the appropriate action, such as deciding which value to keep—the existing property in the target repository, or the property in the base object that you are trying to promote. Repository Manager displays a properties panel that shows a side-by-side comparison of property values.

Related Topics:

- "Properties Panel for the Selected Design Object" on page 36

Dependency Conflicts

Dependency conflicts arise during the promotion process when the source and target design objects have different collections of child design objects.

For example, if you try to promote changes in a Person base object, and the column definitions differ between the source and the target repository, dependency conflicts can arise.

In this case, Repository Manager can analyze the two repositories and create a conflict list. When you attempt to promote changes, Repository Manager flags dependency conflicts and for each conflict, prompt you to take the appropriate action.

Related Topics:

- "Design Object Dependencies" on page 66
- "Conflicts When Copying Objects Between Repositories" on page 11

Actions to Resolve Conflicts

After you initiate the process of applying changes to the target repository, Repository Manager prompts you to specify how to handle property and dependency conflicts if they are encountered.

You can select one of the following actions:

- **Merge**
  - Manually merge the two design objects, creating a combination of both the source and target design objects

    Note: A manual merge is useful when you want the target object to contain some property values from the source and others from the target. For example, if you were promoting a base object from a development environment to a production environment, you might want to exclude certain property settings from promotion (such as the batch size) that could affected performance in the production environment.
  - Keep the source design object and discard the target design object
  - Keep the target design object and discard the source design object
- **Replace**
  - Replace the target design object completely with the source design object (overwrite the target)

Note:

- The Replace option applies to both property and object conflicts, while the Merge options apply to property conflicts only.
- Choose the **Replace the target objects completely with source objects** option to resolve conflicts when promoting *deleted* objects (such as base objects, staging tables, columns, etc.).

Repository Manager also flags related design objects for more complex dependencies, such as mappings that depend on columns in landing and staging tables.

Related Topics:

- "Applying Changes to the Target Repository" on page 39
- "Applying a Change List to the Target Repository" on page 45

Considerations for the Promotion Process

Before you promote changes, consider the following information:

- Before you promote changes, consider making a back up copy of the target repository.
- Promotion involves changes to the target repository only. The source repository remains unchanged.
- For promote operations, the Repository Manager requires a repository that has been validated and free of errors, or fatal issues.
- The `deleted_[*]` columns that are defined as user columns, instead of system columns, can be promoted like any other user column.
- You cannot reduce column lengths in the promotion process. For example, if you have a varchar(50) column in both the source and target repository, the Repository Manager generates an error when you reduce the column to varchar(20) in the source repository and then try to promote the change to the target repository.
- The ROWID_SYSTEM values must match in the source and target repositories. If these values do not match, before attempting promotion operations, you must synchronize system rowids in C_REPOS_SYSTEM for the source and target repositories. Use the rowids from the source repository to update the target repository so that the source and target have the same rowids for the same systems.
- Before you promote changes between repositories, ensure that both the source and target repositories have sufficient privileges to all required tablespaces.
- The Repository Manager supports the migration of design objects in an ORS but not in the Master Database. This includes user accounts, user groups, user account assignments to databases, and user or role assignments. On the target schema, you must manually synchronize user information.
- If you promote a package that is based on a custom query, the Repository Manager cannot guarantee the correctness of the custom query on which it is based. If the package is not valid after promotion, you must save the package again through the Packages tool in the Hub Console.
- If your custom query uses a custom table, you cannot promote the custom table. Instead, after migrating the custom query, you must create the custom table on the target repository.
- When you promote the saved queries associated with business entities between repositories, the Repository Manager promotes the queries only. The user information associated with the queries are not promoted to the target repository. After the promotion, you must reassign users to the saved queries in the target repository.
- In an environment with Elasticsearch, when you promote changes to the searchable properties of a field after you index your data, the indexes are deleted. You must run the Initially Index Smart Search Data batch job to reindex the data.
- When you promote match rule changes between repositories, the Repository Manager generates a change warning to indicate that you might have to reset the associated match tables.
- The Repository Manager might flag the attempted promotion of invalid configurations. For example, if trust settings are configured for a column in the target repository, but trust is not enabled for that column in the source repository, the Repository Manager flags this as invalid. When you apply the changelist, this object does not promote. If you promote visually, you can use the **Replace the target objects completely with source objects** option to promote it anyway.

- To ensure a date default promotion is successful, set default date values for Date columns. Informatica recommends to use cleanse functions to assign default values during the stage process.
- To migrate a unique column, you must configure the column to have a default value. For more information, see the *Multidomain MDM Configuration Guide*.
- When Production Mode is enabled for an ORS, you must enable Transition Mode, which allows you to run the Repository Manager Promote actions. For more information, see the *Multidomain MDM Configuration Guide*.
- To ensure any promotion is successful in an Oracle repository, use a Unicode-enabled database. To verify the character set of the database, enter the following command:

```
SQL> select * from v$nls_parameters where parameter Like '%CHARACTERSET';
```

For additional considerations, such as promoting Hierarchy Manager metadata, see "Considerations When Copying Metadata" on page 12.

Related Topics:

- "Issue Severity Levels" on page 20
- "Considerations When Copying Metadata" on page 12
- "Exporting a Repository" on page 57

Reducing the Column Length

If you want to reduce the length of a column, you must drop and re-create the column through changelists.

1. In the source repository, drop the column that is too long.
2. Create the changelist.
3. Apply the changelist to the target repository.
4. In the source repository, add the column with a shorter length.
5. Create the changelist.
6. Apply the changelist to the target repository.

Promoting Changes Visually

This section describes how to promote visually between repositories.

Related Topics:

- "Selective Promotion" on page 29

Overview of Visual Promotion Tasks

Note: Before you promote changes, make a back up copy of the target repository.

To promote changes visually:

1. Navigate to the **Promote / Visual** tab.
2. Select the source repository.

3. Select the target repository.
4. Select and promote the design objects that you want to promote from the source repository, resolving conflicts as needed.
5. Optionally, you can save proposed changes to a change list XML file.
6. Apply the changes to the target repository.

Related Topics:

- "Exporting a Repository" on page 57
- "Navigating to the Promote / Visual Tab" on page 33
- "Selecting the Target Repository for Visual Promotion" on page 34
- "Visually Promoting Changes to the Target Repository" on page 36
- "Saving Changes in a Comparison Change List File" on page 39
- "Applying Changes to the Target Repository" on page 39

Navigating to the Promote / Visual Tab

To promote changes visually:

1. Start the Repository Manager tool.
2. Click the **Promote** tab.

Complete the remaining tasks in this section.

Related Topics:

- "Starting Repository Manager" on page 15

Command Buttons on the Visual Tab

The Visual tab has the following command buttons.

Button	Description
	Promote the selected design object in the source repository to the target repository.
	View the design object hierarchy with markups.
	Go to the previous conflict in the design object hierarchy.
	Go to the next conflict in the design object hierarchy.
	Save as a change list.

Button	Description
	Apply the changes to the target repository.
	Start the Schema Viewer for the selected repository.

Selecting the Source Repository for Visual Promotion

For the source repository, you can choose either a database (ORS) or a creation change list XML file.

To select the source repository for promotion:

1. Click the **Select** button next to the Source drop-down list.

   The Repository Manager displays the Open Repository window.
2. Select a source repository in the list.
   - For a database repository, select one from the list.
     If you selected a database repository that has not yet been validated, click the **Validate** button and complete the validation process.

     Note: Repository Manager allows you to use a repository only after it has been validated and found to be free of Errors or FATAL issues.

     Click **OK**.

     Repository Manager loads the source repository.
   - For a change list XML file, click the **File Repository** tab.
     Click the **Open** button.

     Select the change list XML file (navigate to the folder if needed) and click **Open**.

     Note: You must select a creation change list that represents a complete schema as a basis of comparison. Repository Manager does not allow you to choose a comparison change list.

     Repository Manager loads and validates the repository from the selected file.

     Choose **OK**.
3. Review the loaded source repository, which Repository Manager displays as a hierarchical tree (design object hierarchy).

Related Topics:

- "Operational Reference Store Metadata Validation" on page 21
- "Issue Severity Levels" on page 20

Selecting the Target Repository for Visual Promotion

To select the target repository for promotion:

1. Click the **Target** drop-down list.

2. Select a repository in the list.

   Note: Repository Manager allows you to use a repository only after it has been validated and found to be free of Fatal errors.

   Repository Manager loads the target repository.

   Repository Manager displays the loaded target repository.

Related Topics:

- "Operational Reference Store Metadata Validation" on page 21
- "Issue Severity Levels" on page 20

Navigating the Design Object Hierarchy for Visual Promotion

For both the source and target repositories, the design object hierarchy contains the collection of design objects defined in the repository. The top level of the hierarchy consists of a list of design object types.

Expanding and Collapsing the Design Object Hierarchy

Click the following buttons to expand and collapse levels in the design object hierarchy.

Button	Description
	Expand the tree in the design object hierarchy.
	Collapse the tree in the design object hierarchy.

Conflict Indicators in the Source Repository

In the source design object hierarchy, icons provide additional information about the adjacent design object or its associated collection of child objects. The following indicators can appear in the source design object hierarchy:

Icon	Description
	Difference exists between the source and the target repository for this design object its associated collection of child objects. For example: The collection of associated child object contains a new or different design object. The design object exists in both the source and target repositories, but one or more design object's properties differ between the source and target repositories.
	Design object exists in one repository but not the other repository. If this icon appears next to a design object in the source repository, then the design object does not exist in the target repository. If the design object were promoted and changes applied, then the design object will be added to the target repository. If this icon appears next to a design object in the target repository, then the design object does not exist in the source repository. You can decide to keep the design object as is, or remove it.

Additional indicators can appear in markup mode.

Related Topics:

- "Viewing with Markup" on page 38
- "Conflicts When Copying Objects Between Repositories" on page 11

Properties Panel for the Selected Design Object

When a design object is selected in the design object hierarchy, Repository Manager displays the properties associated with that design object in a properties panel. The properties panel contains the following columns:

Column	Description
Property	Name of the selected property.
Value from source	Value of the selected property in the source repository.
Value from target	Value of the selected property in the target repository.
Final result	Value that will appear in the target repository after changes are applied.

Note: Differences in binary files—such as custom cleanse Java libraries or icons used in Hierarchy Manager—cannot be detected in Repository Manager. For such objects, use your own discretion about migrating these kinds of design objects.

Context Menus

Right-clicking a design object in the design object hierarchy displays a pop-up menu of available operations. In the source repository, you can choose **Promote**.

In the target repository, while in markup mode, you can choose **Revert** to revert a previously-promoted value to its former value.

Multiple Design Object Selections

Multiple design objects can be selected in the design object hierarchy in order to move them in a single operation.

If you select a parent design object (such as a base object) in the design object hierarchy, all of its child objects are included automatically (such as its columns, match settings, staging tables, and so on).

Related Design Objects

When promoting a design object to the target repository, you likely want to include related design objects as well. The design object hierarchy child objects or other related in the design object tree, even though the design object is of a different type. For example, a mapping has dependence on landing tables and staging tables. The following example shows that trust settings are configured for the LAST_NAME column in a base object.

Visually Promoting Changes to the Target Repository

To promote changes from the source repository to the target repository:

1. In the source list, select the design objects or properties that you want to promote.

2. Do one of the following:
   - Click the **Promote** button.
   - Drag selected design objects from the source repository and drop them anywhere onto the target repository.
   - Right-click in the source repository and choose **Promote** from the pop-up menu.
3. If Repository Manager detects any conflicts in the target system, it prompts you to choose an action.

   Select the action that is most appropriate for your particular metadata based on the desired outcome. Some conflicts can be resolved automatically based on whether the source or target value should be the final result after promotion. Other conflicts might require selective, manual intervention. For example, if you want to retain some values from the source and others from the target.

   Note: Property conflicts must be resolved, automatically or manually, before applying the change to the target repository. In contrast, dependency conflicts do not need to be resolved before applying the change. For example, if a base object in the source repository has an extra column, then the column will be added regardless of the action selected.
4. Select one of the following actions.

Option	Description
Merge	
Manually merge the conflicts	Provides a side-by-side comparison of values from the source and target design objects, allowing you to manually choose (on a case-by-case basis) the value to use in the target design object. Proceed to "Manual Conflict Resolution" on page 37.
Use source values as final result for conflict	Overwrites values in the target design object with the values from the source design object. Proceed to "Automatic Conflict Resolution" on page 38.
Use target values as final result for conflict	Retains values from the target design object. The target design object is unchanged. Proceed to "Automatic Conflict Resolution" on page 38.
Replace	
Replace the target object with source object	Completely replaces the target design object with the source design object. Used when the source design object contains the newer version of the design object. Proceed to "Automatic Conflict Resolution" on page 38.

Related Topics:

- "Manual Conflict Resolution" on page 37
- "Automatic Conflict Resolution" on page 38

Manual Conflict Resolution

If you chose to manually merge the conflicts, Repository Manager adds check boxes next to the conflicts in the source and target columns.

For each property conflict, select the check box next to the value that you want to survive after promotion.

Automatic Conflict Resolution

If you chose to automatically resolve conflicts based on a specific rule (value from source, value from target, or replace target with source), Repository Manager displays the Impact Analyzer to show you the impact of the promotion selections.

If the impact is acceptable, then click **OK**.

Viewing with Markup

The Markup button is a toggle that displays or hides visual indicators of proposed changes in the target repository.

The following indicators can appear in the target design object hierarchy in markup mode:

Icon	Description
	Design object has been added.
	Design object is to be modified.
	Design object has been deleted.
	Design object has been modified.

Finding Conflicts

Use the following buttons to jump to conflicts in the design object hierarchy for objects that are selected for promotion.

Button	Description
	Go to the previous conflict in the design object hierarchy.
	Go to the next conflict in the design object hierarchy that must resolved.

Reverting Changes

If you have not yet applied changes to the target repository, you can selectively undo any changes that you made.

To revert a change:

- In the target repository, right-click the object associated with the change that you want to restore, and then choose **Revert** from the pop-up menu.

  Repository Manager reverts the change and replaces it with the original value in the target repository.

Saving Changes in a Comparison Change List File

Repository Manager allows you to save proposed changes to a comparison change list XML file. You might save changes before applying them, for example, to:

- have them reviewed and approved first
- edit the changes manually in the XML file, such as selecting a subset of changes to apply
- save a log of changes for future reference
- apply the same changes to multiple target repositories

To save changes in a comparison change list XML file:

1. Click the **Save** button.

   Repository Manager displays a progress bar while it processes the changes.

   Repository Manager display the Save Change List dialog.
2. Navigate to the target directory where you will save the change list XML file.
3. Specify the following information about the repository.

Field	Description
Name	Logical name for this change list file. This value will be stored in the <name> tag of the change list XML file.
Description	Description of this change list file. This value will be stored in the <description> tag of the change list XML file.
File Name	Name of the change list file to save. Repository Manager will add an extension to this file (*.change.xml).

4. Click **Save**.

   The Repository Manager saves the specified change list file, displaying a progress bar while writing to the target location.
5. You can open the change list XML file in an editor to review its contents.

   The specific name and description, along with a time stamp, file name, and other information, is saved as attributes to the <changelist> element.

Applying Changes to the Target Repository

To apply changes to the target repository:

1. Click the **Apply** button.

   Repository Manager prompts you to select a rollback strategy in the event of a failure during the import process.

2. Select one of the following options:

Strategy	Description
Full rollback	Rolls back all of the changes attempted during the process of applying promoted changes to the target repository.
Rollback to last change	Rolls back to the last successful change prior to the interruption of applying promoted changes to the target repository. For example, if changes in base object columns A and B were successfully applied to the target repository, but the promotion application process failed before changes in column C were successfully applied to the target repository, then changes are rolled back to column B. Use this option if you are promoting many changes and want to keep successfully-applied changes in the event of a failure.

   Rollbacks for each design object involve rolling back two changes: the physical change in the repository (for example, removing a physical column), as well as the metadata about the design object (the metadata descriptor of the column).

3. Choose **OK**.

   If the target repository is currently registered with a proxy user, Repository Manager prompts you to enter the owner password for this repository.

4. If prompted, enter the password of the ORS schema owner.

5. Data integrity validation is performed based on the user choice. A dialog box asks the user to select if the data integrity validation is to be performed before promoting the changes (either in promote ChangeList or Visual).

   Repository Manager displays a progress bar.
   When finished, Repository Manager displays a message indicating whether the promote process succeeded.

Related Topics:

- "Repositories Currently Registered with a Proxy User" on page 14
- "Monitoring the Results of Changes" on page 14

Promoting Changes Using Change Lists

This section describes how to promote changes using change lists.

Overview of Change List Promotion Tasks

Note: Before you promote changes, make a back up copy of the target repository.

To promote changes using change lists:

1. Navigate to the **Promote / Change List** tab.
2. Select the target repository.
3. Open the change list XML file containing the changes that you want to apply to the target repository.
4. Optionally, inspect the proposed changes in the change list XML file.

5. Optionally, save proposed changes to a change list XML file.
6. Test the changes before applying them by running a simulation.
7. Apply the changes to the target repository.

Related Topics:

- "Exporting a Repository" on page 57
- "Navigating to the Promote / Change List Tab" on page 41
- "Select the Target Repository for Change List Promotion" on page 42
- "Opening a Comparison Change List XML File" on page 43
- "Navigating the List of Changes" on page 43
- "Viewing the Brief Description of a Change" on page 43
- "Viewing the Detailed Description of a Change" on page 44
- "Saving Changes in a Comparison Change List XML File" on page 44
- "Running a Simulation of Applying a Change List" on page 44
- "Applying a Change List to the Target Repository" on page 45

Navigating to the Promote / Change List Tab

To navigate to the Change List tab on the Promote tab:

1. Start the Repository Manager tool.
2. Click the **Promote** tab.
3. Click the **Change List** tab.

   Complete the remaining tasks in this section.

Command Buttons on the Change List Tab

The Change List tab has the following command buttons.

Button	Description
	Create a new change list by comparing two repositories.
	Open a change list.
	Save as a change list.
	Simulate applying a change list to the target repository.
	Apply a change list to the target repository.
	View a brief description of a change.

Button	Description
	View a detailed description of a change.
	Start the Schema Viewer for the selected repository.

Select the Target Repository for Change List Promotion

To select the target repository:

1. Click the Target drop-down list.
2. Select a repository in the list.

   Note: Repository Manager allows you to use a repository only after it has been validated and found to be free of Errors, or FATAL issues.

Related Topics:

- "Issue Severity Levels" on page 20
- "Operational Reference Store Metadata Validation" on page 21

Creating a Comparison Change List by Comparing Repositories

You can create a new change list by comparing the selected target repository with a source repository that you select.

1. Click the **Create Change List** button.

   Repository Manager displays the Create Change List dialog.
2. Select the source repository to use for the change list. You can choose either a database (ORS) or a creation change list XML file.
   - For a database repository, select one from the source repository list.

     If you selected a database repository that has not yet been validated, click the **Validate** button and complete the validation process.

     Note: Repository Manager allows you to use a repository only after it has been validated and found to be free of Errors, or FATAL issues.

     Click **OK**.

     Repository Manager loads the source repository.
   - For a change list XML file, click the **File Repository** tab.
     Click the **Open** button.

     Select the change list XML file (navigate to the folder if needed) and click **Open**.

     Note: You must select a creation change list that represents a complete schema as a basis of comparison. Repository Manager does not allow you to choose a comparison change list.

     Repository Manager loads and validates the repository from the selected file.

     Click **OK**.
3. Click **OK**.

   Repository Manager compares the two repositories and generates the change list.

Related Topics:

- "Navigating the List of Changes" on page 43
- "Issue Severity Levels" on page 20
- "Operational Reference Store Metadata Validation" on page 21

Opening a Comparison Change List XML File

You can open a comparison change list that contains the changes that you want to apply to the selected target repository. You can review the changes before applying them to the target repository.

To open a comparison change list XML file:

1. Click the **Open** button.
2. Specify the name of the change list XML file that you want to open, navigating to the folder where it is stored, if necessary.
3. Click **Open**.

   Repository Manager loads the specified change list file and displays the list of changes it contains.

Related Topics:

- "Navigating the List of Changes" on page 43

Navigating the List of Changes

To search for specific text in the list of changes for an opened change list XML file:

▶ Type the string that you want to search for in the **Search** field.

Repository Manager selects the first change that contains the specified string.

Use the following buttons to find other matches in the change list:

Button	Description
	Go to the previous instance of the string.
	Go to the next instance of the string.

Viewing the Brief Description of a Change

To view a brief description of a change in the change list:

1. Select a change in the change list.
2. Click the **View Description** button.

   Repository Manager displays a description of the change.
3. Click **Close**.

Viewing the Detailed Description of a Change

To view a detailed description of a change in the change list:

1. Select a change in the change list.
2. Click the **View Details** button.

   Repository Manager displays a detailed description of the change.
3. Click **Close**.

Saving Changes in a Comparison Change List XML File

Repository Manager allows you to save proposed changes to a change list XML file. You might save changes before applying them, for example, to:

- have them reviewed and approved first
- edit the changes manually in the XML file, such as selecting a subset of changes to apply
- save a log of changes for future reference
- apply the same changes to multiple target repositories

This process generates a comparison change list XML file.

To save changes as in a comparison change list XML file:

1. Click the **Save** button.

   Repository Manager displays a progress bar while it processes the changes.

   Repository Manager displays the Save Change List dialog.
2. Navigate to the target directory where you will save the change list XML file.
3. Specify the following information about the repository.

Field	Description
Name	Logical name for this change list file. This value will be stored in the <name> tag of the change list XML file.
Description	Description of this change list file. This value will be stored in the <description> tag of the change list XML file.
File Name	Name of the change list file to save. Repository Manager will add an extension to this file (*.change.xml).

4. Click **Save**.

   The Repository Manager saves the specified change list file, displaying a progress bar while writing to the target location.
5. You can open the change list XML file in an editor to review its contents.

   The specific name and description, along with a time stamp, file name, and other information, is saved as attributes to the <changelist> element.

Running a Simulation of Applying a Change List

You can simulate the process of applying a change list to the target repository to see the results and fix any errors before actually executing the changes.

To simulate applying a change list:

1. Click the  **Simulate** button.

   Repository Manager prompts you to confirm running the simulation.

2. Click **Yes** to confirm.

   If the target repository is currently registered with a proxy user, Repository Manager prompts you to enter the owner password for this repository.

3. If prompted, enter the password of the ORS schema owner.

4. Data integrity validation is performed based on the user choice. A dialog box asks the user to select if the data integrity validation is to be performed before promoting the changes (either in promote ChangeList or Visual).

   Repository Manager runs the simulation and displays a message about the results.

5. Click **OK**.

   If you encounter a failure message, you should fix the problem(s) and run the simulation again before actually applying the changes to the target repository. Otherwise, the process of applying changes will likely fail.

Related Topics:

- "Repositories Currently Registered with a Proxy User" on page 14

Applying a Change List to the Target Repository

To apply changes to the target repository:

1. Click the **Apply** button.

   Repository Manager prompts you to select a rollback strategy in the event of a failure during the import process.

2. Select one of the following options:

Strategy	Description
Full rollback	Rolls back all of the changes attempted during the promote process.
Rollback to last change	Rolls back to the last successful change prior to the interruption of the promotion application process. For example, if changes in base object columns A and B were successfully applied to the target repository, but the promotion application process failed before changes in column C were successfully applied to the target repository, then changes are rolled back to column B. Use this option if you are promoting many changes and want to keep successfully-applied changes in the event of a failure.

   Rollbacks for each design object involve rolling back two changes: the physical change in the repository (for example, removing a physical column), as well as the metadata about the design object (the metadata descriptor of the column).

3. Choose **OK**.

   If the target repository is currently registered with a proxy user, Repository Manager prompts you to enter the owner password for this repository.

4. If prompted, enter the password of the ORS schema owner.

5. Data integrity validation is performed based on the user choice. A dialog box asks the user to select if the data integrity validation is to be performed before promoting the changes (either in promote ChangeList or Visual).

   Repository Manager displays a progress bar.

   When finished, Repository Manager displays a message indicating whether the promote process succeeded.

Related Topics:

- "Repositories Currently Registered with a Proxy User" on page 14
- "Monitoring the Results of Changes" on page 14

Promoting Changes from the Command Line

To promote changes to a repository from the command line, use the `MetCommand` utility that is included in the MDM Hub Resource Kit. A repository is also known as an Operational Reference Store (ORS).

Before you promote changes to a target repository, export the source repository to a change list XML file. Also, ensure that the Resource Kit is installed in the MDM Hub environment. Run the commands to promote changes to a repository from the command line.

1. Edit the `SiperianConnection.properties` file in the following location:

   ```
   <Resource Kit installation directory>/samples/MetCommand/source/resources/properties
   ```

   Ensure that the application server connection property values match the application server configuration.

2. From a command line, change to the `MetCommand` directory, which is in the following location:

   ```
   <Resource Kit installation directory>/samples/MetCommand
   ```

3. To create a change list, run the following command:

   ```
   metcommand -createChangeList
   -sourceXmlFilename "<Source repository change list XML file name>"
   -targetOrsId "<Target repository ID>"
   -outputFilename "<Change list XML file name>"
   -propertiesFilename "<Resource Kit installation directory>/samples/MetCommand/source/
   resources/properties/SiperianConnection.properties"
   ```

   The process to create the change list compares the source repository change list with the target repository metadata and generates a change list for the target repository.

4. To validate the change list, run the following command:

   ```
   metcommand -validateChangeList
   -sourceXmlFilename "<Change list XML file name>"
   -targetOrsId "<Target repository ID>"
   -propertiesFilename "<Resource Kit installation directory>/samples/MetCommand/source/
   resources/properties/SiperianConnection.properties"
   ```

5. To apply the change list to the target repository, run the following command:

   ```
   metcommand -applyChangeList
   -sourceXmlFilename "<Change list XML file name>"
   -targetOrsId "<Target repository ID>"
   -propertiesFilename "<Resource Kit installation directory>/samples/MetCommand/source/
   resources/properties/SiperianConnection.properties"
   ```

6. To validate the target repository metadata, run the following command:

```
metcommand -validateMetadata
-targetOrsId "<Target repository ID>"
-propertiesFilename "<Resource Kit installation directory>/samples/MetCommand/source/
resources/properties/SiperianConnection.properties"
```

CHAPTER 5

Importing Design Objects

This chapter includes the following topics:

Overview

This chapter describes how to use the Repository Manager in the Hub Console to import new design objects into a repository.

About Importing Design Objects

This section describes concepts that you need to understand before importing design objects into an empty repository.

Import Process

In the Repository Manager, you can selectively import design objects from a source repository or change list into an empty target repository. The import process inserts metadata for design objects that are not yet defined in the target repository. For example, you can add a template of standard design objects to a new repository.

Note: You cannot import a change list created in one database type into the repository of another database type. For example, you cannot import a change list created in Oracle into a Microsoft SQL Server repository.

Importing contrasts with promoting design objects, in which design objects already exist in the target repository. If you want to add design objects to a repository that contains metadata, use the features on the Promote tab.

Related Topics:

- "Promoting Changes Between Repositories" on page 28

Design Objects That Can Be Imported

For a complete list of design objects that can be imported, see "Design Objects Supported in Repository Manager" on page 64.

Considerations for the Import Process

Before you import, consider the following information:

- Before you import design objects, consider making a back up copy of the target repository.
- Importing (and renaming) involves changes to the target repository only—the source repository remains unchanged.
- For import operations, Repository Manager requires a repository that has been validated and free of Errors, or FATAL issues.
- In Informatica MDM Hub version 9.5, the Period_Start_Date and Period_End_Date columns replace the Rel_Start_Date and Rel_End_Date columns. If you import a pre-9.5 change list into a fresh 9.5 installation, you must run script `migrate_hm_rel_start_end_dates.sql` after you import the change list to update the pre-9.5 schema. See "Update Relationship Base Object Start Date and End Date Information" on page 54.
- The Repository Manager export and import processes do not preserve ROWID_OBJECT values in design object records. When you export design objects, the ROWID_OBJECT values from the source ORS are not preserved in the export file. When you import these design objects into a target ORS, the import process assigns new ROWID_OBJECT values, which might differ from the corresponding records in the source ORS.
  Problems can arise if you depend on ROWID_OBJECT values to uniquely identify design objects. For example, Hierarchy Manager could be configured to use ROWID_BO_CLASS as the reference to the C_RBO_BO_CLASS table on the source schema. If a SIF client application depended on this configuration, it would work correctly in the source ORS but not necessarily in the target ORS. Therefore, instead of ROWID_OBJECT values, use unique identifiers to identify specific design objects, such as BO_CLASS_CODE for entities, REL_TYPE_CODE for relationship types, and HIERARCHY_CODE for hierarchies.
- Custom match populations are not imported from a change list. Instead, remove the custom match population element from the change list XML file, then either modify the target schema manually, or use direct database import/changelist options.
- Before importing into a repository, make sure that both the source and target repositories have sufficient privileges to all required tablespaces.
- If the source repository has the Admin system enabled as a state management override system, the import procedure does not update the state management override system indicator for the Admin system in the target repository. You must manually enable the Admin system in the target repository as a state management override system.

Related Topics:

- "Issue Severity Levels" on page 20
- "Considerations When Copying Metadata" on page 12
- "Exporting a Repository" on page 57

Importing Design Objects

This section describes how to import design objects in the Repository Manager.

Overview of Import Tasks

Note: Before you import design objects, make a back up copy of the target repository.

To import design objects into a repository:

1. Navigate to the **Import** tab.
2. Select the source repository.
3. Select the target repository.
4. Select the design objects that you want to import from the source repository.
5. Optionally, rename objects that you want to import.
6. Execute the import process.

Related Topics:

- "Selecting Design Objects to Import" on page 52
- "Renaming Design Objects" on page 52
- "Exporting a Repository" on page 57
- "Selecting the Source Repository to Import" on page 50
- "Selecting the Target Repository for Import" on page 51
- "Importing Selected Design Objects" on page 53

Command Buttons on the Import Tab

The Import tab has the following command buttons.

Button	Description
	Import objects by applying the change list to the target repository.
	Rename the selected design object.
	Collapse the node.
	Expand the node.

Selecting the Source Repository to Import

For the source repository, you can choose either a database (ORS) or a creation change list XML file.

To select the source repository for import:

1. Click the **Select** button next to the Source field.

   The Repository Manager displays the Open Repository window.

2. Select a source repository from the list.

   - For a database repository, select a source repository from the list.
     If you selected a database repository that has not yet been validated, click the **Validate** button and complete the validation process.

     Note: Repository Manager allows you to import a repository only after it has been validated and found to be free of Errors, or FATAL issues.

     Click **OK**.

     Repository Manager loads the source repository.

   - For a change list XML file, click the **File Repository** tab.
     Click the **Open** button.

     Select the change list XML file (navigate to the folder if needed) and click **Open**.

     Note: You must select a *creation* change list. Repository Manager will not allow you to select a comparison change list as a source for design objects to import.

     Repository Manager loads and validates the repository from the selected file.

     Choose **OK**.

3. Review the loaded source repository, which Repository Manager displays as a design object hierarchy.

Related Topics:

- "Issue Severity Levels" on page 20
- "Operational Reference Store Metadata Validation" on page 21

Selecting the Target Repository for Import

To select the target repository:

1. Click the Target drop-down list.
2. Select a repository from the list.

   Repository Manager loads the target repository and compares the two repositories.

Showing and Hiding Design Objects in the Hierarchy

Use the following buttons to expand and collapse the design object hierarchy.

Button	Description
	Expand the node.
	Collapse the node.

When you expand a node, Repository Manager expands the entire tree of child objects associated with the selected node.

Selecting Design Objects to Import

After selecting the source and target repositories, you can navigate the design objects hierarchy and select the design object(s) to import. When a design object is selected, Repository Manager analyzes its dependencies and then automatically selects all associated design objects. For example, if a package is selected, then all tables and queries on which the package is based are automatically selected (checked). Likewise, if a query is removed (unselected), then any packages based on that query are also removed. You can individually remove (unselect) any objects that you want to exclude from import.

In general, consider importing an entire collection of design objects at once, rather than importing subsets separately, so that required design objects are not inadvertently omitted. For example, rather than importing landing tables, staging tables, mappings, and packages separately, consider importing the entire base object so that the collection of dependent design objects is complete.

Selecting a Design Object

To select a design object:

- Click the check box next to the design object.

  For example, when a base object is selected, then the queries and packages based on it are optional.

Selecting an Object Type

To select all design objects of a particular object type:

- Select the check box next to the object type.

  Repository Manager automatically selects all design objects of that type.

Selecting All Design Objects

To select all design objects in the source repository:

- Select the **Repository** check box, which is at the top of the hierarchy.

Renaming Design Objects

You can rename design objects. For example, before importing a template of design objects, you might want to rename objects to comply with an organization's naming conventions.

Related Topics:

- "Design Objects That Can Be Renamed" on page 67

Design Objects with Globally-Unique Names

Certain design objects have globally unique names: staging tables and match path components. The names of other design objects are associated with the parent base object or column. Repository Manager ensures that name changes to parent design objects are properly implemented in child design objects. For example, if you rename a base object, the queries that reference that base will be updated accordingly.

Renaming the Selected Object

To rename the selected design object:

1. Do one of the following:
   - Select the object and click the **Rename** button.
   - Or, right-click the design object.

     Repository Manager prompts you to specify the new name.
2. Enter the following information:

Strategy	Description
New Name	Physical name of the design object. The new name must conform with the Informatica MDM Hub naming rules for the object type.
New Display Name	Name that will be used to display this design object in the Hub Console.

3. Choose **OK**.

   Repository Manager displays the renamed design object in the hierarchy.

   Note: Although the new name appears in the design object hierarchy for the source repository, the new name will apply to the target repository after the import process is complete. The name of the design object in the source repository will remain unchanged.

Importing Selected Design Objects

To import the selected design object(s):

1. Click the **Apply** button.

   Repository Manager prompts you to select a rollback strategy in the event of a failure during the import process.
2. Select one of the following options:

Option	Description
Full rollback	Rolls back all of the changes attempted during the import process.
Rollback to last change	Rolls back to the last successful change prior to the interruption of the import process. For example, if base object columns A and B were successfully imported, but the import process failed before column C was successfully created in the target repository, then changes are rolled back to column B. Use this option if you are importing many design objects and want to keep successfully-applied changes in the event of a failure.

   Rollbacks for each design object involve rolling back two changes: the physical change in the repository (for example, removing a physical column), as well as the metadata about the design object (the metadata descriptor of the column).
3. Choose **OK**.

   If the target repository is currently registered with a proxy user, Repository Manager prompts you to enter the owner password for this repository.
4. If prompted, enter the password of the ORS schema owner.

5. Data integrity validation is performed based on the user choice. A dialog box asks the user to select if the data integrity validation is to be performed before promoting the changes (either in promote ChangeList or Visual).

   Repository Manager displays a progress bar.

   When finished, Repository Manager displays a message indicating whether the import process succeeded.

Related Topics:

- "Repositories Currently Registered with a Proxy User" on page 14
- "Monitoring the Results of Changes" on page 14

Update Relationship Base Object Start Date and End Date Information

If you are upgrading to Informatica MDM Hub version 9.5 by installing a 9.5 installation and then importing a pre-9.5 change list, the following changes are required:

- The Rel_Start_Date and Rel_End_Date columns must be made nullable.
- Mappings to the Rel_Start_Date and Rel_End_Date columns need remapped to the Period_Start_Date and Period_End_Date columns.
- References to the Rel_Start_Date and Rel_End_Date columns need removed from the Hierarchy Manager relationship packages.

After importing the change list, perform the following steps to make the required changes:

1. In SQL*Plus, run the `migrate_hm_start_end_dates.sql` script found in the following locations:
   - On Windows: <*infamdm_install_directory*>\server\resource\database\migration_readiness\oracle
   - On UNIX: <*infamdm_install_directory*>/server/resource/database/migration_readiness/oracle

   The script runs and makes the required changes.
2. Restart the application server.

   The mappings to the Period_Start_Date and Period_End_Date columns are active.

CHAPTER 6

Exporting Repositories

This chapter includes the following topics:

Overview

This chapter describes how to use the Repository Manager in the Hub Console to export a repository to a change list XML file in your Informatica MDM Hub implementation.

About Exporting a Repository

This section describes what you need to understand before using the Repository Manager to export a repository.

About Exporting

You can use the Repository Manager to export an entire repository to a change list XML file, which can then be used to import design objects into another repository or to save it in a source control system for archival purposes.

Related Topics:

- "Change Lists" on page 11
- "Change List Reference" on page 68

Design Objects That Can Be Exported

For a complete list of design objects that can be exported, see "Design Objects Supported in Repository Manager" on page 64.

Related Topics:

- "Design Objects Supported in Repository Manager" on page 64
- "Change List Reference" on page 68

How Exported Change List XML Files Get Used

Once created, you can use the creation change list XML file for:

- importing new design objects into a repository.
- promoting changes between repositories. A creation change list can be used as a source repository for promotion.
- archiving in a source control system.

Related Topics:

- "Importing Design Objects" on page 48
- "Promoting Changes Between Repositories" on page 28
- "Selecting the Source Repository for Visual Promotion" on page 34

Considerations for the Export Process

Before you export a repository, consider the following information:

- Exporting involves no changes to the source repository.
- To export, Repository Manager requires a source repository that has been validated and free of Errors, or FATAL issues.
- The Repository Manager export and import processes do not preserve ROWID_OBJECT values in design object records. When you export design objects, the ROWID_OBJECT values from the source ORS are not preserved in the export file. When you import these design objects into a target ORS, the import process assigns new ROWID_OBJECT values, which might differ from the corresponding records in the source ORS.

Related Topics:

- "Issue Severity Levels" on page 20
- "Operational Reference Store Metadata Validation" on page 21

Command Buttons on the Export Tab

The Export tab has the following command buttons.

Button	Description
	Save as a change list.
	Start the Schema Viewer for the selected repository.

Exporting a Repository

To export a repository as a creation change list:

1. Start the Repository Manager tool.
2. Click the **Export** tab, if it is not already selected.
3. Select a repository to export from the drop-down list.

   Note: Repository Manager allows you to use a repository only after it has been validated and found to be free of Errors, and FATAL issues.
4. Click the **Save** button.

   Repository Manager displays the Export Repository dialog.
5. Navigate to the target directory where you will save the change list XML file.
6. Specify the following information about the repository.

Field	Description
Name	Logical name for this change list file. This value will be stored in the <name> tag of the change list XML file.
Description	Description of this change list file. This value will be stored in the <description> tag of the change list XML file.
File Name	Name of the change list file to save. Repository Manager will add an extension to this file (*.change.xml).

7. Click **Export**.

   The Repository Manager saves the specified change list file, displaying a progress bar while writing to the target location.
8. You can open the change list XML file in an editor to review its contents.

   The specific name and description, along with a time stamp, file name, and other information, is saved as attributes to the <changelist> element.

   Note: After you have created an export file and before using the file for import or promotion operations, you can edit the file, rename design objects, delete extraneous sections, and make other changes as needed. However, Informatica is not responsible for any consequences resulting from XML files that you have modified in this way.

Related Topics:

- "Issue Severity Levels" on page 20
- "Starting Repository Manager" on page 15

Exporting a Subset of Design Objects

Exporting generates a creation change list XML file that includes all the design objects in the source repository.

You might want to create a change list to share only a subset of selected design objects instead. For example, you might want to contribute only an individual cleanse function. To export a subset of design objects, complete the following tasks in the Promote tab:

1. Select the source repository that contains the design object(s) that you want to export.
2. Select an empty target repository.
3. In the source repository, select the object(s) that you want to promote.

   Optionally, you can selectively rename design objects if you want to save them in the change list under a different name.
4. Save the changes to a change list.

Once saved, you can then apply the changes in this comparison change list to any target repository by completing the following steps.

1. Open the target repository to which you want to add the design objects.
2. Open the comparison change list XML file that you created earlier.
3. Apply the changes in the comparison change list.

CHAPTER 7

Common Warehouse Model Support

This chapter includes the following topics:

Overview

The Repository Manager tool supports files in the Common Warehouse Model (CWM) format. A CWM file can be generated using a third-party software such as Erwin. The advantage to using CWM is that you can apply CWM schema on seed ORS using the import from CWM option in the Hub console.

If you apply a CWM schema on a seed ORS, system columns such as ROWID_OBJECT and CREATE_DATE are automatically added. ROWID_OBJECT is the primary key of the table enforced by MDM.

You must not import content metadata tables such as EMI, EMO, and XREF tables to a seed ORS, as these are created automatically.

Note: An EMI table is a system table and its structure is based on the match columns you define.

The Repository Manager tool provides the **Import from CWM file** tab and the **Export to CWM file** tab for working with CWM files.

Import from CWM File Tab

This tool is used to convert a CWM file into a MET changelist that can be applied against an existing ORS or saved for future use. The steps involved in this operation are:

1. Load CWM file – the file is loaded and the importable elements (tables, views, and foreign keys) are presented in a selection tree format. At this point the user can select the objects that will be imported and for the tables, can select the import table type (BO, Landing). The selection/type change is done by consecutive clicks on the table checkbox. The corresponding FKs and views are automatically selected/deselected.
2. Save changelist – using the selected objects a MET changelist is generated and saved.
3. Apply changelist - using the selected objects a MET changelist is generated and and applied against the selected ORS.

Command Buttons on the Import from CWM file Tab

The Import from CWM file tab has the following command buttons:

Button	Description
	Load a CWM file.
	Save as a change list.
	Import objects by applying the CWM change list to the target repository.

Importing Design Objects from a CWM File

This section describes how to import design objects from CWM file in the Repository Manager.

Note: Before you import from a CWM file, back up the target repository according to the instructions in "Exporting a Repository" on page 57.

To import from CWM file into a repository, perform the following steps:

1. In Repository Manager, click the **Import from CWM file** tab.
2. Load the CWM file for import.
   a. Click the **Load CWM File** button.

      The Repository Manager displays the Open dialog.

   b. Navigate to the `.cwm` file that you must import, select the CWM file, and click **Open**.

      Repository Manager loads and validates the selected CWM file.

   c. Review the loaded source repository, which Repository Manager displays as a design object hierarchy.
3. Select design objects to import.

- To select a design object, click the check box next to the design object.
  When a design object is selected, all associated design objects are automatically selected. For example, if a package is selected, then all tables and queries on which the package is based are automatically selected (checked). Likewise, if a query is removed (unselected), then packages based on that query are also removed. You can individually remove (unselect) any objects that you want to exclude from import.
- To select all design objects of a particular object type, select the check box next to the object type. Repository Manager automatically selects all design objects of that type.
- To select all design objects in the source repository, select the check box next to the repository at the top of the hierarchy.

In general, consider importing an entire collection of design objects at once, rather than importing subsets separately, so that required design objects are not inadvertently omitted. For example, rather than importing landing tables, staging tables, mappings, and packages separately, consider importing the entire base object so that the collection of dependent design objects is complete.

4. Import selected design objects.
   a. Click the **Apply Change List** button.

      Repository Manager displays an Open Repository dialog with a database repository list.
5. Select the target repository for import.
   a. Select a target repository from the database repository list and click **OK**.
   b. Repository Manager prompts you to select a rollback strategy in the event of a failure during the import process.

      Select one of the following options:

Option	Description
Full rollback	Rolls back all of the changes attempted during the import process.
Rollback to last change	Rolls back to the last successful change prior to the interruption of the import process. For example, if base object columns A and B were successfully imported, but the import process failed before column C was successfully created in the target repository, then changes are rolled back to column B. Use this option if you are importing many design objects and want to retain successfully-applied changes in the event of a failure.

   c. Click **OK**.

      If the target repository is currently registered with a proxy user, Repository Manager prompts you to enter the owner password for the repository.
   d. If prompted to enter the password, enter the password of the ORS schema owner.

      Repository Manager displays a progress bar. When the import process is complete, Repository Manager displays a message indicating whether the import process succeeded.
6. Save changes in a comparison change list XML file.
   a. Click the **Save** button.

      Repository Manager displays the Save Change List dialog.
   b. Navigate to the target directory where you need to save the change list XML file.
   c. In the File name field, enter the name of the change list file to be saved.

Repository Manager adds an extension to the file (`*.change.xml`).

d. Click **Save**.

Repository Manager saves the specified `change.xml` file, displaying a progress bar while writing to the target location.

Note: You can open the change XML file in an editor to review its contents.

Export to CWM File Tab

This tool is used to export the ORS metadata to CWM format. There are two steps involved in this operation:

1. Load repository – loads the source ORSrepository into the tool. The result of this operation is the selection tree of the exportable ORS objects. By selecting the tree elements the user can specify the ORS objects exported to CWM format.
2. Save CWM file – the selected objects are exported to CWM format and the file is saved (with an XML extension). Click on the tab and then click on the folder icon to import or export CWM files.

Command Buttons on the Export to CWM file Tab

The Export to CWM file tab has the following command buttons:

Button	Description
	Load repository.
	Save change list.

Exporting a Repository to a CWM File

To export a repository as a CWM XML file, perform the following steps:

1. In Repository Manager, click the **Export to CWM file** tab.
2. Load the database repository for export.
   a. Click the **Load Repository** button.

   The Repository Manager displays the Open Repository dialog.

   b. Select a repository from the Database Repository list and click **OK**.

   Note: Repository Manager allows you to use a repository only after it has been validated and found to be free of Error, Critical, or Fatal errors.

   c. Select the repository objects to be exported to CWM file.

3. Save the CWM file.
    a. Click the **Save** button.

       Repository Manager displays the Save dialog.
    b. Navigate to the target directory where you need to save the change list XML file.
    c. In the File name field, enter the name of the change list file to be saved.

       Repository Manager adds an extension to the file (`*.CWM.xml`).
    d. Click **Save**.

       Repository Manager saves the specified `CWM.xml` file, displaying a progress bar while writing to the target location.

Note: You can open the CWM XML file in an editor to review its contents.

APPENDIX A

Design Objects Reference

This appendix includes the following topics:

Overview

This appendix provides a reference for design objects that can be managed by Repository Manager.

Design Objects Supported in Repository Manager

Repository Manager allows you to manage the metadata in Operational Reference Store. The following table provides a detailed list of specific design objects that you can manage using Repository Manager. The table also describes whether Repository Manager supports:

- **validation** of this type of design object.
- **copying** design objects between repositories.

Finally, the table refers to a topic in the *Multidomain MDM Configuration Guide* (or other document) that provides more information about the specific element.

The following table lists design objects that can be managed by Repository Manager:

Informatica MDM Hub Component	Validate	Promote, Import, Export
mappings	yes	yes
cleanse		
- Process Server	yes	no
- cleanse function	yes	yes

Informatica MDM Hub Component	Validate	Promote, Import, Export
queries		
- queries	yes	yes
- custom queries using custom tables	yes	no
packages	yes	yes
- packages	yes	yes
- packages using custom queries	yes	no
schema		
- base object tables and columns	yes	yes
- match, including match rules, rule sets, and path components	yes	yes
- external match	yes	yes
- validation	yes	yes
- message triggers	yes	yes*
- relationships	yes	yes
- staging table	yes	yes
- custom index on base object	yes	yes
Trust		
- source system trust	yes	yes
- column trust	yes	yes
system state	yes	yes
batch groups (except custom objects)	yes	yes
message queues	partial (message triggers only)	partial (message triggers only)
Security Access Manager	yes	yes
user objects	yes	partial
Hierarchy Manager	yes	yes
Business Entity/Business Entity Configuration		
- business entities	yes	yes

Informatica MDM Hub Component	Validate	Promote, Import, Export
- business entity services	yes	yes
- REST business entity service configuration	yes	yes
- SearchableCO business entity service configuration	yes	yes
- WriteCO business entity service configuration	yes	yes

Related Topics:

- "Validating Metadata" on page 18
- "Promoting Changes Between Repositories" on page 28
- "Importing Design Objects" on page 48
- "Exporting Repositories" on page 55
- "Design Objects and Changes in a Change List XML File" on page 69

Design Object Dependencies

The following table describes dependencies between design objects.

Object Type	Dependencies
system	none
base object	- columns - match configuration - validation rules **Note:** - If staging table lookups, match paths, and validation rules refer to other design objects. then such design objects are required dependencies. - A foreign key does *not* imply a required dependency to the related design object. The foreign key is imported only if both design objects are imported. Before importing, the user is informed of any foreign keys that are not imported.
staging table	base object, columns, system
landing table	columns
query	Any design objects used in a query (such as base objects)
package	query
mapping	- landing tables - staging tables

Object Type	Dependencies
cleanse function	Any other cleanse functions used. **Note:** If a cleanse function imports one function from a custom Java library, all functions from that library are imported.
Hierarchy Manager	base objects and packages used in the configuration
business entities	base objects
business entity field	- base object column - parent business entity node
business entity child base object	relationship between parent base object and child base object
business entity referenceOne and referenceMany element	referenced base object
business entity service	inputs that are generated from business entity
writeCO business entity service	business entity definitions
searchCO business entity service	business entity definitions
business entities based on Hierarchy Manager enabled base objects	Hierarchy Manager items such as hierarchies and relationship types

Related Topics:

- "Dependencies" on page 10
- "Dependency Conflicts" on page 30

Design Objects That Can Be Renamed

You can rename only the following design objects, during a promotion or import:

- Base Object
- Cleanse Library

  Note: You can rename only internal cleanse libraries (user libraries or Java libraries) created with the Cleanse Functions tool. Renaming of external cleanse functions from third-party cleanse engines is not supported.

APPENDIX B

Change List Reference

This appendix includes the following topics:

Overview

This appendix provides a reference for Informatica MDM Hub change list XML files.

Note: A change list XML file can contain references to system objects—such as the Admin system, system cleanse functions, and the Hierarchy Manager RBO objects—that are not included in the change list XML file. The Repository Manager resolves these references internally.

Related Topics:

- "Change Lists" on page 11

Change List XSD File

The siperian-changelist.xsd file defines the structure of change list XML files. This file is provided in the MDM Hub Resource Kit. For more information, see the *Multidomain MDM Resource Kit Guide*.

Root Tags and Attributes in a Change List XML File

Tag	Description
<changeList>	Root tag for a change list XML file. Attributes include: - xmlns:java="http://java.sun.com" - xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" - xsi:noNamespaceSchemaLocation="siperian-changelist.xsd" - xsi:schemaLocation="http://java.sun.com java.xsd" - creationDate=time stamp when the file was created - description=description specified by the user. For comparison change lists, default description refers to the source and target repositories (for example: "Compare docs--ORS2 (source) to docs--ORS1 (target)") - listType=one of the following values: "creation" or "comparison" - name=descriptive name specified by the user - version=version number of the change list XML schema (such as version=2)
<changes>	List of changes to apply to the target repository.

Types of Changes in a Change List XML File

In the Repository Manager, a *change* is an operation in the change list that is executed against the target repository. For example, a change might add a table, update settings on a match rule, delete a package, move a query to different query group, and so on. The following table describes the types of changes that are defined in a change list XML file.

Change	Description
addObjectType	Adds the specified design object with the associated properties.
modifyObjectType	Adds the specified design object with the associated properties.
deleteObjectType	Adds the specified design object with the associated properties.
revertObjectType	Applies to Hierarchy Manager design objects and refers to the process of reverting a Hierarchy Manager entity object or relationship object to a Informatica MDM Hub base object.

Design Objects and Changes in a Change List XML File

The following table lists the types of design objects, and associated changes, that can occur in a change list XML file. For detailed descriptions of the properties associated with each design object, refer to the design object's documentation in the *Multidomain MDM Configuration Guide* or other document as appropriate. For

other objects specified in the change list XML file, such as ColumnDataType and ChangeError, refer to the XSD file described in "Change List XSD File" on page 68.

Component	Design Object	Change
Batch	BatchGroup	addBatchGroup
		deleteBatchGroup
		modifyBatchGroup
Cleanse	CleanseFunction	addCleanseFunction
		deleteCleanseFunction
		modifyCleanseFunction
	CleanseLibrary	addCleanseLibrary
		deleteCleanseLibrary
		modifyCleanseLibrary
Hierarchy Manager	HmBlob	addHmBlob
		deleteHmBlob
		modifyHmBlob
	HMColumnPackage	addHmColumnPackage
		deleteHmColumnPackage
		modifyHmColumnPackage
	HmEntityObject	addHmEntityObject
		revertHmEntityObject
	HmEntityType	addHmEntityType
		deleteHmEntityType
		modifyHmEntityType
	HmHierarchy	addHmHierarchy
		deleteHmHierarchy
		modifyHmHierarchy
	HMPackage	addHmPackage
		deleteHmPackage
	HmProfile	addHmProfile

Component	Design Object	Change
		deleteHmProfile
		modifyHmProfile
	HMRelationshipObject	addHmRelationshipObject
		revertHmRelationshipObject
	HmRelationshipType	addHmRelationshipType
		deleteHmRelationshipType
		modifyHmRelationshipType
	HmSandbox	addHmSandbox
		deleteHmSandbox
		modifyHmSandbox
Match	MatchColumn	addMatchColumn
		deleteMatchColumn
		modifyMatchColumn
	MatchPathComponent	addMatchPathComponent
		deleteMatchPathComponent
		modifyMatchPathComponent
	MatchPathComponentFilter	addMatchPathComponentFilter
		deleteMatchPathComponentFilter
		modifyMatchPathComponentFilter
	MatchPopulation	addMatchPopulation
		modifyMatchPopulation
	MatchRuleSet	addMatchRuleSet
		deleteMatchRuleSet
		modifyMatchRuleSet
	PrimaryKeymatchRule	addPrimaryKeyMatchRule
		deletePrimaryKeyMatchRule
		modifyPrimaryKeyMatchRule

Component	Design Object	Change
Message Triggers	Message Triggers	addMessageTrigger
		deleteMessageTrigger
		modifyMessageTrigger
Package	Package (including packages based on custom queries)	addPackage
		deletePackage
		modifyPackage
	PackageColumn	modifyPackageColumn
Query	Query	addQuery
		deleteQuery
		modifyQuery
	QueryGroup	addQueryGroup
		deleteQueryGroup
		modifyQueryGroup
Schema	BaseObject	addBaseObject
		deleteBaseObject
		modifyBaseObject
		modifyCascadeUnmerge
	BaseObjectColumn	addBaseObjectColumn
		deleteBaseObjectColumn
		modifyBaseObjectColumn
	ForeignKey	addForeignKey
		deleteForeignKey
	Index	addIndex
		deleteIndex
		modifyIndex
	LandingTable	addLandingTable
		deleteLandingTable

Component	Design Object	Change
		modifyLandingTable
	LandingTableColumn	addLandingTableColumn
		deleteLandingTableColumn
		modifyLandingTableColumn
	Mapping	addMapping
		deleteMapping
		modifyMapping
	StagingTable	addStagingTable
		deleteStagingTable
		modifyStagingTable
	StagingTableColumn	addStagingTableColumn
		deleteStagingTableColumn
		modifyStagingTableColumn
	SystemColumnTrust	modifySystemColumnTrust
	SystemTable	addSystemTable
	SystemTableColumn	systemTableColumn
	SystemTableOrderColumn	orderColumn
Security Access Manager	ResourceGroup	addResourceGroup
		deleteResourceGroup
		modifyResourceGroup
	Role	addRole
		deleteRole
		modifyRole
	SecureResource	addSecureResource
		deleteSecureResource
Search	SearchableField	addSearchableField
		modifySearchableField

Component	Design Object	Change
		deleteSearchableField
Source Systems	DistinctSystem	addDistinctSystem
		deleteDistinctSystem
		modifyDistinctSystem
	ImmutableSystem	modifyImmutableSystem
	System	addSystem
		deleteSystem
		modifySystem
Validation	ValidationRule	addValidationRule
		deleteValidationRule
		modifyValidationRule

Related Topics:

- "Design Objects Supported in Repository Manager" on page 64
- "Change List XSD File" on page 68

APPENDIX C

MetCommand Reference

This appendix includes the following topics:

Overview

This appendix provides a reference for the MetCommand utility.

About MetCommand

MetCommand is a command-line wrapper for the Repository Manager APIs, which are used to manage the metadata in an MDM Hub implementation.

API Call	Description
applyChangeList	Apply a changelist.
createChangeList	Create a changelist.
getOrsMetadata	Export metadata to the specified file.
validateChangeList	Validate a changelist.
validateMetadata	Validate a repository.

For more information about these APIs, see the *Multidomain MDM Services Integration Framework Guide* and the *Multidomain MDM Javadoc*.

Before You Begin

Complete the instructions in this section before you begin using MetCommand.

Prerequisites

MetCommand is a Java program. Therefore, a JRE or JDK must be installed, and the `java` command must be in the OS path.

Ensure that the `informatica-bpm-adapter.jar` file is present in the MetCommand utility. If the JAR file is missing, locate it in the `<MDM Hub installation directory>/hub/server/lib` directory, and then copy it into the `MetCommand\lib` directory.

Connection Setup

The properties for connecting to the Informatica MDM Hub are found in the following file:

```
MetCommand\source\resources\properties\SiperianConnection.properties
```

You should edit this file and configure the username, password, and `orsId` for your Informatica MDM Hub installation.

- Uncomment the protocol for the application server you are using.
- Comment out the application servers you are not using.
- If your application server is not on `localhost`, replace `localhost` with the host name of the application server.
- The `orsId` is used by default as the `sourceOrsId`. This can be overridden with a command-line argument. The `orsId` is the name of the ORS as registered in the Databases tool in the Hub Console.

Usage

This section describes how to use MetCommand.

Help Output

The `metcommand.cmd` file (on Windows) displays usage information if you run the utility from the command line without any parameters.

```
>metcommand
usage: MetCommand
-applyChangeList            apply a changelist (-sourceXmlFilename, -targetOrsId)
-createChangeList           create a changelist (-sourceOrsId|-sourceXmlFilename,

                        -targetOrsId, -outputFilename)
-getOrsMetadata             export metadata to the specified file (-sourceOrsId,
                            -outputFilename)
-outputFilename <arg>       output file name
-password                   Owner password
-propertiesFilename <arg>   hub client properties file name
-rollbackToLast             rollback to last change
-sourceOrsId <arg>          source ors id
-sourceXmlFilename <arg>    source ors file
-targetOrsId <arg>          target ors id
```

```
-validateChangeList         validate a changelist (-sourceXmlFilename, -targetOrsId)
-validateMetadata           validate a ORS (-targetOrsId)
```

Command-line Arguments

The following table describes the command-line arguments.

Argument	Description	Associated Argument(s)
-applyChangeList	Applies a changelist.	-sourceXmlFilename -targetOrsId -rollbackToLast -password
-createChangeList	Creates a changelist.	-sourceOrsId or -sourceXmlFilename -targetOrsId -outputFilename
-getOrsMetadata	Exports metadata to the specified file.	-sourceOrsId -outputFilename
-outputFilename <arg>	Output file name.	<arg>=valid filename
-password	If specified, MetCommand prompts the user for the password of the ORS schema owner in the command window or shell.	
-propertiesFilename <arg>	Client properties file name.	<arg>=valid filename
-rollbackToLast	Rolls back to the last change. If not specified, applyChangeList defaults to a full rollback.	
-sourceOrsId <arg>	ORS id of the source repository.	<arg>=valid ORS ID
-sourceXmlFilename<arg>	Filename of the source repository (changelist).	<arg>=valid filename
-targetOrsId <arg>	ORS id of the target repository.	<arg>=valid ORS ID
-validateChangeList	Validates a changelist.	-sourceXmlFilename -targetOrsId -password
-validateMetadata	Validates an ORS.	-targetOrsId

XML Over HTTP

Note: The properties are set to communicate with the Hub via XML over HTTP. The `metcommand.cmd` file references only the JAR files that support this protocol – not the ones needed for EJB or SOAP. This is done because there would be little benefit to using the other protocols here, and the JAR files user is not dependent on the application server that is being used.

Proxy User Access

If an ORS is currently registered with a proxy user, in order to validate a repository or apply a changelist, MetCommand must be configured to prompt for the password of the schema owner (`-password` argument). The user must provide a valid password in order for MetCommand to execute successfully on the target ORS.

For example:

```
cmd /c metcommand -validateChangeList -targetOrsId %target_ors% -sourceXmlFilename
     %changelist_file%  -password
cmd /c metcommand -applyChangeList -targetOrsId %target_ors% -sourceXmlFilename
     %changelist_file%   -password
```

The password prompt is not used with other MetCommand operations. For information about proxy user configuration, see the *Multidomain MDM Installation Guide* and the *Multidomain MDM Configuration Guide*.

Rolling Back Changes When Applying a Changelist

When using the applyChangeList argument with MetCommand, the `-rollbackToLast` argument is used to perform a partial rollback. For example:

```
cmd /c metcommand -applyChangeList -targetOrsId %target_ors% -sourceXmlFilename
     %changelist_file% -rollbackToLast  -password
```

If not specified, applyChangeList defaults to a full rollback.

Examples

This section provides examples of MetCommand usage.

Get Metadata

```
metcommand -getOrsMetadata -sourceOrsId localhost-orcl-cmx_ors2
ORS Metadata has been written to: localhost-orcl-cmx_ors2.change.xml
```

Create ChangeList

```
metcommand -createChangeList -sourceOrsId localhost-orcl-cmx_ors1 -targetOrsId
     localhost-orcl-cmx_ors2
Change list has been written to: localhost-orcl-cmx_ors1.change.xml
```

Or

```
metcommand -createChangeList -sourceXmlFilename localhost-orcl-cmx_ors1.change.xml
     -targetOrsId localhost-orcl-cmx_ors2 -outputFilename changelist.change.xml
Change list has been written to: changelist.change.xml
```

Validate ChangeList

```
metcommand -validateChangeList -sourceXmlFilename changelist.change.xml
    -targetOrsId localhost-orcl-cmx_ors2
The change list is valid.
```

For an ORS that is currently registered with a proxy user (`-password` argument is required):

```
metcommand -validateChangeList -sourceXmlFilename changelist.change.xml -targetOrsId
     localhost-orcl-cmx_ors2 -password
The change list is valid.
```

Apply ChangeList

```
metcommand -applyChangeList -sourceXmlFilename changelist.change.xml -targetOrsId
     localhost-orcl-cmx_ors2
The change list has been applied.
```

For an ORS that is currently registered with a proxy user (`-password` argument is required):

```
metcommand -applyChangeList -sourceXmlFilename changelist.change.xml -targetOrsId
    localhost-orcl-cmx_ors2 -password
The change list has been applied.
```

RollbackToLast

```
metcommand -applyChangeList -sourceXmlFilename changelist.change.xml -targetOrsId
     localhost-orcl-cmx_ors2 -rollbackToLast
A partial rollback will happen.
```

For an ORS that is currently registered with a proxy user (`-password` argument is required):

```
metcommand -applyChangeList -sourceXmlFilename changelist.change.xml -targetOrsId
     localhost-orcl-cmx_ors2 -rollbackToLast  -password
A partial rollback will happen.
```

Validate Metadata

```
metcommand -validateMetadata -targetOrsId localhost-orcl-cmx_ors2
The ORS is valid.
```

Return Codes

Each METCommand API call option returns an integer value (0 for success, -1 for failure), which can be trapped and handled in a script.

The following table describes the meaning associated with a Failure (return code=-1) for the API call options.

API Call	Meaning
validateMetadata	MET validation was not successfully initiated, or validation messages with severity level Errors and FATAL were returned.
validateChangeList	Validation of changelist was not successful.
createChangeList	Changelist was not created.
applyChangeList	Changelist was not applied.
rollbackToLast	Rollback to last change not done.
getOrsMetadata	ORS Metadata was not written to file.

Script Execution

MetCommand can be executed inside a script that automates the promotion of design objects from one ORS to another.

Running a Custom Script

If you create a custom script:

1. Run the newly-created script from the following directory:

   Windows:

   ```
   resourcekit\samples\MetCommand
   ```

   Unix:

   ```
   resourcekit/samples/metcommand
   ```

2. Check for any error messages in the command line window or redirect the output and check the execution of Met API calls.

Example Script

The following code shows a Windows batch script that makes various MetCommand calls. You can adapt this script using your schema names.

```
REM Sample Windows batch script using METCommand
echo off
set target_ors=localhost-orcl-target_ors
set source_ors=localhost-orcl-newtest1
set changelist_file=changelist1.change.xml
cmd /c metcommand -validateMetadata -targetOrsId %source_ors%
IF NOT %ERRORLEVEL% == 0 GOTO METCOMMAND_ERRORED
cmd /c metcommand -validateMetadata -targetOrsId %target_ors%
IF NOT %ERRORLEVEL% == 0 GOTO METCOMMAND_ERRORED
cmd /c metcommand -createChangeList -targetOrsId %target_ors% -sourceOrsId %source_ors%
   -outputFilename %changelist_file%
IF NOT %ERRORLEVEL% == 0 GOTO METCOMMAND_ERRORED
cmd /c metcommand -validateChangeList -targetOrsId %target_ors% -sourceXmlFilename
   %changelist_file%
IF NOT %ERRORLEVEL% == 0 GOTO METCOMMAND_ERRORED

cmd /c metcommand -applyChangeList -targetOrsId %target_ors% -sourceXmlFilename
   %changelist_file%
IF NOT %ERRORLEVEL% == 0 GOTO METCOMMAND_ERRORED
cmd /c metcommand -validateMetadata -targetOrsId %target_ors%
IF NOT %ERRORLEVEL% == 0 GOTO METCOMMAND_ERRORED
GOTO DONE
:METCOMMAND_ERRORED
ECHO MetCommand Failed
:DONE

ECHO Done
```

Extending MetCommand

The source code and build files for MetCommand are included in the Resource Kit as an example of how to use the Repository Manager APIs. This allows for modifications or extensions to be made if needed.

INDEX

R

S

T

U

V