Informatica®
10.1

Metadata Manager Custom Metadata Integration Guide

Informatica Metadata Manager Custom Metadata Integration Guide
10.1
June 2016

Publication Date: 2018-07-02

Table of Contents

Preface

The *Metadata Manager Custom Metadata Integration Guide* provides methodology and procedures for integrating custom metadata into the Metadata Manager warehouse. This book is written for system administrators who want to load metadata from a repository type for which Metadata Manager does not package a model. This book assumes that system administrators have knowledge of relational database concepts, models, and PowerCenter.

Informatica Resources

Informatica Network

Informatica Network hosts Informatica Global Customer Support, the Informatica Knowledge Base, and other product resources. To access Informatica Network, visit https://network.informatica.com.

As a member, you can:

- Access all of your Informatica resources in one place.
- Search the Knowledge Base for product resources, including documentation, FAQs, and best practices.
- View product availability information.
- Review your support cases.
- Find your local Informatica User Group Network and collaborate with your peers.

As a member, you can:

- Access all of your Informatica resources in one place.
- Search the Knowledge Base for product resources, including documentation, FAQs, and best practices.
- View product availability information.
- Find your local Informatica User Group Network and collaborate with your peers.

Informatica Knowledge Base

Use the Informatica Knowledge Base to search Informatica Network for product resources such as documentation, how-to articles, best practices, and PAMs.

To access the Knowledge Base, visit https://kb.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

To get the latest documentation for your product, browse the Informatica Knowledge Base at https://kb.informatica.com/_layouts/ProductDocumentation/Page/ProductDocumentSearch.aspx.

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

Informatica Product Availability Matrixes

Product Availability Matrixes (PAMs) indicate the versions of operating systems, databases, and other types of data sources and targets that a product release supports. If you are an Informatica Network member, you can access 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. Developed from the real-world experience of hundreds of data management projects, Informatica Velocity represents the collective knowledge of our consultants who have worked with organizations from around the world to plan, develop, deploy, and maintain successful data management solutions.

If you are an Informatica Network member, you can access 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 augment, extend, or enhance your Informatica implementations. By leveraging any of the hundreds of solutions from Informatica developers and partners, you can improve your productivity and speed up time to implementation on your projects. You can access Informatica Marketplace at https://marketplace.informatica.com.

Informatica Global Customer Support

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

To find your local Informatica Global Customer Support telephone number, visit the Informatica website at the following link:
http://www.informatica.com/us/services-and-training/support-services/global-support-centers.

If you are an Informatica Network member, you can use Online Support at http://network.informatica.com.

Part I: Concepts and Models

This part contains the following chapters:

CHAPTER 1

Understanding Custom Metadata Integration

This chapter includes the following topics:

Understanding Custom Metadata Integration Overview

You can use Metadata Manager to load custom metadata into the Metadata Manager warehouse. Custom metadata is metadata that you define. Custom metadata includes metadata from custom metadata sources as well as custom properties and relationships for all resource types.

You can store the following types of custom metadata into the Metadata Manager warehouse:

Metadata from a custom metadata source

Add or load custom metadata from a metadata source for which you create a model and define the model classes, properties, and relationships. For example, you can load a custom resource that represents a Microsoft Access database or a custom program that you use to transform and load data.

Classes

Classes define the types of objects that a metadata source contains. When you create a custom model, you create classes to represent the data structures and fields in the metadata source.

Properties

Add custom properties to existing metadata in the Metadata Manager warehouse. For example, you can add a property to the Cognos model on the **Model** tab, and then edit the value for the property on the **Browse** tab.

Relationships

Create relationships between custom objects and objects in the same resource or in other resources. Create relationships to indicate that the objects are associated or to enable data lineage analysis. Before

you create relationships between objects, you might have to create relationships between the object classes.

After you add or load the custom metadata into the Metadata Manager warehouse, use Metadata Manager to analyze the metadata.

You can also migrate custom metadata from one Metadata Manager instance to another instance.

Custom Metadata Integration Process

The process for custom metadata integration involves adding and loading custom metadata. You can add or load metadata for a custom metadata source. You can also add custom properties for models and configure relationships between models.

To add or load metadata for a custom metadata source, you must first define a model for the source metadata. The model defines the representation of the metadata that Metadata Manager will contain. To define the custom model, you create classes, properties, and relationships. After you define the model, you can add or load the custom metadata.

You can add the metadata to the metadata catalog on the **Browse** tab. Add the metadata through the metadata catalog if the number of custom objects, properties, and relationships is small. You can also add metadata through the metadata catalog when you do not need to sync the metadata in the metadata source and Metadata Manager warehouse on a regular basis.

You can load metadata using a custom XConnect. Create a custom XConnect if you want to keep the metadata in the metadata source and the Metadata Manager warehouse in sync. With a custom XConnect, you can repeatedly load metadata from the custom metadata source to the Metadata Manager warehouse to capture changes in the metadata.

You can use either of the following methods to create a custom XConnect:

Create load templates.

Load templates contain rules that map the custom metadata to the model that you created. After you apply the load templates to the custom metadata source files, Metadata Manager can extract the metadata from the files and load it into the Metadata Manager warehouse.

Use this method when you want to accomplish any of the following tasks:

- Apply one load template to multiple resources of the same model.
- Create reusable rules that define data lineage relationships between groups of objects.
- Create enumerated links that define data lineage relationships between specific pairs of objects.

Use the Custom Metadata Configurator.

Use the Custom Metadata Configurator to create a custom resource template and a PowerCenter workflow that extracts the custom metadata. The resource template maps the custom metadata to the model that you created.

Use this method when you want to accomplish the following tasks:

- Create object-level relationships that enable data lineage between specific objects.
- Use multiple Custom Metadata Configurator templates for the same model.

Note: Custom XConnects created with a load template have better logging, greater validation, and better performance than custom XConnects created with the Custom Metadata Configurator. As of 9.5.1, the

Custom Metadata Configurator is provided for backward compatibility. The Custom Metadata Configurator will be deprecated in a future release.

Related Topics:

- "Custom XConnect Created with a Load Template" on page 34
- "Custom XConnect Created with the Custom Metadata Configurator" on page 99

Adding Custom Metadata

To add metadata for a custom metadata source, complete the following steps:

1. Create the model.

   Create the model to represent the metadata in the metadata source on the **Model** tab in Metadata Manager.
2. Add classes, properties, and relationships.

   Add custom classes, properties, and relationships to the model on the **Model** tab.
3. Add the metadata to the Metadata Manager warehouse.

   Create a resource in the metadata catalog that represents the source metadata on the **Browse** tab. Add custom metadata objects based on the classes that you create.

Loading Custom Metadata with a Load Template XConnect

To load custom metadata using a Load Template XConnect, complete the following steps:

1. Create the model.

   Create the model that represents the custom metadata on the **Model** tab in Metadata Manager.
2. Add classes, properties, and relationships.

   Add custom classes, properties, and relationships to the model on the **Model** tab.
3. Create the load template.

   Generate a default load template. When you generate a default load template, you also create empty metadata source files. After you populate the metadata source files with the source metadata that you want to load, update the default load template based on the metadata source files.
4. Upload the load template.

   Upload the load template on the **Model** tab in Metadata Manager.
5. Create the resource for the custom metadata.

   Create the resource on the **Load** tab. When you create the resource, you specify the metadata source files, configure the rules for mapping metadata to model components, and upload enumerated links information.
6. Configure linking rules.

   Configure linking rules to run data lineage across metadata sources. Define linking rules in a rule set XML file, and then create a rule set in the Metadata Manager repository from the XML file.
7. Load the resource.

   Load the metadata for the custom resource.
8. Create links between resources.

   Create links between objects in the custom resource and objects in other resources using the linking rules.

9. Edit the resource.

   Edit the resource, if required, to add other metadata source files, enumerated links, or linking rules.

Loading Custom Metadata with a Custom Metadata Configurator XConnect

To load custom metadata using a Custom Metadata Configurator XConnect, complete the following steps:

1. Create the model.

   Create the model that represents the custom metadata on the **Model** tab in Metadata Manager.

2. Add classes, properties, and relationships.

   Add custom classes, properties, and relationships to the model on the **Model** tab.

3. Create the custom resource.

   Create a resource for the model on the **Load** tab.

4. Create the template and generate PowerCenter objects.

   Use the Custom Metadata Configurator to create a custom resource template and generate the PowerCenter workflow that loads the custom metadata into the Metadata Manager warehouse.

5. Configure and load the resource.

   Configure the custom resource template and metadata source files for the resource, and then load the metadata for the custom resource.

6. Configure rule-based links.

   Configure rule-based links to run data lineage across metadata sources. Define linking rules in a rule set XML file, and then create a rule set in the Metadata Manager repository from the XML file.

7. Create links between resources.

   Create links between objects in the custom resource and objects in other resources using the linking rules.

Metadata Manager Concepts

Metadata Manager uses the following concepts to define custom metadata in the Metadata Manager warehouse:

- Models
- Classes
- Properties
- Groups
- Relationships
- Business Name

Metadata Manager stores the models, including the associated classes, properties, and relationships in the Metadata Manager repository. You can run Metadata Manager reports in the JasperReports application to get more information about the models, classes, properties, and relationships in the Metadata Manager repository.

Models

A model is a group of classes, properties, and relationships designed for a particular type of metadata source. Metadata Manager uses models to classify metadata stored in the Metadata Manager warehouse. When you load metadata into the Metadata Manager warehouse, Metadata Manager extracts the metadata defined in the model.

Metadata Manager uses the following types of models:

Packaged models

Models that define the metadata that Metadata Manager can extract from specific application, business glossary, business intelligence, data integration, data modeling, and relational metadata sources.

You can edit packaged application, business intelligence, data integration, data modeling, and relational models to add properties. You cannot edit the packaged business glossary model.

Universal models

Models that define metadata from business intelligence, data integration, data modeling, and database management metadata sources for which Metadata Manager does not package a model.

You create universal models. To create a universal model, you create a plug-in, copy the plug-in to the Metadata Manager Service plug-ins directory, and recycle the Metadata Manager Service.

Custom models

Models in which you define the representation of metadata. You create custom models on the **Model** tab.

View all models on the **Model** tab.

Classes

Classes define the types of objects that a metadata source contains. For example, the Source Definition class defined in the PowerCenter model contains PowerCenter source definitions.

View the classes for packaged and universal models and add custom classes for custom models on the **Model** tab.

Properties

Properties are characteristics of metadata objects. For example, a metadata object can have a property called Usage that contains a description of how to use the object or how it is used within a metadata source.

You can create properties that apply to all objects of a class for any model except the business glossary model.

Groups

Groups are metadata objects of the same object class type.

Metadata Manager can include metadata objects in logical groups in the metadata catalog when it extracts metadata. You can specify a group when you create a class for a metadata object. When you view the metadata for the class in the metadata catalog, Metadata Manager groups objects from the same class type and with the same parent object in a logical group.

Objects of different classes can also belong to the same group. Metadata Manager does not store groups as metadata in the Metadata Manager repository. This means, in reporting on Metadata Manager, that you do not find an object that represents a group.

Relationships

Relationships are associations between two classes or two metadata objects.

You create the following types of relationships in Metadata Manager:

- Parent-child relationships between classes within a model
- Class-level relationships between custom classes and classes in other models
- Object-level relationships between individual metadata objects

You can create relationships when you create a custom model, edit custom classes, or edit custom metadata objects. You cannot create relationships for classes or metadata objects in packaged or universal models.

For example, you can edit a custom metadata class and create a relationship from the custom class to the Oracle Table class. Alternatively, you can edit the custom metadata class and create a relationship to the custom class from the Oracle Table class. However, you cannot edit the Oracle Table class and create a relationship to or from any other metadata class type.

Parent-Child Relationships

Parent-child relationships establish the hierarchy of classes within a model. View and create parent-child relationships when you create a model on the **Model** tab.

Create a parent-child relationship to specify that one class is the child class of a different class in the same model. For example, you create a custom model that contains the DataStructure and DataField classes. You want to specify that the DataField class is a child class of the DataStructure class. To establish a parent-child relationship between the classes, first create the DataStructure class. Then select the DataStructure class and create the DataField class. The DataField class becomes the child, or subclass, of the DataStructure class.

Class-Level Relationships

Class-level relationships are associations between a custom class and a different custom class or between a custom class and a class in a packaged or universal model. View and create class-level relationships for custom classes on the **Model** tab.

You must create class-level relationships before you can perform the following tasks:

- Add related catalog objects for a custom object.
- Create object-level relationships through the Object Relationships Wizard.
- Create lineage associations between custom objects and other objects through the Custom Metadata Configurator.

You do not have to create class-level relationships to create lineage associations between custom objects and other objects through linking rules or enumerated links.

Create class-level relationships so that you can define relationships between individual metadata objects. For example, you create a custom model that contains the DataStructure class. You want to relate objects of the DataStructure class with objects of the Oracle Table class. First create a class-level relationship between the DataStructure class and the Oracle Table class. You can then create object-level relationships between any object of the DataStructure class and any Oracle table.

Object-Level Relationships

Object-level relationships are associations between two metadata objects. You can create object-level relationships for groups of objects or for individual objects.

Create object-level relationships for multiple custom objects through linking rules files, enumerated links files, the Custom Metadata Configurator, or the Object Relationships Wizard. Create object-level relationships for individual custom objects when you edit the related catalog objects in the object properties.

View relationships between metadata objects on the **Browse** tab. Metadata Manager displays object-level relationships in the object properties. If the classes to which the objects belong are enabled for data lineage, Metadata Manager also displays the object-level relationships in the data lineage diagram.

Business Name

A business name is a property used to identify metadata objects according to their business usage instead of the metadata object name in the metadata catalog. For example, a table named CUST_ADDR identifies a customer shipping address. Configure the business name property as "Customer Shipping Address" to indicate that the table contains the customer shipping address.

All classes, except classes in business glossary models, include a business name property. Packaged and universal models do not populate the business name property.

Edit business name properties for metadata objects on the **Browse** tab. To edit business name properties for multiple metadata objects in a packaged or universal resource, export a subtree of the resource to an Excel file, edit the property values, and import the properties from the Excel file into the metadata catalog.

View the business name property for metadata objects in the **Details** panel on the **Browse** tab.

Model Browsing and Editing

Browse and edit the models in the Metadata Manager repository from the **Model** tab. The **Model** tab displays all types of models.

The **Model** tab includes the following components:

Models navigator

Displays all models in the Metadata Manager repository in a hierarchical structure.

Content panel

Displays child classes, folders, and groups of models and classes that you select in the **Models** navigator.

Properties panel

If you select a model, this panel displays the model name, description, and rule set definitions associated with the model. If you select a class, this panel displays the class details, properties, and relationships.

Browsing Models

You can browse the model hierarchy, view model and class properties, and view the rule set definitions associated with the model.

Use the **Model** tab to complete the following tasks:

Browse the model hierarchy.

Browse all models and view the model hierarchy in the **Models** navigator.

View model and class properties.

View the hierarchy for a specific class and view the model and child classes and properties in the **Content** and **Properties** panels. Model properties include name and description. Class properties include the name, description, group name, and whether instances of the class show in a lineage diagram. Packaged and universal models might include folders in the model hierarchy.

You can view details for multiple models and classes simultaneously. Each time you open a class, Metadata Manager opens a tab for the class.

View rule set definitions.

View the rule set definitions associated with a model on the **Rule Set Definitions** tab in the **Properties** panel. The **Rule Set Definitions** tab displays information about the rule set definition such as the source model, target model, whether the rule set definition is valid, and whether the rule set definition is associated with a pair of resources. You must upload a rule set parameter file for a resource to associate the rule set definition with a pair of resources.

Editing Models

You can edit model and class properties for custom models and classes. You can add custom properties for classes. You can also add and remove rule set definitions for models.

Use the **Model** tab to perform the following tasks:

Create custom models and edit model properties.

Create or edit a custom model from the **Models** navigator.

Create and edit classes, class properties, and relationships.

Use the **Content** and **Properties** panels to perform the following tasks:

- Create and edit custom classes, properties, and relationships.
- Create classes for custom models. You cannot create classes for packaged or universal models.
- View properties and relationships for classes in packaged and universal models.
- Add custom properties for classes in all models except business glossary models.
- Organize the way in which you display class properties.

Note: Class properties are called attributes in the **Model** tab.

Upload and remove rule set definitions.

Upload and remove rule set definitions for a model on the **Rule Sets** tab in the **Properties** panel.

AccessDB Example

This book uses an example of a Microsoft Access database to show the concepts for custom metadata integration. The Access database contains tables and views. It includes the model, AccessDB, and the following classes:

- AccessSchema
- AccessTable

- AccessTableColumn
- AccessView
- AccessViewColumn

AccessSchema

Schema class for the tables and views in the Access database. The class is not configured to display in a data lineage diagram. AccessSchema is the root class for the AccessDB model.

The following table describes the properties for this class:

Property	Datatype
Name	String
Description	String

AccessTable and AccessView

Table and view classes for the Access database. The classes are configured to display in data lineage. AccessTable and AccessView are child classes of the AccessSchema class. Group names are Tables and Views.

The following table describes the properties for the class:

Property	Datatype
Name	String
Description	String
Abbreviation	String
Business Usage	String
Date Created	Date

AccessTableColumn and AccessViewColumn

Column class for the Access database. AccessTableColumn and AccessViewColumn are configured to display in data lineage and are child classes of AccessTable and AccessView.

The following table describes the properties for the class:

Property	Datatype
Name	String
Description	String
Abbreviation	String

Property	Datatype
Business Usage	String
Datatype	String
Length	Integer
Date Created	Date

CHAPTER 2

Creating and Configuring Custom Models

This chapter includes the following topics:

Creating and Configuring a Custom Model Overview

Create and configure custom models to define the custom metadata you want to add to the Metadata Manager warehouse. You can create and edit custom models. You can create and edit custom properties for classes in universal models and all packaged models except business glossary models.

To create and configure a custom model, complete the following steps:

1. Create the model.
2. Configure classes.
3. Configure the class properties.
4. Configure the class relationships.

Note: You cannot create a custom model using the packaged business glossary model as template.

After you create and configure a model, add or load the metadata through one of the following methods:

Load Template

Create a load template that contains rules for mapping custom metadata to the model. Create a custom resource on the **Load** tab in Metadata Manager, and apply the mapping rules to metadata source files. When you load the resource, Metadata Manager extracts metadata from the files and loads it into the Metadata Manager warehouse.

Custom Metadata Configurator

Use the Custom Metadata Configurator to create a custom resource template to load the metadata.

Browse Tab

Add the metadata objects and relationships through the metadata catalog on the **Browse** tab in Metadata Manager.

Establish the Model Structure

Before you begin to create the model, you must establish the required classes and subclasses, and the properties and relationships between the classes.

Identify the following components of the model:

Model name and description

The model serves as the container for the classes in the model. It appears as a resource type in the metadata catalog when you add or load metadata. For example, AccessDB is the model name for the Access database example.

Parent classes and subclasses

The classes represent the metadata objects that you want to add or load into the Metadata Manager warehouse. For example, AccessSchema is the root class for the Access database example. AccessTable and AccessView are the child classes of the AccessSchema class.

Properties and relationships

Determine the properties for each class and the relationships between the custom metadata classes and classes in packaged or universal models. For example, in the Access database example, the columns in the AccessTable schema have a relationship to source columns in an Oracle database.

Configuring a Custom Model

You can create, edit, or delete a custom model. You can also upload rule set definitions if you use rule-based links for pairs of models.

When you create or edit a model, you configure the model name, the description, and the rule set definitions. You can use a previously created custom model as a template for the new custom model.

Creating a Model

Create a model and configure the name, the description, and optionally, the template to base the model on.

1. On the **Model** tab, click **Actions** > **New** > **Model**.

   The **New Model** window appears.
2. Enter the name and description for the model.

   The model name cannot include the following characters:

   `/ \ : * ' ? " < > | [ ]`
3. Optionally, select a model to use as a template.
4. Click **OK**.

Metadata Manager creates the model. The model appears in the **Models** navigator.

Editing a Model

You can edit a custom model to change the name and description for the model. If you change a model name, Metadata Manager updates the model name in the **Resource** wizard on the **Load** tab.

Note: You cannot edit the model name or the description for a packaged model or a universal model.

1. On the **Model** tab, select the model that you want to edit.
2. In the **Properties** panel, click the **Edit** icon.

   The **Edit Model** window appears.
3. Update the name or description.

   The name cannot include the following characters:

   `/ \ : * ' ? " < > | [ ]`
4. Click **OK**.

Uploading Rule Set Definitions

If you use rule-based links to define linking rules for a pair of models, you must upload the rule set definition file for the source model or the target model.

Upload the rule set definition file to create or update the rule set definition in the Metadata Manager repository. You can upload multiple rule set definition files for a model.

1. On the **Model** tab, select a model.
2. Click **Actions** > **Upload Rule Set Definition**.

   The **Upload Rule Set Definition** dialog box appears.
3. Click **Browse**, select the rule set definition XML file, and click **Open**.
4. Click **OK**.

   Metadata Manager uploads the file and creates or updates the rule set definition. It also validates the rule set definition. If the rule set definition is not valid, check the service log for more information.

The **Rule Set Definitions** tab in the **Properties** panel displays the rule set definition information for the model. To remove a rule set definition, select the rule set definition, and click **Delete**.

Deleting a Model

You can delete any custom model you create. When you delete a model, Metadata Manager removes any classes, properties, and associations for the model.

You cannot delete a model if you added metadata based on the model. If you loaded metadata using the model, or if you added metadata to the metadata catalog based on the model, purge the metadata from the Metadata Manager warehouse or delete the resource in the metadata catalog before you delete the model.

To delete a model:

1. Select the model that you want to delete in the Model navigator.
2. Click Actions > Delete.
3. Click OK.

Metadata Manager deletes the model and all classes for the model from the Model navigator and the Metadata Manager repository.

Configuring Classes

You can create, edit, or delete classes for custom models. When you create a class, you select the level in the model hierarchy where you want to create the class and configure the class properties. You can edit and delete classes and change the location of the class in the model hierarchy.

The following table describes the class properties that you can configure:

Property	Description
Class Name	Name of the class. Do not use the INPUT or OUTPUT reserved words as the class name. The PowerCenter workflow fails to generate.
Description	Description for the class.
Group Name	Name of the group under which metadata objects of this class appear in the metadata catalog. Select a group name when you create a class. After you create the class, you cannot edit the group name.
Icon File Name	Name of the graphic file you want to use as the icon for the class. The icon appears in the Model navigator for the class and in the metadata catalog for any objects based on the class. You must place the graphic file in the following location: `<Informatica installation directory>\ services\MetadataManagerService\mmapps \mm\images`
Show in Lineage	Enables data lineage for metadata objects of this class. If you do not enable this property, Metadata Manager does not display data lineage for metadata objects of this class. Default is disabled.

Creating a Class

Create a class at any hierarchy level for a custom model. Optionally, you can add a previously created class.

For example, in the Access database example, you want to add an AccessTableColumn class as a subclass to the AccessTable class. You must select the AccessTable class to create a subclass for AccessTable.

To create a class:

1. On the Model tab, select the model or class under which you want to create the class.
2. If you select the model or class in the Model navigator, click Actions > New > Class.

   The New Class window appears.

   If you selected a model in the Model navigator, a different New Class window appears.
3. Enter the class properties.
4. Click OK.

Metadata Manager creates the class at the selected level of the hierarchy. After you create the class, configure the class properties and relationships.

Editing a Class

You can edit a class to change the name, description, icon file name, or lineage properties. If you edit a class to show the metadata objects for the class in data lineage, you must load the resource again to view the metadata objects for the class in the data lineage diagram.

To edit a class:

1. On the Model tab, select the class in the Model navigator or Content panel.
2. In the Properties panel, click the Edit icon.

   The Edit Class window appears.
3. Edit the class properties.
4. Click OK.

Metadata Manager updates the class with the changed properties.

Deleting a Class

You can delete a class from a model. If you delete a class and the metadata catalog contains a metadata object for the class, you cannot add more classes of the same type to the catalog. The metadata objects for the class will remain in the catalog.

Note: You cannot delete a class if it has a relationship to another class.

To delete a class:

1. On the Model tab, select the class in the Model navigator or the Content panel.
2. Click Actions > Delete.

Moving and Copying Classes

To move or copy a class from one level of the model hierarchy to another in the Model navigator, drag the class to another location.

Use the following rules and guidelines when you move or copy classes:

- If you drag a class from one level of the hierarchy to another for the same model, Metadata Manager moves the class.
- If you drag a class from one custom model to another custom model, Metadata Manager copies the class.
- If you move or copy a class, Metadata Manager moves or copies all subclasses for the class.
- You cannot drag classes into packaged or universal models.

To move a class:

1. In the Model navigator, select the class you want to move or copy.
2. Drag the class to another location.

   If the location to which you want to move or copy the class is collapsed, move the pointer over the parent class to expand the child classes in the navigator.

Configuring Properties

You can configure properties for custom classes and for classes in all other models except business glossary models. You can create and edit properties that apply to all metadata objects for a custom class or a packaged class. You can delete properties that apply to all metadata objects for a custom class.

To configure properties, open the class on the **Model** tab from the model navigator or the **Content** panel.

Note: You cannot configure properties for classes in the business glossary model. To customize business glossaries, use the Analyst tool.

The following table describes the properties:

Property	Description
Name	Name of the property. This name appears for all metadata objects for the class in the metadata catalog when you add or load metadata.
Description	Description of the property.
Type	Data type for the property. You can configure the following datatypes: - Integer - String - Long - LongString - Date

After you configure the properties, use one of the following methods to configure the values for the properties based on the method you used to add or load the metadata:

Added manually in the Browse tab

Use the **Browse** tab to edit values for the custom properties.

Loaded with a custom XConnect

If you create a custom resource to load the metadata, you configure the values that Metadata Manager uses for the properties in the **Attributes Map** tab of the Custom Metadata Configurator or in the load template.

You can also change the order in which properties appear for objects on the **Browse** tab. To change the order of a property, click the property and drag it to a different location.

Creating Properties

When you create properties, you configure the name, description, and datatype for the property. By default, Metadata Manager includes Name, Label, Description, and Business Name properties for each class.

1. On the Model tab, select the class for which you want to configure properties.
2. In the Model navigator or the Content panel, click Actions > Open.

   Metadata Manager opens a tab for the class and lists the default properties for the class.
3. Click Actions > Add Attribute and configure the name, description, and datatype of the property.

   The datatype can be Integer, Long, String, Date, or LongString.
4. Click Save.

Editing Properties

You can edit properties for classes in all models except for business glossary models.

If you change the name of a property for which metadata objects exist in the metadata catalog, Metadata Manager updates the objects to use the property name. If you change the type of a property for which

metadata objects exist in the metadata catalog, Metadata Manager does not update the metadata to use the data type. You must delete the value for type and enter a value that is valid for the data type.

1. On the **Model** tab, select the class for which you want to configure properties.
2. In the Models navigator or the Content panel, click **Actions** > **Open Selected Class**.

   Metadata Manager opens a tab for the class and lists the properties in the **Attributes** view.
3. To edit the properties, click the **Name**, **Description**, or **Type** box.

   The data type can be Integer, Long, String, or Date. For custom properties, the data type can also be LongString.
4. To organize properties, drag them to change their order or to ensure that they appear in either the Basic or Advanced section of the class properties, in all Metadata Manager perspectives. You can also use **Actions** > **Move Up** or **Actions** > **Move Down** to change the order of the class properties.
5. Click **Save**.

Organizing Properties

You can organize the way in which you want to display the class properties.

When you edit the properties, you can drag them to change their order or to ensure that they appear in either the Basic or Advanced section of the class properties, in all Metadata Manager perspectives.

If you move the properties below the properties divider, users need to click More in the Properties pane to view these properties in the Advance section. Users need to click Less if they want to hide the properties listed in the Advanced section.

For a class, Source Creation Date, Source Update Date, MM Creation Date, and MM Update Date properties are referred as the synthetic date properties. You can set the Show_Synthetic_Dates_In_Basic_Section property in the imm.properties file to specify if these properties should be located in the Basic or Advanced section.

Synthetic date properties are always grouped together and you cannot change their order. You cannot move any property below the synthetic date properties.

The Class and Location properties are grouped together and you cannot move them below the properties divider.

The Name and Description properties are grouped together and you cannot move any other property above these properties.

Deleting Properties

You can delete properties for custom classes and custom properties you added for packaged and universal model classes. You cannot delete the default **Name** and **Description** properties for any class.

If you added metadata for the property to any metadata object in the metadata catalog, Metadata Manager also removes the metadata for the property. For example, you added a custom property to the OracleTable class and then added the values for the property on the **Browse** tab. If you delete the property, Metadata Manager removes the property from all metadata objects of class OracleTable in the metadata catalog.

1. On the **Model** tab, select the class for which you want to delete a property.
2. In the **Models** navigator or the **Content** panel, click **Actions** > **Open**.

   Metadata Manager opens a tab for the class and lists the properties for the class.
3. Select the property and click **Actions** > **Delete Attribute**.

Configuring Relationships

Configure relationships so that you can create associations between objects. You might have to create class-level relationships before you can create and configure object-level relationships.

You must create class-level relationships if you want to perform the following tasks:

- Add related catalog objects for a custom object.
- Create object-level relationships through the Object Relationships Wizard.
- Create lineage associations between custom objects and other objects through the Custom Metadata Configurator.

You do not have to create class-level relationships to create lineage associations between custom objects and other objects with linking rules or enumerated links.

Create class-level relationships on the **Model** tab. You can create class-level relationships between custom classes or from custom classes to classes in packaged or universal models. When you create a class-level relationship, you configure properties such as the relationship name, model and class to which you want to relate the custom class, and the direction of data flow for lineage.

Note: You cannot create class-level relationships for the business glossary model. To customize business glossaries, use the Analyst tool.

Create object-level relationships through the following methods:

Load template

If you use the load template method to load custom metadata, you can create object-level relationships for multiple objects through linking rules files or enumerated links files. Linking rules files define the rules that Metadata Manager uses to link objects in a custom resource with other objects. Enumerated links files define pairs of objects to link within a custom resource or across resources.

Specify linking rules files or enumerated links files on the **Load** tab when you create or edit a custom resource.

Custom Metadata Configurator

If you use the Custom Metadata Configurator to load the custom metadata, establish object-level relationships for multiple objects on the **Associations** tab of the Custom Metadata Configurator.

Browse tab

Use the Object Relationships Wizard on the **Browse** tab to create relationships from multiple metadata objects. Use the **Related Catalog Objects** section on the **Browse** tab to create individual object-level relationships.

Class-Level Relationship Properties

When you create a relationship, you configure properties such as the relationship name, model and class to which you want to relate the custom class, and the direction of data flow for lineage.

The following table describes the properties for class-level relationships:

Property	Description
Name	Name of the relationship.
Description	Description of the relationship.

Property	Description
Model	Model of the class to which you want to create the relationship. Use the **Any Model** option for this property for the most flexibility in creating object-level relationships to objects of this class.
Class	Class to which you want to create the relationship. Use the **Any Class** option for this property for the most flexibility in creating object-level relationships to or from objects of this class.
Direction	Direction of data flow for lineage associations between objects of the two related classes. Select one of the following options: - **>>**. Specifies a downstream data flow for lineage. Data flows from the custom class that you are editing into the related class. - **<<**. Specifies an upstream data flow for lineage. Data flows from the related class into the custom class that you are editing. **Note:** If the **Show in Lineage** property for the custom class that you are editing is disabled, Metadata Manager ignores this property.

Creating Class-Level Relationships

When you create a class-level relationship, you configure the model and class to which you want to create the relationship.

To create a relationship:

1. On the Model tab, select a class.
2. In the Model navigator or the Content panel, click Actions > Open.

   Metadata Manager opens a tab for the class.
3. Select Relationships.
4. Click Actions > Add Relationship and configure the relationship properties.
5. Click the Save icon.

Editing and Removing Class-Level Relationships

Edit a relationship to change any of the properties or delete a relationship. If you edit a relationship and change the model, class, or parent-child relationships, Metadata Manager does not remove object-level relationships. The object-level relationships that you create use the updated properties.

For example, you edit the relationship for the AccessTableColumn class and remove the OracleColumn class from the relationship. Existing object-level relationships in the metadata catalog do not change, but you cannot create additional object-level relationships from AccessTableColumn objects to the OracleColumn metadata objects.

If you remove a relationship, Metadata Manager does not remove object-level relationships.

To edit a relationship:

1. On the Model tab, select a class.
2. In the Model navigator or the Content panel, click Actions > Open.

   Metadata Manager opens a tab for the class.
3. Select Relationships.
4. Edit the relationship properties.
5. Click the Save icon.

6. Optionally, select the relationship and click Actions > Delete Relationship.

Part II: Custom XConnect Created with a Load Template

This part contains the following chapters:

CHAPTER 3

Load Templates

This chapter includes the following topics:

Load Templates Overview

The load template defines how to load metadata from metadata source files into a custom resource. The load template consists of mapping rules that map the metadata in the metadata source files to model components, like classes, class attributes, associations, and lineage associations. You can use one load template for multiple resources of the same model.

You can use mmcmd to generate a default load template. The default load template contains all the mapping rules required to load all attributes of the top-level classes. For all child classes, the default load template contains parent-child associations.

When you generate the default load template, you also generate an empty metadata source file for each class in the model. The metadata source files contain column headers that are referenced in the default load template.

After you generate the default load template and metadata source files, you must populate the metadata source files with the custom metadata. In addition, you can configure mapping rules in the default load template based on the column headings in the metadata source file.

Load Template Process

You create a load template to define how to load metadata from metadata source files into a custom resource. Use mmcmd to generate a default load template. After you generate the default load template, you can update it based on the content of the metadata source files.

To create a load template, perform the following steps:

1. Generate the default load template file using the mmcmd generateDefaultLoadTemplate command.

   When you generate the default load template, by default, you also generate metadata source files based on the defined custom model. The generated files do not contain any custom metadata.
2. Populate the metadata source files with custom metadata.
3. If required, update the default load template.

   You might need to update the load template to create additional associations or lineage associations, or to change the references to column headers in the metadata source file that changed when you populated the custom metadata.

After you create the load template, upload it on the **Model** tab in Metadata Manager.

Note: You can also manually create a load template based on the model.

Metadata Source Files

A metadata source file contains custom resource metadata that you want to load into the Metadata Manager warehouse. You can have more than one metadata source file for a custom resource. For example, you might have multiple metadata source files if you stored information about different classes of objects in separate files. Metadata source files are stored in a CSV format. The metadata source files are also called the CSV files.

The metadata source files can contain the following type of metadata:

Metadata object names and attributes

A metadata object is an instance of the class that you defined in the model. You map the object attributes in the metadata source file to the class attributes defined in the corresponding model.

Object associations

An object association is a relationship between two metadata objects. The associations between metadata objects are instances of the class associations that you defined in the model.

Lineage associations

A lineage association between two classes of objects enables data lineage between those objects. In a data lineage diagram, Metadata Manager displays a lineage association as an arrow between associated objects.

You generate metadata source files when you generate the default load template. If you want to create your own metadata source files, you can export the custom metadata from the custom source repository. You can also create an extraction command file that exports the metadata to metadata source files each time you run the custom XConnect.

Note: If you run a custom XConnect with an extraction command file and the batch file fails, the XConnect also fails. Metadata Manager writes output and error logs to the resource load log.

Regardless of how you create the metadata source files, ensure that the column headers in the metadata source files match the references to the column headers in the load template.

Configuring an Extraction Command File to Generate Metadata Source Files

You can use an extraction command file to generate and populate the metadata source files before each metadata load into the Metadata Manager warehouse.

Upload the extraction command file when you create the custom resource. Before you can upload an extraction command file, you must set a custom property for the Metadata Manager Service.

Use the Administrator tool to set the following custom property for the Metadata Manager Service:

Property	Value
mm.allow.preload.command	true

Restart the Metadata Manager Service after you set this property.

Metadata Source File Rules and Guidelines

Use the following rules and guidelines when you create a metadata source file:

- A metadata source file can contain multiple rows for a single element.
- A metadata source file can contain row and column delimiters. For example, the metadata source file can contain commas to separate column values and newline characters to separate rows.
- Each metadata source file can have a different encoding, such as UTF-8, UTF-16, and ASCII.
- A metadata source file can contain metadata in different locales, such as en-us and ja-jp.
- The column headers in the metadata source files can contain letters, numbers, and underscore characters.
- Each column header must start with a letter or underscore.
- The name and the business name of each metadata object in a metadata source file must contain 255 or fewer characters. If the name or business name contains more than 255 characters, Metadata Manager truncates the string to the first 255 characters.
- If you do not specify a date format in the load template, date values in a metadata source file must be in the format `yyyy/MM/dd`.

Special Characters in Metadata Source Files

You can enter special characters such as double quotation marks and newline characters within a string in a metadata source file.

You can enter the following special characters within a string:

Double quotation marks (")

To enter double quotation marks within a string, use the tilde character (~) as an escape character.

For example, to specify the description for a class as `Shortcut to "Class 1"`, enter the following string in the metadata source file:

```
Shortcut to ~"Class 1~"
```

Newline character

To enter a newline character within a string, replace the newline character with `<br/>`.

Date Formats in Metadata Source Files

You can specify the format for dates in a metadata source file. To specify the date format, set the dateFormat attribute in the loadTemplate element of the load template. You can specify any valid Java date format.

The following table describes the characters that you use to construct a date format:

Character	Description
G	Era designator, for example, AD.
y	Year: - Enter `yy` for two-digit year values. For example, 14. - Enter `yyyy` for four-digit year values. For example, 2014. Metadata Manager interprets two-digit year values to be within 80 years before and 20 years after the date the resource is loaded. For example, if you load the resource on Jan. 1, 2015, Metadata Manager interprets the year "30" as 2030, however, it interprets the year "40" as 1940.
Y	Week year: - Enter `yy` for two-digit year values. For example, 98. - Enter `yyyy` for four-digit year values. For example, 1998. Metadata Manager interprets two-digit year values to be within 80 years before and 20 years after the date the resource is loaded.
M	Month in year: - Enter `MM` for numeric, two-digit month values. For example, 07. - Enter `MMM` for three-character abbreviations for the month. For example, Jul. - Enter `MMMM` for the full names of the month. For example, July. Metadata Manager does not accept abbreviated month values that are not three characters in length. For example, Metadata Manager does not accept "Sept" for September.
w	Numeric week in year.
W	Numeric week in month.
D	Numeric day in year.
d	Numeric day in month.
F	Numeric day of week in month, for example, 2 for the second Wednesday in June.
E	Day name in week. Metadata Manager accepts three-character abbreviations for the day or the full name of the day, for example, "Tue" or "Tuesday." Metadata Manager does not accept abbreviations that are not three characters in length. For example, Metadata Manager does not accept "Thurs," "R," or "Th" for Thursday.
u	Numeric day in week, for example, 1 for Sunday.
a	AM/PM designator, for example, AM. Metadata Manager does not accept period characters in the AM/PM designator. For example, Metadata Manager does not accept "a.m." or "p.m."
H	Numeric hour in day (0-23).
k	Numeric hour in day (1-24).

Character	Description
K	Numeric hour in AM/PM (0-11).
h	Numeric hour in AM/PM (1-12).
m	Numeric minute in hour.
s	Numeric second in minute.
S	Numeric millisecond.
z	General time zone: - Enter `zzz` for the abbreviated name of the time zone. For example, PST. - Enter `zzzz` for the full name of the time zone. For example, Pacific Standard Time.
Z	RFC 822 time zone, for example, -0800.
X	ISO 8601 time zone: - Enter `X` for the sign and two-digit hours. For example, -08. - Enter `XX` for the sign, two-digit hours, and minutes. For example, -0800. - Enter `XXX` for the sign, two-digit hours, a colon (`:`), and minutes. For example, -08:00.
'	Escape character for alphabetic characters.
''	Single quote character, for example: `'`

For numeric values, Metadata Manager ignores the number of format characters unless the number of characters is needed to separate adjacent numeric fields.

Examples

The following example shows how to set the dateFormat attribute for date values such as 12/25/2002:

```
dateFormat="MM/dd/yyyy"
```

The following example shows how to set the dateFormat attribute for date values such as 05-Feb-96:

```
dateFormat="dd-MMM-yy"
```

The following example shows how to set the dateFormat attribute for date values such as Fri 2014.07.18 at 04:14:09 PM PDT:

```
dateFormat="E yyyy.MM.dd 'at' hh:mm:ss a zzz"
```

The following example shows how to set the dateFormat attribute for date values such as 1996-03-14T12:08:56.235-07:00:

```
dateFormat="yyyy-MM-dd'T'HH:mm:ss.SSSXXX"
```

The following example shows how to set the dateFormat attribute for date values such as 990731120856-0800:

```
dateFormat="yyMMddHHmmssZ"
```

Related Topics:

- "Load Template Elements" on page 40

Load Template Components

The load template is an XML file that consists of multiple XML elements. Each element describes a component of the load template.

The load template contains the following components:

Mapping rules

Rules that map the content of a metadata source file to the model. You can create one mapping rule for each class. You can group mapping rules into a mapping rule set. You create the rule definitions in the load template. When you configure the resource, you can apply the mapping rules and mapping rule sets to the appropriate metadata source files.

Conditions

Conditions that filter the rows that are loaded from the metadata source file for a specific class, class attribute, association, or lineage association.

Classes

Classes that map to objects in the metadata source file.

Properties

Class properties that map to object attributes in the metadata source file.

Associations

Associations between each class of objects and another class of objects. You can create more than one association for each class of objects.

Lineage associations

Associations between two classes of objects that enable data lineage between the two sets of objects. The objects must be within the same resource. You can use rule-based links to enable data lineage between objects from different resources.

Load Template Elements

The load template contains XML elements that describe the load template and define mapping rules. Each type of mapping rule uses a specific set of elements.

The load template XML contains the following element that describes the load template:

loadTemplate

The following table describes the attributes for the loadTemplate element:

Attribute	Description
name	Required. Name of the load template.
targetModel	Required. Name of the model with which you want to associate the load template.

Attribute	Description
templateSpecVersion	Required. Version of the load template.
dateFormat	Optional. Format for date values in all metadata source files that use the load template. Must be a valid Java date format. If you specify a date format, dates in the metadata source file must be in the date format you specify or in the format `yyyy/MM/dd`. If you do not specify a date format, all dates in the metadata source files must be in the format `yyyy/MM/dd`.

The following example shows a loadTemplate element configured for a custom resource:

```
<loadTemplate name="AS400_1" targetModel="AS400" templateSpecVersion="1.0"
dateFormat="dd-MM-yyyy">
```

Related Topics:

- "Date Formats in Metadata Source Files" on page 38

Elements in a Class Rule

You can add specific elements when you define a class rule. You can add an element that uniquely identifies each object in the class. You can also add elements that map object properties to class attributes and create associations and lineage associations between objects.

You can use the following elements in a class rule:

class

The following table describes the attributes for the class element:

Attributes	Description
ruleName	Required. Name of the rule defined for the specified class.
name	Required. Name of the class.
condition	Optional. Condition to apply while loading metadata for the specified class.

id

ID that uniquely identifies each object in the specified class. You can create an expression to uniquely identify each object. The id element does not contain any attributes.

property

To map an object property in the metadata source file to a class attribute, add the property element within the properties element in the class rule.

The following table describes the attribute for the property element:

Element Attribute	Description
name	Required. Name of a class attribute defined in the model. Sample class attributes include name, infa_label, infa_description, and businessName. Map the class attribute to the appropriate column name in the metadata source file.

For example, you create the following class rule:

```
<class condition="" ruleName="CustomClassRule" name="CustomClass">
  <id>ID</id>
  <properties>
    <property name="name">NAME</property>
    <property name="infa_label">LABEL</property>
    <property name="infa_description">DESCRIPTION</property>
    <property name="businessName">BUSINESS_NAME</property>
    <property name="Business Rule">BUSINESS_RULE</property
  </properties>
  <associations>
  </associations>
  <lineageAssociations>
  </lineageAssociations>
</class>
```

In this example, the name, infa_label, infa_description, and businessName properties correspond to the default class attributes Name, Label, Description, and Business Name for class CustomClass. The Business Rule property corresponds to a custom class attribute called “Business Rule” that is defined for class CustomClass. The metadata object names come from the NAME column in the metadata source file. Other object attributes come from the LABEL, DESCRIPTION, BUSINESS_NAME, and BUSINESS_RULE columns.

association

You can create associations between objects of the class specified in the class rule and objects of other classes. The child object is of the class specified in the class rule. The parent object is the object specified in the parent element within the association element. To add multiple associations for a class, add multiple association elements within the associations element. The association element does not contain any attributes.

parent

Use the parent element to identify the parent object in a parent-child association. Add the parent element within the association element defined in a class rule.

For example, you create the following class rule:

```
<loadTemplate name="AS400_1" targetModel="AS400" templateSpecVersion="1.0">
<class ruleName="AS400TableRule" name="AS400Table" condition="TYPE='TABLE'">
  <id>PARENT+'~'+ELEMENT</id>
  <associations>
    <parent>PARENT</parent>
  </associations>
```

In this example, the class rule defines the associations between child objects and parent objects. The child objects come from the ELEMENT column in the metadata source file and are objects of the AS400Table class. The parent objects come from the PARENT column in the metadata source file.

lineageAssociation

You can create lineage associations between objects of the class specified in the class rule and objects of other classes. To add multiple lineage associations for a class, add multiple lineageAssociation elements within the lineageAssociations element.

The following table describes the attribute for the lineageAssociation element:

Element Attribute	Description
direction	Required. Identifies the direction in which data flows between two objects. The direction enables Metadata Manager to display data lineage. Valid values include 'to' and 'from.' You must also specify the target element to which the lineage association applies.

targetElement

The targetElement element identifies the downstream object in a lineage association between two objects. The lineage association represents data flowing from the source object to the target object. Use the targetElement element to define a lineage association between other objects and a specific class of objects. To create the lineage association, add the targetElement element within the lineageAssociation element defined in the class rule.

For example, you create the following class rule:

```
<loadTemplate name="AS400_1" targetModel="AS400" templateSpecVersion="1.0">
<class ruleName="AS400TableRule" name="AS400Table" condition="TYPE='TABLE'">
  <id>PARENT+'~'+ELEMENT</id>
    <lineageAssociations>
    <lineageAssociation direction="to">
      <linkExpression>LINK_EXPRESSION</linkExpression>
      <targetElement>TABLE</targetElement>
    </lineageAssociation>
  </lineageAssociations>
```

In this example, the lineage association rule defines lineage associations from source objects to target objects. The source objects come from the ELEMENT column in the metadata source file and are objects of the AS400Table class. The target objects come from the TABLE column in the metadata source file.

linkExpression

The linkExpression element specifies the expression text that appears in the data lineage diagram when you hover over the target object.

For example, you create the following lineage association rule:

```
<lineageAssociation ruleName="AS400ViewLinkRule">
  <linkExpression>LINK_EXPRESSION</linkExpression>
  <from>FROM_ELEMENT</from>
  <to>TO_ELEMENT</to>
</lineageAssociation>
```

In this example, the lineage association rule defines lineage associations between source objects in the FROM_ELEMENT column of the metadata source file and target objects in the TO_ELEMENT column of the metadata source file. Text in the LINK_EXPRESSION column appears in the data lineage diagram when you hover over the target objects.

Elements in a Property Rule

You can add specific elements when defining a property rule. You can add multiple name-value pairs to identify each object property and it's value.

You can use the following elements in a property rule:

property

The following table describes the attributes for the class element:

Attributes	Description
ruleName	Required. Name of the rule defined for the specified property.
element	Required. Name of the element to which the property belongs.

name

You can specify the name of the column in the metadata source file that contains the property name.

value

You can specify the name of the column in the metadata source file that contains the value of the property.

Elements in an Association Rule

You can add specific elements when defining an association rule. You must specify the "from" element and "to" element to identify the "from" and "to objects in the association.

You can use the following elements in an association rule:

association

The following table describes the attributes for the association element:

Attributes	Description
ruleName	Required. Name of the rule defined for the specified association.
name	Required. Name of the association.

from

You can add the "from" element within the association element in an association rule. The "from" element identifies the "from" object in the association. You must also specify the "to" element.

For example, you create the following association rule:

```
<loadTemplate name="AS400_1" targetModel="AS400" templateSpecVersion="1.0">
<association ruleName="AS400ParentRule" name="ParentAssociation">
<from>FROM_ELEMENT</from>
<to>TO_ELEMENT</to>
</association>
```

In this example, the association rule defines associations between "from" objects and "to" objects. The "from" objects come from the FROM_ELEMENT column in the metadata source file. The "to" objects come from the TO_ELEMENT column in the metadata source file.

to

You can add the "to" element within the association element in an association rule. The "to" element identifies the "to" object in the association. You must also specify the "from" element.

Elements in a Lineage Association

You can add specific elements when defining a lineage association rule. You must specify the "from" element and "to" element to identify the "from" and "to" objects in the association. You can also specify a linkExpression element to define the expression that appears in the data lineage diagram when you hover over the "to" object.

You can use the following elements in a lineage association rule:

lineageAssociation

The following table describes the attributes for the lineageAssociation element:

Attributes	Description
ruleName	Required. Name of the rule defined for the specified lineage association.

from

You can add the "from" element within the association element in an association rule or add the "from" element within the lineageAssociation element in a lineage association rule. The "from" element identifies the "from" object in the association. You must also specify the "to" element.

For example, you create the following association rule:

```
<loadTemplate name="AS400_1" targetModel="AS400" templateSpecVersion="1.0">
<association ruleName="AS400ParentRule" name="ParentAssociation">
  <from>FROM_ELEMENT</from>
  <to>TO_ELEMENT</to>
</association>
```

In this example, the association rule defines associations between "from" objects and "to" objects. The "from" objects come from the FROM_ELEMENT column in the metadata source file. The "to" objects come from the TO_ELEMENT column in the metadata source file.

to

You can add the "to" element within the association element in an association rule or add the "to" element within the lineageAssociation element in a lineage association rule. The "to" element identifies the "to" object in the association. You must also specify the "from" element.

linkExpression

The linkExpression element specifies the expression text that appears in the data lineage diagram when you hover over the "to" object.

For example, you create the following lineage association rule:

```
<lineageAssociation ruleName="AS400ViewLinkRule">
  <linkExpression>LINK_EXPRESSION</linkExpression>
  <from>FROM_ELEMENT</from>
  <to>TO_ELEMENT</to>
</lineageAssociation>
```

In this example, the lineage association rule defines lineage associations between source objects in the FROM_ELEMENT column of the metadata source file and target objects in the TO_ELEMENT column of the metadata source file. Text in the LINK_EXPRESSION column appears in the data lineage diagram when you hover over the target objects.

Load Template Conditions

You can add conditions in a load template to filter the rows of the metadata source file that apply to the mapping rule. You can apply conditions to mapping rules for the following objects: classes, class attributes, associations, and lineage associations.

Use load template expressions to create a condition. For example, you have a metadata source file that contains rows for table and column objects. The TYPE field of the metadata source file specifies the type of object for each row. You want to create a lineage association between the columns only. You create the following lineage association:

```
<lineageAssociation ruleName="linkRule" condition="TYPE='COLUMN'">
        <from>PARENT</from>
        <to>ELEMENT</to>
    </lineageAssociation>
```

When you run the custom XConnect, Metadata Manager retrieves only rows from the metadata source file where the type is a column.

Load Template Mapping Rules and Rule Sets

Mapping rules determine how to map the metadata in the metadata source files to model components, like classes, class attributes, associations, and lineage associations. You can group mapping rules into a mapping rule set. You create mapping rules and mapping rule sets in a load template.

You can create the following types of mapping rules:

Class Rules

You can create a rule for a class of objects. Each class rule maps class attributes to object properties in the metadata source files. In each class rule, you can also create associations and lineage associations between the class and other classes.

Property Rules

You can create a rule for a class attribute. Create the property rule to map a class attribute to the object property.

Association Rules

You can create a rule for an association. In the rule, you specify the "from" and "to" objects in the association. Associations identify relationships between parent and child objects. For example, a database table is the parent object of a column in a database table.

Lineage Association Rules

You can create a rule for a lineage association. In the rule, you specify the "from" and "to" objects in the lineage association. Lineage associations identify data lineage between two classes of objects within the same custom resource. You can create multiple lineage associations for a class. To create lineage between two objects in different resources, use rule-based links.

You can apply a condition to each mapping rule to filter rows from the metadata source file that do not apply. You specify an expression in the condition. When you run the custom XConnect, Metadata Manager evaluates the expression for every row in the metadata source file.

You can group mapping rules into mapping rule sets in the load template. You apply mapping rules and mapping rule sets to a custom resource when you create the resource. You can apply the same mapping rule or mapping rule set to one or more metadata source files.

You can reuse the mapping rules and mapping rule sets for other custom metadata resources of the same resource type. Because mapping rules are specific to a model, you can only apply them to resources of the same resource type.

Load Template Expressions

You can add expressions to mapping rules. You can use operators and functions in the expressions.

You can add expressions to the following types of load template objects:

Element values

For example, you can use an expression to create a unique value for each object ID. You concatenate the following values to populate the id element:

```
<id>PARENT+'~'+ELEMENT</id>
```

Conditions

For example, you can use the non-equality operator to get all rows except for those of a particular type.

Load Template Operators

You can use operators in load template expressions.

You can use the following operators in load template expressions:

- Logical operators AND and OR.
- Comparison operators = and !=.
- String operator + (concatenate).

substr Function

Returns a portion of a string. substr counts all characters, including blanks, starting at the beginning of the string.

Note: The function name is case sensitive.

Syntax

```
substr( string, start [,length] )
```

The following table describes the substr arguments:

Argument	Required/ Optional	Description
string	Required	Must be a character string. You can enter an expression. Passes the strings you want to search. If you pass a numeric value, the function converts it to a character string.
start	Required	Must be an integer. The position in the string where you want to start counting. You can enter an expression. If the start position is a positive number, substr locates the start position by counting from the beginning of the string. If the start position is a negative number, substr locates the start position by counting from the end of the string. If the start position is 0, substr searches from the first character in the string.
length	Optional	Must be an integer greater than 0. The number of characters you want substr to return. You can enter an expression. If you omit the length argument, substr returns all of the characters from the start position to the end of the string. If you pass a negative integer or 0, the function returns an empty string. If you pass a decimal, the function rounds it to the nearest integer value.

Return Value

String.

Empty string if you pass a negative or 0 length value.

NULL if a value passed to the function is NULL.

Example

The following expressions return the phone number without the area code for each row in the PHONE column:

```
substr( PHONE, 5, 8 )
```

PHONE	RETURN VALUE
808-555-0269	555-0269
809-555-3915	555-3915
357-687-6708	687-6708
NULL	NULL

trim Function

Removes blanks from the beginning and end of a string.

Note: The function name is case sensitive.

Syntax

```
trim( string )
```

The following table describes the trim arguments:

Argument	Required/ Optional	Description
string	Required	Any string value. Passes the values you want to trim. You can enter an expression. Use operators to perform comparisons or concatenate strings before removing blanks from the end of a string.

Return Value

String. The string values with the starting and ending blanks removed.

NULL if a value passed to the function is NULL.

Creating and Uploading a Load Template

Generate a default load template through mmcmd. Upload the load template on the **Model** tab in Metadata Manager.

Before you create a load template, you must define the model associated with the custom metadata.

1. Open the command prompt.
2. Run the mmcmd generateDefaultLoadTemplate command.

   The command generates the default load template file and metadata source files.
3. Populate the metadata in the generated metadata source files.
4. Use a text editor to update the load template, if required.

   You might need to update the load template to create additional associations or lineage associations, or to change the column headers if you changed them in the metadata source file.
5. To upload the load template, open the **Model** tab in Metadata Manager, right-click the model, and select **Upload Load Template**.

   When you upload the load template, Metadata Manager validates the metadata against the model to prevent repository inconsistencies.

Updating and Deleting a Load Template

You can update or delete a load template on the **Model** tab in Metadata Manager. When you update a load template, Metadata Manager replaces the existing load template with the updated load template. You can get a copy of the existing load template and change it.

To get a copy of the latest load template for the resource, right-click the model and select **Get Load Template**.

To update the load template, right-click the model, and select **Update Load Template**.

To delete a load template, right-click the model and select **Delete Load Template**.

Note: You cannot delete a load template if it is used in one or more resources. You must delete the associated resources before you can delete the load template.

Load Template Sample

The sample load template shows how you can structure a load template. The sample load template contains mapping rules for classes, class properties, associations, and lineage associations.

You can use the following sample to help you create your own load template:

```
<loadTemplate name="AS400_1" targetModel="AS400"  templateSpecVersion="1.0">
    <class ruleName="AS400TableRule" name="AS400Table" condition="TYPE='TABLE'">
        <id>PARENT+'~'+ELEMENT</id>
        <properties>
            <property name="Name">ELEMENT</property>
            <property name="LongName">TRIM(LONG_NAME,0,256)</property>
        </properties>
        <associations>
```

```
            <parent>PARENT</parent>
        </associations>
        <lineageAssociations>
            <lineageAssociation direction="to">
                <linkExpression>LINK_EXPRESSION</linkExpression>
                <targetElement>TABLE</targetElement>
            </lineageAssociation>
        </lineageAssociations>
    </class>
    <property ruleName="AS400ElementPropertyRule" element="ELEMENT">
        <name>NAME</name>
        <value>VALUE</value>
    </property>
    <association ruleName="AS400ParentRule" name="ParentAssociation">
        <from>FROM_ELEMENT</from>
        <to>TO_ELEMENT</to>
    </association>
    <lineageAssociation ruleName="AS400ViewLinkRule">
        <linkExpression>LINK_EXPRESSION</linkExpression>
        <from>FROM_ELEMENT</from>
        <to>TO_ELEMENT</to>
    </lineageAssociation>
</loadTemplate>
```

CHAPTER 4

Custom Resource Creation

This chapter includes the following topics:

Custom Resource Creation Overview

Use custom resources to add or load custom metadata from metadata sources for which Metadata Manager does not package a model. When you create a custom resource, you select the metadata source files associated with the resource and configure mapping rules. You can also configure enumerated links and attach a schedule.

Before you create a custom resource that is based on a load template, you must create a custom model and upload the load template.

To create a custom resource, complete the following steps:

1. Upload the extraction command file, if you use one.
2. Add or upload the metadata source files.
3. Configure mapping rules.
4. Configure linking rules.
5. Configure enumerated links.
6. Optionally, attach a schedule.

After you create the custom resource, you can load the resource and configure linking rules.

Mapping Rules Configuration

Mapping rules determine how to map the metadata in the metadata source files to model components such as classes, class attributes, associations, and lineage associations. You create mapping rules in the load template for the custom resource. You configure mapping rules when you create or edit a custom resource.

Configure mapping rules on the **Mapping Rules** tab when you create or edit a custom resource. You can upload or add metadata source files for the resource. If you upload metadata source files, Metadata Manager automatically assigns the metadata source files to the mapping rules. If you add metadata source files, you must assign metadata source files to mapping rules manually. You can apply multiple mapping rules to each metadata source file.

Related Topics:

- "Load Templates" on page 35

Linking Rules Configuration

To run data lineage on a custom resource, you must link objects in the custom resource with other objects in the resource or with objects in another resource. You can use enumerated links or rule-based links to link objects. Rule-based links are expressions that Metadata Manager uses to link matching objects.

You define rule-based links in an XML file. When you edit a custom resource, you can upload XML files that contain linking rules for the resource. Specify linking rules files on the **Linking Rules** tab. You can specify multiple linking rules files for a resource. Metadata Manager creates the links when you load the resource.

Related Topics:

- "Rule-Based Links" on page 56

Enumerated Links Configuration

To run data lineage on a custom resource, you must link objects in the custom resource with other objects in the resource or with objects in another resource. To create links between pairs of objects, you can define linking rules or use enumerated links. Enumerated links are pairs of individually identified objects that you want to link.

Generally, linking rules are easier to use and more powerful than enumerated links. However, sometimes it is not possible to define a linking rule. If you cannot define a linking rule, you can use enumerated links to link objects. You can also combine rule-based links and enumerated links. Use both methods when not all objects comply with a linking rule. Define a linking rule for most objects and use enumerated links for the exceptions to the rule.

When you create or edit a custom resource, you can specify additional input files that contain enumerated links. Add or upload the enumerated links files on the **Enumerated Links** tab. You can specify multiple enumerated links files for a resource. Metadata Manager creates the links when you load the resource.

Related Topics:

- "Enumerated Links" on page 83

Schedule Assignment

You can attach a schedule to a custom resource when you create or load the resource. The schedule determines when to load the resource.

Attach a schedule to a resource on the **Schedule** tab. You can attach one schedule to a resource.

Note: If you attach a schedule to a resource, add metadata source files to the resource instead of uploading the files. If you upload metadata source files and attach a schedule, Metadata Manager reloads the same metadata source files every time it loads the resource.

For more information about scheduling resource loads, see the *Metadata Manager Administrator Guide*.

Creating a Custom Resource

Create a custom resource to extract metadata from sources for which Metadata Manager does not package a resource type.

Before you create a custom resource, you must create the custom model and upload the load template. If you use an extraction command file, you must also set the mm.allow.preload.command custom property for the Metadata Manager Service to true.

1. On the **Load** tab, click **Actions** > **New Resource**.

   The **Resource Selection** window appears.
2. In the **Other** folder, select the custom resource type.
3. Click **Next**.

   The **Properties** window appears.
4. Enter the resource name and optional description.
5. Click **Next**.

   The **Input Files** window appears.
6. If you use an extraction command file, click **Upload** and select the file.
7. Add or upload the metadata source files:
   - Add metadata source files when you store the files in a directory that the Metadata Manager web application can access and the files change.
   - Upload metadata source files when the files do not change. Metadata Manager uploads the files into the Metadata Manager repository.
8. Update the file properties for each metadata source file, if required.
9. Click **Next**.

   The **Mapping Rules** window appears.
10. Assign each metadata source file to one or more mapping rules.

    If you upload metadata source files, Metadata Manager displays the mapping rule associations at the bottom of the **Mapping Rules** window. If you add metadata source files, you must assign the files to mapping rules manually.

    To assign a metadata source file to mapping rules, select the file, select the mapping rules, and click **Assign**.

    To delete an association between a metadata source file and a mapping rule, click **Remove** at the bottom of the **Mapping Rules** window.
11. Click **Next**.

    The **Enumerated Links** window appears.
12. Optionally, add or upload the files that contain enumerated links information:
    - Add enumerated links files when you store the files in a directory that the Metadata Manager web application can access and the files change.
    - Upload enumerated links files when the files do not change. Metadata Manager uploads the files into the Metadata Manager repository.
13. Update the file properties for each enumerated links file, if required.
14. Click **Next**.

    The **Schedule** window appears.
15. Specify whether to attach a schedule.

16. Click **Finish**.

    Metadata Manager creates the custom resource.

After you create the resource, you can configure linking rules, load the resource, and create links between resources.

Metadata Source File Properties

For each metadata source file that you associate with a custom resource, you specify the file properties. The file properties include the file path and file name, whether the file has a header, and the file encoding.

The following list describes the metadata source file properties:

File

File path and file name for the metadata source file.

Has header

Specifies whether the file has a header row. Default is enabled.

Encoding

Code page for the file. Default is UTF-8.

Column separator

Character used to separate columns of data. The column separator must be a printable character and must be different from the text qualifier character and from the escape character. Default is the comma (,) character.

Text qualifier

Character that defines the boundaries of text strings. Metadata Manager ignores delimiters within pairs of the text qualifier character. The text qualifier character must be a printable character and must be different from the column separator character and from the escape character. Default is the double quotation marks (") character.

Escape character

Character used to escape a column separator character or a text qualifier character in an unquoted string if the character is the next character after the escape character. If you specify an escape character, Metadata Manager reads the column separator character or the text qualifier character as a regular character embedded in the string. The escape character must be a printable character and must be different from the column separator character and from the text qualifier character. Default is the backslash (\) character.

Rows to skip

Number of rows after which Metadata Manager starts importing data. Use this option to skip multiple header rows. If you enable the **Has header** option, Metadata Manager skips the first row plus the number of rows you specify in this option. Default is 0.

Always use latest source files

Uses the metadata source file in the location you provide each time you load the resource. Enable this option if you use an extraction command file.

If you enable this option, the path to the file must include an absolute path that is accessible from the Metadata Manager web application. If you disable this option, Metadata Manager copies the file to the Metadata Manager application directory when you finish configuring the resource. Each time you load the resource, Metadata Manager uses the copied file in the Metadata Manager application directory.

Default is enabled for metadata source files that you add and disabled for metadata source files that you upload.

Enumerated Links File Properties

For each enumerated links file that you associate with a custom resource, you specify file properties. File properties include the file path and file name, whether the file has a header, and whether to use the latest source files.

The following list describes the enumerated links file properties:

File

File path and file name for the enumerated links file.

Has header

Specifies whether the file has a header row. Default is disabled.

Always use latest source files

Uses the enumerated links file in the location you provide each time you load the resource. Enable this option if you use an extraction command file.

If you enable this option, the path to the file must include an absolute path that is accessible from the Metadata Manager web application. If you disable this option, Metadata Manager copies the file to the Metadata Manager application directory when you finish configuring the resource. Each time you load the resource, Metadata Manager uses the copied file in the Metadata Manager application directory.

Default is enabled for enumerated links files that you add and disabled for enumerated links files that you upload.

CHAPTER 5

Rule-Based Links

This chapter includes the following topics:

Rule-Based Links Overview

To run data lineage across metadata sources, you must create links between matching objects in different sources. Use rule-based links to define the rules that Metadata Manager uses to link objects. You define rule-based links in a linking rules XML file.

Create rule-based links to define the rules that Metadata Manager uses to link matching objects in a pair of resources. For example, you might create a linking rule that links a business glossary term to an Oracle column when the business term "Technical Name" field matches the Oracle column name.

Use rule-based links to create links between the following resource types:

- Custom resource to another custom resource
- Custom resource to a packaged resource
- Custom resource to a universal resource
- Custom resource to a business glossary resource
- Business glossary resource to a packaged resource
- Business glossary resource to a universal resource

You can also use rule-based links to create links between packaged resources, between universal resources, or between packaged and universal resources when connection assignments do not create all of the required links.

To use rule-based links, you create linking rules as expressions that Metadata Manager uses to create link relationships between matching objects across different resources.

When you define a linking rule, you specify the following information:

- Set of possible objects in a source resource
- Set of possible objects in a target resource
- Direction that indicates whether the link originates from the source object or from the target object
- Expression that defines which source and target objects match

You can create multiple linking rules for the pair of resources that you want to link. You group linking rules that apply to the same pair of resources into a linking rule set. A linking rule set is a group of rules that links objects between two resources.

You define a linking rule set in a linking rules XML file. You create different types of linking rules files based on how you want to apply a linking rule set. You can apply a linking rule set to a pair of models or a pair of resources.

Linking Rules Files

You define a set of linking rules in a linking rules file. A linking rules file contains a set of linking rules that Metadata Manager uses to link objects in a pair of resources.

The type of linking rules file that you create depends on whether you want to create linking rules for a pair of models or a pair of resources.

Define a linking rule set for a pair of models when you want to apply the same linking rules to different resources associated with the models. For example, you develop a custom model, "CustomETL." You need to link objects in CustomETL resources with objects in Oracle resources. You can create a set of linking rules that allow you to link objects in any CustomETL resource with objects in any Oracle resource.

To define a linking rule set for a pair of models, create the following types of linking rules files:

Rule set definition file

An XML file that defines a linking rule set for a pair of models. A rule set definition file takes a parameter file that identifies the pair of resources to which to apply the linking rules. The rule set definition file must conform to the structure of the rule set definition file XML schema definition (XSD).

After you create a rule set definition file, you upload it on the **Model** tab. Metadata Manager associates the rule set definition with the source and target model identified in the file.

Rule set parameter file

An XML file that specifies the pair of resources to which to apply a rule set definition. It also contains parameter values for resource-specific attributes such as connection names and table names. The rule set parameter file must conform to the structure of the rule set parameter file XSD.

After you create a rule set parameter file, you upload it to the source resource or the target resource on the **Load** tab. When you upload the file, Metadata Manager creates a rule set for the resources. To create the rule set, Metadata Manager substitutes the parameters defined in the rule set definition file with the parameter values defined in the rule set parameter file.

Define a linking rule set for a pair of resources when you want to apply the linking rules to specific resources. For example, you want to link objects in resource "BusinessGlossary1" with objects in resource "Oracle1."

To define a linking rule set for a specific pair of resources, create the following type of linking rules file:

Rule set file

An XML file that defines a linking rule set for pair of resources. The rule set file must conform to the structure of the rule set file XSD.

After you create a rule set file, you upload it to the source resource or the target resource on the **Load** tab.

You define linking rule sets in rule set definition files and in rule set files. Rule syntax is identical in both file types except that rules in rule set definition files can contain parameters. The parameters represent resource-specific attributes such as connection names, package names, and string literals such as table names.

Process to Use Rule-Based Links

Use rule-based links to define the rules that Metadata Manager uses to create links between resources.

The process to use rule-based links varies based on whether you create linking rules for a pair of models or a pair of resources.

Process to Use Rule-Based Links for Models

You can use rule-based links to create linking rules for a pair of models.

To create linking rules for a pair of models, complete the following steps:

1. Create a rule set definition file.

   Configure the rule set in the file. Also define parameters for connections, packages, and string literals that are specific to the source or target resource.

2. Upload the rule set definition file to the source model or the target model on the **Model** tab.

   When you upload the file, Metadata Manager creates the rule set definition in the Metadata Manager repository. Metadata Manager associates the rule set definition with the models that you specify in the rule set definition file.

3. For each pair of resources to which you want to apply the rule set definition, create a rule set parameter file.

   Specify the source resource and the target resource in the file. Also specify a value for each parameter that you define in the rule set definition file.

4. Upload the rule set parameter file to the source resource or the target resource on the **Load** tab.

   When you upload the file, Metadata Manager creates the rule set in the Metadata Manager repository with the same name as the rule set parameter name. Metadata Manager associates the rule set with the resources that you specify in the rule set parameter file.

5. Link the objects in the resources by completing one of the following tasks:

   - Load the resources. The load process uses the rules to create the links between matching objects.
   - Use the **Resource Link Administration** window in the **Load** tab to create the links. If you loaded the resources, direct Metadata Manager to use the rules to create the links between matching objects.

Process to Use Rule-Based Links for Resources

You can use rule-based links to create linking rules for a pair of resources.

To create linking rules for a pair of resources, complete the following steps:

1. Create a rule set file and configure the rule set in the file.
2. If the rule set file contains endpoint links for a PowerCenter resource, load the PowerCenter resource to ensure that Metadata Manager configures connection assignments for the resource.
3. Upload the rule set file to the source resource or the target resource on the **Load** tab.

   When you upload the file, Metadata Manager creates the rule set in the Metadata Manager repository. Metadata Manager associates the rule set with the resources that you specify in the rule set file.
4. Link the objects in the resources by completing one of the following tasks:
   - Load the resources. The load process uses the rules to create the links between matching objects.
   - Use the **Resource Link Administration** window in the **Load** tab to create the links. If you loaded the resources, direct Metadata Manager to use the rules to create the links between matching objects.

Endpoint and Non-Endpoint Links

When you define a linking rule, you specify whether you are linking an object in one resource to an endpoint or to a non-endpoint in another resource. An endpoint is an object that has a connection to another object in a different resource.

Business Intelligence, Data Integration, and Data Modeling resource types contain endpoints. For example, in a PowerCenter resource, a Source Qualifier port is an endpoint because it can have a cross-resource relationship to a table column in a database management resource. In a PowerCenter resource, a Filter port is not an endpoint because it cannot have a relationship to an object in another resource.

When you link another resource to a Business Intelligence, Data Integration, or Data Modeling resource, you usually link to endpoints. However, you can link to a non-endpoint object in these resources. The rule syntax required to link endpoints differs from the syntax required to link non-endpoints.

Custom, SAP R/3, database management, and business glossary resource types do not contain endpoints. When you use linking rules to link objects in a custom resource to an object in these resource types, use non-endpoint links.

Endpoints in PowerCenter Resources

A PowerCenter resource contains endpoint and non-endpoint objects. In the PowerCenter model, endpoint classes are classes that can have relationships to classes in another model.

The following table lists the endpoint classes in the PowerCenter model:

Parent Class	Endpoint Class
Source Qualifier Instance	Source Qualifier Port
Target Definition Instance	Target Definition Port

Parent Class	Endpoint Class
Lookup Procedure Instance	Lookup Transformation Port
Mapping	Stored Procedure Instance

All other PowerCenter classes are non-endpoint classes.

Endpoints in Business Intelligence, Data Modeling, and Informatica Platform Resources

Business Intelligence, Data Modeling, and Informatica Platform resources contain endpoints and non-endpoint classes. Endpoint classes are classes that can have relationships to classes in another model.

You can determine which classes in a model are endpoint classes. Metadata Manager includes text files that list the endpoint classes for Business Intelligence, Data Modeling, and Informatica Platform models. The endpoint class files are named `<model type>.endpoint.txt`.

The endpoint class files are located in the following directory:

```
<Installation directory>\services\MetadataManagerService\md-repo\xconnects
```

You can also determine whether an individual class is an endpoint. To determine whether an individual class is an endpoint, check the **Properties** panel on the **Model** tab. The **Is an endpoint** property indicates whether the class is an endpoint.

Upload a Rule Set Definition to a Model

After you configure a rule set definition file, upload the file to the source or target model to create the rule set definition in the Metadata Manager repository.

When you upload a rule set definition file, Metadata Manager associates the rule set definition with the source and target models configured in the file. If either the source or the target model does not exist, Metadata Manager still creates the rule set definition. When you create the source or target model, Metadata Manager associates the saved rule set definition with the model.

Uploading a Rule Set Definition

Upload a rule set definition file to the source model or the target model. Upload a rule set definition to create or update a rule set definition in the Metadata Manager repository. You can upload multiple rule set definition files for a model.

1. On the **Model** tab, select a model.
2. Click **Actions** > **Upload Rule Set Definition**.

   The **Upload Rule Set Definition** dialog box appears.
3. Click **Browse**, select the rule set definition XML file, and click **Open**.
4. Click **OK**.

   Metadata Manager uploads the file and creates or updates the rule set definition. It also validates the rule set definition. If the rule set definition is not valid, check the service log for more information.

The **Rule Set Definitions** tab in the **Properties** panel displays the rule set definition information for the model. To remove a rule set definition, select the rule set definition, and click **Delete**.

Upload a Rule Set to a Resource

After you configure a rule set file or a rule set parameter file, upload the file to the source resource or the target resource. Upload the file to create or update the rule set in the Metadata Manager repository.

When you upload a rule set file or a rule set parameter file, Metadata Manager associates the rule set with the source and target resources configured in the XML file. If either the source or the target resource does not exist, Metadata Manager still creates the rule set. When you create the source or target resource, Metadata Manager associates the saved rule set with the resource.

When you create or update a rule set, Metadata Manager does not create the links between the resources. To use the rules to link metadata objects between resources, you must load the resources or create the links in the **Resource Link Administration** window.

Uploading a Rule Set

Upload a rule set file or a rule set parameter file to the source resource or the target resource to create or update the rule set in the Metadata Manager repository. You can upload multiple rule set files and multiple rule set parameter files to a resource.

Before you upload a rule set file that contains endpoint links for a PowerCenter resource, load the resource. Before you upload a rule set parameter file for a resource, create and upload the rule set definition file to the source or target model.

1. On the **Load** tab, select a resource in the **Resources** panel.
2. In the **Properties** panel, click **Edit**.

   The **Edit Resource** window appears.
3. Click the **Linking Rules** tab.
4. Click **Upload**.

   The **Upload** dialog box appears.
5. Click **Browse**, select the rule set file or the rule set parameter file, and click **Open**.
6. Click **OK**.

   Metadata Manager uploads the file and creates or updates the rule set. It also validates the rule set. If the rule set is not valid, check the service log for more information.

The **Linking Rules** tab in the **Properties** panel displays the rule set information for the resource.

Removing a Rule Set

To remove a rule set from a resource, edit the resource and delete the rule set.

1. On the **Load** tab, select the resource in the **Resources** panel.
2. In the **Properties** panel, click **Edit**.

   The **Edit Resource** window appears.
3. Click the **Linking Rules** tab.

4. Select the rule set that you want to remove, and click **Delete**.
5. Click **OK** to close the **Edit Resource** window.

   Metadata Manager removes the rule set from the resource.

Link Objects Using Rules

After you define linking rules in a linking rules file and create the rule set in the Metadata Manager repository, create the links between the resources. After you create links, you can run data lineage analysis across the metadata sources.

To link the objects in the resources, complete one of the following tasks:

Load the resources.

The load process creates the links between matching objects.

Use the Resource Link Administration window.

Use the **Resource Link Administration** window in the **Load** tab to create the links. If you loaded the resources, direct Metadata Manager to create the links between matching objects.

Creating Links through the Resource Link Administration Window

When you create links through the **Resource Link Administration** window, Metadata Manager links objects in the resource to objects in other resources based on all of the linking rules.

1. On the **Load** tab, click **Actions** > **Resource Link Administration**.

   The **Resource Link Administration** window appears.
2. Select the resources that you want to link, and click **Create Links**.

   Metadata Manager adds the resource to the link queue, and then starts the link process.
3. To cancel a link process, select the resource, and click **Actions** > **Cancel** in the **Load** tab.

When the linking completes, Metadata Manager updates the Last Status Date and Last Status for the resource.

CHAPTER 6

Linking Rules File Configuration

This chapter includes the following topics:

Linking Rules File Configuration Overview

Define linking rule definitions or linking rule sets for a pair of models or a pair of resources in a linking rules file. A linking rules file is an XML file that contains a set of linking rules or that defines parameters associated with a rule set definition.

You can create the following types of linking rules files:

Rule Set Definition File

An XML file that defines a linking rule set for a pair of models. A rule set definition file takes a parameter file that identifies the pair of resources to which to apply the linking rules. A rule set definition file must conform to the structure of the rule set definition file XML schema definition (XSD).

Rule Set Parameter File

An XML file that specifies the pair of resources to which to apply a rule set definition. It also contains parameter values for connection names, package names, and string literals such as table names. A rule set parameter file does not contain linking rules. A rule set parameter file must conform to the structure of the rule set parameter file XML schema definition (XSD).

Rule Set File

An XML file that defines a linking rule set for pair of resources. A rule set file cannot contain parameters or parameter definitions. A rule set file must conform to the structure of the rule set file XML schema definition (XSD).

You define linking rule sets in rule set definition files and in rule set files. Rule syntax is identical in both file types except that the rules in rule set definition files can contain parameters. Create linking rules files with a text editor or XML editor.

Rule Set Definition File

A rule set definition file is an XML file that defines linking rules for a pair of models. You create a rule set definition file based on the rule set definition file XML schema definition (XSD).

To create a rule set definition file, create an XML file that includes the following XML elements:

```
<?xml version="1.0" encoding="UTF-8"?>

<ruleSetDefinition>

    <sourceModel/>
    <targetModel/>

    <param/>

    <rule>
        <sourceFilter>
            …
        </sourceFilter>
        <targetFilter>
            …
        </targetFilter>

        <link/>
    </rule>

</ruleSetDefinition>
```

The XML attributes and elements that these elements contain depend on whether you configure rules to link to endpoints or to link to non-endpoints.

Rule Set Definition File Elements

The XML elements in the rule set definition file define the rules that Metadata Manager uses to link objects in the source resource to matching objects in the target resource.

Use the following XML elements:

ruleSetDefinition

Contains a group of rules to link objects between two models. A ruleSetDefinition element contains a required name attribute. The ruleSetDefinition name must be unique in the Metadata Manager repository.

A ruleSetDefinition element must contain one sourceModel element, one targetModel element, and at least one rule element. A ruleSetDefinition element can also contain one or more param elements.

sourceModel

If you link endpoints, this element defines the name of the Business Intelligence, Data Integration, or Data Modeling model that contains endpoints. If you link non-endpoints, this element defines the name of one model that you want to link. Contains a required name attribute.

targetModel

If you link endpoints, this element defines the name of the model that does not contain endpoints. If you link non-endpoints, this element defines the name of the other model that you want to link. Contains a required name attribute.

param

Defines a parameter that represents a resource-specific attribute. A parameter can define a connection name, a package name, or a string literal. Contains a required name attribute and optional description, defaultValue, and type attributes.

The following table describes the values for the type attribute:

Value	Description
connection	The parameter defines a connection name for endpoint links.
package	The parameter defines a package name for endpoint links.
string	The parameter defines a string literal in an expression, for example, a table name. If you do not specify a value for the type attribute, the parameter type is string.

rule

Defines the linking rule. The XML attributes and elements that this element contains depends on whether you configure rules to link endpoint classes or to link non-endpoint classes.

Rule Set Parameter File

A rule set parameter file is an XML file that specifies the resources to which to apply a rule set definition. It also contains parameter values for resource-specific attributes. You create a rule set parameter file based on the rule set parameter file XML schema definition (XSD).

To create a rule set parameter file, create an XML file that includes the following required XML elements:

```
<?xml version="1.0" encoding="UTF-8"?>

<ruleSetParams>

    <sourceResource/>
    <targetResource/>

    <param/>

</ruleSetParams>
```

The XML attributes and elements that these elements contain depend on whether you configure rules to link to endpoints or to link to non-endpoints.

Rule Set Parameter File Elements

The XML elements in the rule set parameter file define the resource-specific attributes for a rule set definition. When you upload the rule set parameter file, Metadata Manager creates a rule set for the resource by substituting the parameter values for the parameters defined in the rule set definition.

Use the following required XML elements:

ruleSetParams

Contains the names of the source and target resource and the values for all parameters defined in a rule set definition.

The following table describes the attributes for the ruleSetParams element:

Attribute	Description
definition	Required. Identifies the rule set definition to which the rule set parameter file applies.
name	Required. A string that you specify so that you can update or delete the rule set. The name must be unique in the Metadata Manager repository.
description	Optional. Element description.

A ruleSetParams element must contain one sourceResource element, one targetResource element, and one param element for each parameter defined in the rule set definition file.

sourceResource

If you link endpoints, this element defines the name of the Business Intelligence, Data Integration, or Data Modeling resource that contains endpoints. If you link non-endpoints, this element defines the name of one resource that you want to link. Contains a required name attribute.

targetResource

If you link endpoints, this element defines the name of the resource that does not contain endpoints. If you link non-endpoints, this element defines the name of the other resource that you want to link. Contains a required name attribute.

param

Identifies the parameter and defines the parameter value. Contains a required name attribute and a required value attribute.

Rule Set File

A rule set file is an XML file that defines linking rules for a pair of resources. You create a rule set file based on the rule set file XML schema definition (XSD).

To create a rule set file, create an XML file that includes the following required XML elements:

```
<?xml version="1.0" encoding="UTF-8"?>

<ruleSet>

    <sourceResource/>
    <targetResource/>

    <rule>
        <sourceFilter>
            …
        </sourceFilter>
        <targetFilter>
            …
        </targetFilter>

        <link/>
    </rule>

</ruleSet>
```

The XML attributes and elements that these required elements contain depend on whether you configure rules to link to endpoints or to link to non-endpoints.

Rule Set File Elements

The XML elements in the rule set file define the rules that Metadata Manager uses to link objects in the source resource to matching objects in the target resource.

Use the following required XML elements:

ruleSet

Contains a group of rules to link objects between two resources. A ruleSet element contains a required name attribute. The ruleSet name must be unique in the Metadata Manager repository.

A ruleSet element must contain one sourceResource element, one targetResource element, and at least one rule element.

sourceResource

If you link endpoints, this element defines the name of the Business Intelligence, Data Integration, or Data Modeling resource that contains endpoints. If you link non-endpoints, this element defines the name of one resource that you want to link. Contains a required name attribute.

targetResource

If you link endpoints, this element defines the name of the resource that does not contain endpoints. If you link non-endpoints, this element defines the name of the other resource that you want to link. Contains a required name attribute.

rule

Defines the linking rule. The XML attributes and elements that this element contains depends on whether you configure rules to link endpoint classes or to link non-endpoint classes.

Rule Element Configuration for Endpoints

Use the rule element to define a linking rule that Metadata Manager uses to link endpoints in the source resource to matching objects in the target resource.

The following table describes the attributes for the rule element:

Attribute	Description
name	Required. Name of the rule. The rule name must be unique in the rule set.
direction	Optional. Indicates whether the link originates from the source object or from the target object. For endpoint links, the direction must be Auto. If you do not use the direction attribute, Metadata Manager uses Auto by default.

A rule element must contain the following elements:

sourceFilter

Filters the list of possible endpoints in the source resource that you want to link. A sourceFilter element must contain an endpoint element.

The following table describes the attributes for the endpoint element:

Attribute	Description
class	Optional. Name of the class that the endpoint belongs to. Use a pipe (|) to separate multiple class names. Either the class attribute or the type attribute is required.
type	Optional. Type of the class that the endpoint belongs to. Use a pipe (|) to separate multiple class types. Either the class attribute or the type attribute is required.
connection	Optional. Name of the connection that the endpoint uses to connect to an external source database. To view the list of all connections in the Business Intelligence, Data Integration, or Data Modeling resource, view the **Connection Assignment** tab in the resource properties. In a rule set definition file, you can use a parameter to represent the connection name. Enter the connection name in the following format: `connection="${<parameter_name>}"`
package	Optional. Name of the package or the database schema that the endpoint connects to in the external source database. To view the list of all schema names, view the **Connection Assignment** tab in the properties of the Business Intelligence, Data Integration, or Data Modeling resource. In a rule set definition file, you can use a parameter to represent the package name. Enter the package name in the following format: `package="${<parameter_name>}"`

targetFilter

Filters the list of possible objects in the target resource that you want to link. A targetFilter element must contain an XML element named element.

The following table describes the attributes for the XML element named element:

Attribute	Description
class	Optional. Name of the class that the object belongs to. Use a pipe (|) to separate multiple class names. Either the class attribute or the type attribute is required.
type	Optional. Type of the class that the object belongs to. Use a pipe (|) to separate multiple class types. Either the class attribute or the type attribute is required.
condition	Optional. Expression that filters the list of possible objects to link.

An element can contain another element. The objects selected through the class, type, and condition attributes for this element must be immediate children of the objects selected through the containing element.

To specify a feature in a packaged or universal resource, the structure or the parent class must also be selected in the targetFilter. For example, the following targetFilter element includes the parent table as a containing element of an Oracle column:

```
<targetFilter>
   <element class="Oracle Table">
      <element type="Column"/>
```

```
    </element>
</targetFilter>
```

link

Defines the expression that specifies which filtered source and target objects Metadata Manager links. Contains a required condition attribute.

Expressions to Link Endpoints

When you define a linking rule, you enter expressions in condition attributes. Expressions filter the list of possible objects to link and define which filtered objects Metadata Manager links.

You can include an expression in a condition attribute for the following elements in a rule set definition file or a rule set file configured to link endpoints:

- targetFilter elements use the condition attribute as a selection expression. A selection expression filters the list of possible objects to link in the target resource.
- link elements use the condition attribute as a link expression. A link expression defines which filtered objects are linked. Metadata Manager links the objects that meet the condition defined in the link expression.

You can use the following operators in selection expressions and link expressions:

- Logical operators AND and OR.
- Comparison operators =, !=, IS NULL, and IS NOT NULL.
- Parentheses to group multiple conditions.

In a rule set definition file, you can use parameters to represent string literals in selection expressions and link expressions. Enter each parameter in the format `${<parameter_name>}`. For example:

```
<link condition="source.structName=${ref.name}" />
```

Selection Expressions for Endpoints

You can include class attributes in the selection expression. Use string literal values to define the values of attributes.

You can use any class attribute in a selection expression except for the following attributes:

- Object Class
- Location
- Source Creation Date
- Source Update Date
- MM Creation Date
- MM Update Date

If the attribute name contains spaces, enclose it in the XML character entity for quotation marks, `&quot;`. For example, if the attribute name is "Business Name," enter the name as follows:

```
&quot;Business Name&quot;
```

Literal values for attributes can use the following special characters:

```
! @ # $ % ^ { } | ?
```

Example

The following selection expression filters the list of possible target objects to objects in the class MyCustomClass that have a Description attribute with a value of "MyDescription":

```
<targetFilter>
   <element class="MyCustomClass" condition="Description='MyDescription'">
   </element>
</targetFilter>
```

Link Expressions for Endpoints

A link expression for endpoints must refer to both the source and target resources. When you create link expressions for endpoints, you can use keywords and specific endpoint attributes.

To refer to an object attribute in a link expression, use the following format:

```
<keyword>.<attribute>
```

The following table describes the keywords that you can use in link expressions:

Keyword	Description
source	Represents the object selected in the sourceFilter.
target	Represents the object selected in the targetFilter.
parent	Represents the parent of the object selected in the sourceFilter or targetFilter.

You can use the structName, featureName, and packageName attributes for endpoints in a link expression. You cannot use any other class attribute.

The following table describes the endpoint attributes that you can use in link expressions that link endpoints:

Endpoint Attribute	Description
structName	Structure name. A structure is a metadata object that contains fields. For example, a structure can be a PowerCenter source definition instance or an Oracle table.
featureName	Feature name. A feature is a field in a metadata object. For example, a field can be a PowerCenter source definition port or an Oracle table column.
packageName	Package name. A package is the schema for a database resource that the connection is assigned to.

To refer to object attributes in the target resource, use the same class attributes that are valid for selection expressions.

If the attribute name contains spaces, enclose it in the XML character entity for quotation marks, `&quot;`. For example, if the attribute name is "Business Name," enter the name as follows:

```
&quot;Business Name&quot;
```

When you include multiple conditions in a link expression, Metadata Manager evaluates the conditions from left to right. For better performance during linking, write a condition that checks for parent attributes before a condition that checks for child attributes.

Example

The following link expression links a source endpoint to a target object when the parent names and the object names match:

```
<link condition="source.structName = target.parent.Name AND source.featureName =
target.Name"/>
```

Sample Rule Set Definition Files to Link Endpoints

The following code shows a sample rule set definition file that defines linking rules between a PowerCenter model and a Microsoft SQL Server model:

```
<?xml version="1.0" encoding="UTF-8"?>
<ruleSetDefinition name="test_ep_to_nep_def" description="Link on ref table name">

    <sourceModel name="PowerCenter"/>
    <targetModel name="SQLServer"/>

    <param name="connection" description="PowerCenter connection name"
type="connection" />
    <param name="ref.name" description="Source qualifier reference table name" />
    <param name="table.name" description="Table name" />

    <rule name="source target">

        <sourceFilter>
            <endPoint connection="${connection}" class="Source Qualifier Instance"/>
        </sourceFilter>

        <targetFilter>
            <element class="Sqlserver Table" condition="Name=${table.name}">
                <element class="Sqlserver Column"/>
            </element>
        </targetFilter>

        <link condition="source.structName=${ref.name} AND target.parent.Name =
source.structName AND target.Name = source.featureName" />

    </rule>

</ruleSetDefinition>
```

The following code shows a sample of a rule set parameter file that defines the source and target resources and the parameter values for the rule set definition file:

```
<?xml version="1.0" encoding="UTF-8"?>
<ruleSetParams definition="test_ep_to_nep_def" name="INFA27003" description="Link rbl_pc
to RBLSchema">

    <sourceResource name="rbl_pc" />
    <targetResource name="RBLSchema" />

    <param name="connection" value="rbltest" />
    <param name="ref.name" value="CUSTOMER" />
    <param name="table.name" value="CUSTOMER" />

</ruleSetParams>
```

Sample Rule Set Files to Link Endpoints

The following code shows a sample rule set file that defines linking rules between custom resource objects and PowerCenter resource endpoints:

```
<?xml version="1.0" encoding="UTF-8"?>
<ruleSet name="Link custom objects to PowerCenter endpoints">
```

```
    <sourceResource name="MyPowerCenterResource"/>
    <targetResource name="MyCustomResource"/>

    <rule name="Link custom columns to PowerCenter Source Qualifier or Lookup ports">

        <sourceFilter>
            <endPoint connection="MyConnection" class="Source Qualifier Port|Lookup
Transformation Port"/>
        </sourceFilter>

        <targetFilter>
            <element class="Library" >
                <element class="Table">
                    <element class="TableColumn"/>
                </element>
            </element>
        </targetFilter>

        <!--Link the specified endpoints and objects when the parent names and the
object names match. -->
        <link condition="source.structName = target.parent.Name AND source.featureName =
target.Name"/>

    </rule>

</ruleSet>
```

The following code shows a sample rule set file that defines linking rules between custom resource objects and Informatica Platform resource endpoints:

```
<?xml version="1.0" encoding="UTF-8"?>
<ruleSet name="Link custom objects to Informatica Platform endpoints">

    <sourceResource name="MyInfaPlatformResource"/>
    <targetResource name="MyCustomResource"/>

    <rule name="Link custom columns to Informatica Platform relational or flat file data
object columns.">

        <sourceFilter>
            <endPoint connection="MyConnection" class="Column"/>
        </sourceFilter>

        <targetFilter>
            <element class="Library" >
                <element class="Table">
                    <element class="TableColumn"/>
                </element>
            </element>
        </targetFilter>

        <!--Link the specified endpoints and objects when the parent names and the
object names match. -->
        <link condition="source.structName = target.parent.Name AND source.featureName =
target.Name"/>

    </rule>

</ruleSet>
```

Rule Element Configuration for Non-Endpoints

Use the rule element to define a linking rule that Metadata Manager uses to link non-endpoint objects in the source resource to matching non-endpoint objects in the target resource.

The following table describes the attributes for the rule element:

Attribute	Description
name	Required. Name of the rule. The rule name must be unique in the rule set.
direction	Optional if linking to a business glossary resource. Required if linking between custom resources, between a custom and packaged resource, or between a custom and universal resource. Indicates whether the link originates from the source object or from the target object. Enter SourceToTarget, TargetToSource, or Auto. To link between custom resources, between a custom and packaged resource, or between a custom and universal resource, the direction must be SourceToTarget or TargetToSource. To link to terms in a business glossary, the direction must be Auto. If you do not use the direction attribute, Metadata Manager uses Auto by default.

A rule element must contain the following elements:

sourceFilter

Filters the list of possible objects in the source resource that you want to link. A sourceFilter element must contain an XML element named element.

The following table describes the attributes for the XML element named element:

Attribute	Description
class	Optional. Name of the class that the object belongs to. Use a pipe (|) to separate multiple class names. Either the class attribute or the type attribute is required.
type	Optional. Type of the class that the object belongs to. Use a pipe (|) to separate multiple class types. Either the class attribute or the type attribute is required.
condition	Optional. Expression that filters the list of possible objects to link.

An element can contain another element. The objects selected through the class, type, and condition attributes for this element must be immediate children of the objects selected through the containing element.

To specify a feature in a packaged or universal resource, the structure or the parent class must also be selected in the sourceFilter. For example, the following sourceFilter element includes the parent table as a containing element of an Oracle column:

```
<sourceFilter>
   <element class="Oracle Table">
      <element type="Column"/>
   </element>
</sourceFilter>
```

targetFilter

Filters the list of possible objects in the target resource that you want to link. A targetFilter element must contain an XML element named element.

Use the same syntax to configure elements in sourceFilters and targetFilters.

link

Defines the expression that specifies which filtered source and target objects Metadata Manager links. Contains a required condition attribute.

Expressions to Link Non-Endpoints

When you define a linking rule, you enter expressions in condition attributes. Expressions filter the list of possible objects to link and define which filtered objects Metadata Manager links.

You can include an expression in a condition attribute for the following elements in a rule set definition file or a rule set file configured for non-endpoint links:

- sourceFilter and targetFilter elements use the condition attribute as a selection expression. A selection expression filters the list of possible objects to link in the source or target resource.
- link elements use the condition attribute as a link expression. A link expression defines which filtered objects are linked. Metadata Manager links the objects that meet the condition defined in the link expression.

You can use the following operators in selection expressions and link expressions:

- Logical operators AND and OR.
- Comparison operators =, !=, IS NULL, and IS NOT NULL.
- Parentheses to group multiple conditions.

In a rule set definition file, you can use parameters to represent string literals in selection expressions and link expressions. Enter each parameter in the format `${<parameter_name>}`. For example:

```
<link condition="source.parent.Description=${Desc_string1}" />
```

Selection Expressions for Non-Endpoints

You can include class attributes in the selection expression. Use string literal values to define the values of attributes.

You can use any class attribute in a selection expression except for the following attributes:

- Object Class
- Location
- Source Creation Date
- Source Update Date
- MM Creation Date
- MM Update Date

If the attribute name contains spaces, enclose it in the XML character entity for quotation marks, `&quot;`. For example, if the attribute name is "Business Name," enter the name as follows:

```
&quot;Business Name&quot;
```

Literal values for attributes can use the following special characters:

```
! @ # $ % ^ { } | ?
```

Example

The following selection expression filters the list of possible target objects to objects in the class MyCustomClass that have a Description attribute with a value of "MyDescription":

```
<targetFilter>
   <element class="MyCustomClass" condition="Description='MyDescription'">
   </element>
</targetFilter>
```

Link Expressions for Non-Endpoints

A link expression for non-endpoints must refer to both the source and target resources. When you create link expressions for non-endpoints, you can use keywords and attributes that belong to the non-endpoint classes.

To refer to an object attribute in a link expression, use the following format:

```
<keyword>.<attribute>
```

The following table describes the keywords that you can use in link expressions:

Keyword	Description
source	Represents the object selected in the sourceFilter.
target	Represents the object selected in the targetFilter.
parent	Represents the parent of the object selected in the sourceFilter or targetFilter.

You can use the same class attributes that are valid for selection expressions.

If the attribute name contains spaces, enclose it in the XML character entity for quotation marks, `&quot;`. For example, if the attribute name is "Business Name," enter the name as follows:

```
&quot;Business Name&quot;
```

When you include multiple conditions in a link expression, Metadata Manager evaluates the conditions from left to right. For better performance during linking, write a condition that checks for parent attributes before a condition that checks for child attributes.

Example

The following link expression links non-endpoint objects when the parent names and object names match:

```
<link condition="source.parent.Name=target.parent.Name AND source.Name=target.Name"/>
```

Example

The following link expression links non-endpoint objects when the Label attribute for the source object or target object has a value of "MyLabel":

```
<link condition="source.Label='MyLabel' OR target.Label='MyLabel'"/>
```

Sample Rule Set Definition Files to Link Non-Endpoints

The following code shows a sample rule set definition file that defines two linking rules between a pair of custom models:

```
<?xml version="1.0" encoding="UTF-8"?>
<ruleSetDefinition name="MAL_NEP_def_custom_to_custom_string_link">

    <sourceModel name="Custom_Dataset_MAL"/>
    <targetModel name="Custom_Dataset_Rel_MAL"/>
```

```
    <param name="Desc_string1" description="The description string value"
type="string" />
    <param name="Desc_string2" description="The description string value"
type="string" />

    <rule name="SC_to_SC_Rel_RBL_NEP_12_down" direction="TargetToSource">

        <sourceFilter>
            <element class="Class_lin">
                <element class="SubClass_lin"/>
            </element>
        </sourceFilter>

        <targetFilter>
            <element class="Class_lin_Rel">
                <element class="SubClass_lin_Rel"/>
            </element>
        </targetFilter>

        <link condition="source.Name=target.Description AND source.parent.Description=$
{Desc_string1} AND source.parent.Name=target.parent.Description AND target.parent.Label=$
{Desc_string1}"/>

    </rule>

    <rule name="SC_to_SC_Rel_RBL_NEP_12_up" direction="SourceToTarget">

        <sourceFilter>
            <element class="Class_lin">
                <element class="SubClass_lin"/>
            </element>
        </sourceFilter>

        <targetFilter>
            <element class="Class_lin_Rel">
                <element class="SubClass_lin_Rel"/>
            </element>
        </targetFilter>

        <link condition="source.Name=target.Description AND source.parent.Description=$
{Desc_string2} AND source.parent.Name=target.parent.Description AND target.parent.Label=$
{Desc_string2}"/>

    </rule>

</ruleSetDefinition>
```

The following code shows a sample of a rule set parameter file that defines the source and target resources and the parameter values for the rule set definition file:

```
<?xml version="1.0" encoding="UTF-8"?>
<ruleSetParams definition="MAL_NEP_def_custom_to_custom_string_link"
name="param_custom_string_link_res1" description="testing string parameter">

    <sourceResource name="Custom_Test_MBL" />
    <targetResource name="Custom_Test_MBL_Rel" />

    <param name="Desc_string1" value="grp3" />
    <param name="Desc_string2" value="grp4" />

</ruleSetParams>
```

Sample Rule Set Files to Link Non-Endpoints

The following code shows a sample rule set file that defines two linking rules between custom resource objects and database management resource objects:

```
<?xml version="1.0" encoding="UTF-8"?>
<ruleSet name="Link custom objects to Oracle objects">

    <sourceResource name="MyCustomResource"/>
    <targetResource name="MyOracleResource"/>

    <rule name="Link custom table or view column to Oracle column"
direction="TargetToSource">

        <sourceFilter>
            <element class="Library" >
                <element class="Table|Synonym|View">
                    <element class="TableColumn|ViewColumn"/>
                </element>
            </element>
        </sourceFilter>

        <targetFilter>
            <element class="Oracle Schema" >
                <element type="Table|View|Synonym">
                    <element type="Column"/>
                </element>
            </element>
        </targetFilter>

        <!--Link the specified objects when two levels of the parent names and the
object names match. -->
        <link condition="source.parent.parent.Name=target.parent.parent.Name AND
source.parent.Name=target.parent.Name AND source.Name=target.Name"/>

    </rule>

    <rule name="Link custom procedure to Oracle procedure" direction="TargetToSource">

        <sourceFilter>
            <element class="Library" >
                <element class="Procedure">
                </element>
            </element>
        </sourceFilter>

        <targetFilter>
            <element class="Oracle Schema" >
                <element type="Procedure">
                </element>
            </element>
        </targetFilter>

        <!--Link the specified objects when the parent names and the object names match.
-->
        <link condition="source.parent.Name=target.parent.Name AND
source.Name=target.Name"/>

    </rule>

</ruleSet>
```

The following code shows a sample rule set file that defines a linking rule between custom resource objects and business glossary terms:

```
<?xml version="1.0" encoding="UTF-8"?>
<ruleSet name="Link custom objects to business terms">

    <sourceResource name="MyCustomResource"/>
    <targetResource name="MyBusinessGlossary"/>
```

```
    <rule name="Link custom objects to business terms">

        <sourceFilter>
            <element class="Library" >
                <element class="Table">
                    <element class="TableColumn"/>
                </element>
            </element>
        </sourceFilter>

        <targetFilter>
            <element class="Business Term" />
        </targetFilter>

        <!-- Link columns and business terms when the column and business term names
match. -->
        <link condition="source.Name = target.Name" />

    </rule>

</ruleSet>
```

Linking Rules Files Schema Definitions

A linking rules file must conform to the structure of the XML schema definition (XSD). If the linking rules file does not conform to the schema definition, Metadata Manager cannot create or update the rule set definition in the Metadata Manager repository.

Each type of linking rules file has its own schema definition.

Rule Set Definition File Schema Definition

A rule set definition file must conform to the structure of the rule set definition file XSD.

The following example shows the rule set definition file XML schema definition:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

    <xs:element name="ruleSetDefinition">
        <xs:annotation>
            <xs:documentation>Container of rules. This container should have a globally
unique name</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:sequence>
                <!--Identifiers for selecting the source and target resource. Later it
can be used to just select types and provide the actual resources at application time -->
                <xs:element name="sourceModel">
                    <xs:complexType>
                        <xs:attribute name="name" type="xs:string"></xs:attribute>
                    </xs:complexType>
                </xs:element>
                <xs:element name="targetModel">
                    <xs:complexType>
                        <xs:attribute name="name" type="xs:string"></xs:attribute>
                    </xs:complexType>
                </xs:element>
                <!--A rule set must have at least one rule -->
                <xs:element name="param" type="paramDefType" maxOccurs="unbounded"
minOccurs="0"></xs:element>
                <xs:element maxOccurs="unbounded" name="rule" type="rule" />
            </xs:sequence>
```

```
            <xs:attribute name="name" type="xs:string" use="required" />
            <xs:attribute fixed="1.0" name="version" type="xs:string" />
            <!--Mandatory attributes for rule set-->
            <xs:attribute name="description" type="xs:string" use="optional"></
xs:attribute>
        </xs:complexType>
    </xs:element>

    <xs:complexType name="rule">
        <xs:annotation>
            <xs:documentation>A rule in a ruleset. A rule is uniquely identified by its
name within a rule set</xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:element name="sourceFilter" type="sourceElementSelectorType"/>
            <xs:element name="targetFilter" type="targetElementSelectorType"/>
            <xs:element name="link" type="linkType"/>
        </xs:sequence>
        <xs:attribute name="name" type="xs:string" use="required"/>
        <xs:attribute name="direction" type="directionType"/>
        <!--A rule must have source, target, link in the sequence-->
    </xs:complexType>

    <xs:simpleType name="directionType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="SourceToTarget"/>
            <xs:enumeration value="TargetToSource"/>
            <xs:enumeration value="Auto"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:complexType name="sourceElementSelectorType">
        <xs:choice>
            <xs:element name="element" type="elementFilterType"/>
            <xs:element name="endPoint" type="endPointType"/>
        </xs:choice>
    </xs:complexType>

    <xs:complexType name="targetElementSelectorType">
        <xs:choice>
            <xs:element name="element" type="elementFilterType"/>
        </xs:choice>
    </xs:complexType>

    <!-- Right now we are not creating simple types that can define a pattern for
specifying identifiers (like class name, feature name, type etc). If we have leisure
time later :) we can do that -->

    <xs:complexType name="elementFilterType">
        <xs:choice>
            <xs:element maxOccurs="unbounded" minOccurs="0" name="element"
type="elementFilterType"/>
        </xs:choice>
        <xs:attribute name="class" type="xs:string"/>
        <xs:attribute name="type" type="xs:string"/>
        <xs:attribute name="condition" type="xs:string"/>
    </xs:complexType>

    <xs:complexType name="endPointType">
        <xs:attribute name="class" type="xs:string"/>
        <xs:attribute name="type" type="xs:string"/>
        <xs:attribute name="connection" type="xs:string"/>
        <xs:attribute name="package" type="xs:string"/>
    </xs:complexType>

    <xs:complexType name="linkType">
        <xs:attribute name="condition" type="xs:string" use="required"/>
    </xs:complexType>

    <xs:complexType name="paramDefType">
        <xs:attribute name="name" type="xs:string"></xs:attribute>
```

```
        <xs:attribute name="description" type="xs:string"></xs:attribute>
        <xs:attribute name="defaultValue" type="xs:string"></xs:attribute>
        <xs:attribute name="type" default="string">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="connection"></xs:enumeration>
                    <xs:enumeration value="package"></xs:enumeration>
                    <xs:enumeration value="string"></xs:enumeration>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
    </xs:complexType>

</xs:schema>
```

Rule Set Parameter File Schema Definition

A rule set parameter file must conform to the structure of the rule set parameter file XSD.

The following example shows the rule set parameter file XML schema definition:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

    <xs:element name="ruleSetParams">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="sourceResource" type="resourceType" maxOccurs="1"
minOccurs="1"></xs:element>
                <xs:element name="targetResource" type="resourceType" maxOccurs="1"
minOccurs="1"></xs:element>
                <xs:element name="param" maxOccurs="unbounded" minOccurs="0">
                    <xs:complexType>
                        <xs:attribute name="name" type="xs:string" use="required"></
xs:attribute>
                        <xs:attribute name="value" type="xs:string" use="required"></
xs:attribute>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
            <xs:attribute name="definition" type="xs:string" use="required"></
xs:attribute>
            <xs:attribute name="name" type="xs:string" use="required"></xs:attribute>
            <xs:attribute name="description" type="xs:string" use="optional"></
xs:attribute>
        </xs:complexType>
    </xs:element>

    <xs:complexType name="resourceType">
        <xs:attribute name="name" type="xs:string" use="required"></xs:attribute>
    </xs:complexType>

</xs:schema>
```

Rule Set File Schema Definition

A rule set file must conform to the structure of the rule set file XSD.

The following example shows the rule set file XML schema definition:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

    <xs:element name="ruleSet">
        <xs:annotation>
            <xs:documentation>Container of rules. This container should have a globally
unique name</xs:documentation>
        </xs:annotation>
```

```
        <xs:complexType>
            <xs:sequence>
                <!--Identifiers for selecting the source and target resource. Later it
can be used to just select types and provide the actual resources at application time -->
                <xs:element name="sourceResource" type="resourceType" />
                <xs:element name="targetResource" type="resourceType" />
                <!--A rule set must have at least one rule -->
                <xs:element maxOccurs="unbounded" name="rule" type="rule" />
            </xs:sequence>
            <xs:attribute name="name" type="xs:string" use="required" />
            <xs:attribute fixed="1.0" name="version" type="xs:string" />
            <!--Mandatory attributes for rule set-->
            <xs:attribute name="description" type="xs:string" use="optional"></
xs:attribute>
        </xs:complexType>
    </xs:element>

    <xs:complexType name="resourceType">
        <xs:attribute name="name" use="required"/>
        <!--Mandatory attribute in this release. Later we will allow type also which
will make this attribute non-mandatory. Any one of them would be sufficient-->
    </xs:complexType>

    <xs:complexType name="rule">
        <xs:annotation>
            <xs:documentation>A rule in a ruleset. A rule is uniquely identified by its
name within a rule set</xs:documentation>
        </xs:annotation>
            <xs:sequence>
                <xs:element name="sourceFilter" type="sourceElementSelectorType"/>
                <xs:element name="targetFilter" type="targetElementSelectorType"/>
                <xs:element name="link" type="linkType"/>
            </xs:sequence>
            <xs:attribute name="name" type="xs:string" use="required"/>
            <xs:attribute name="direction" type="directionType"/>
            <!--A rule must have source, target, link in the sequence-->
    </xs:complexType>

    <xs:simpleType name="directionType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="SourceToTarget"/>
            <xs:enumeration value="TargetToSource"/>
            <xs:enumeration value="Auto"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:complexType name="sourceElementSelectorType">
        <xs:choice>
            <xs:element name="element" type="elementFilterType"/>
            <xs:element name="endPoint" type="endPointType"/>
        </xs:choice>
    </xs:complexType>

    <xs:complexType name="targetElementSelectorType">
        <xs:choice>
            <xs:element name="element" type="elementFilterType"/>
        </xs:choice>
    </xs:complexType>

    <!-- Right now we are not creating simple types that can define a pattern for
specifying identifiers (like class name, feature name, type etc). If we have leisure
time later :) we can do that -->

    <xs:complexType name="elementFilterType">
        <xs:choice>
            <xs:element maxOccurs="unbounded" minOccurs="0" name="element"
type="elementFilterType"/>
        </xs:choice>
        <xs:attribute name="class" type="xs:string"/>
        <xs:attribute name="type" type="xs:string"/>
        <xs:attribute name="condition" type="xs:string"/>
```

```
    </xs:complexType>

    <xs:complexType name="endPointType">
        <xs:attribute name="class" type="xs:string"/>
        <xs:attribute name="type" type="xs:string"/>
        <xs:attribute name="connection" type="xs:string"/>
        <xs:attribute name="package" type="xs:string"/>
    </xs:complexType>

    <xs:complexType name="linkType">
        <xs:attribute name="condition" type="xs:string" use="required"/>
    </xs:complexType>

</xs:schema>
```

CHAPTER 7

Enumerated Links

This chapter includes the following topics:

Enumerated Links Overview

When you create a custom resource, you can specify an additional input file that contains pairs of objects that you want to link. Enumerated links are the pairs of objects that you want to link. Use enumerated links when you cannot create links through a linking rule.

The enumerated links file contains one row for each pair of objects that you want to link. The file links objects in the custom resource with objects in a custom, packaged, universal, or business glossary resource so that you can run data lineage analysis across the metadata sources. The enumerated links file has a CSV file format.

Use enumerated links to create links between the following resource types:

- Custom resource to another custom resource
- Custom resource to a packaged resource
- Custom resource to a universal resource
- Custom resource to a business glossary resource
- Business glossary resource to a packaged resource
- Business glossary resource to a universal resource

You can also use enumerated links to create links between objects in the same custom resource.

Process to Use Enumerated Links

Use the enumerated links file to link objects in a custom resource with objects in another custom resource, in a packaged resource, or in a universal resource.

To use enumerated links, complete the following steps:

1. Create a CSV file and configure the enumerated links in the file.
2. Create a custom resource based on a custom model.
3. Add or upload the enumerated links file.

   Add a path to the enumerated links file to use the latest version of the file. Upload the enumerated links file to the Metadata Manager repository if the file will not change.
4. Use the **Enumerated Links** tab on the **Properties** panel of the **Load** tab to view or update information about the enumerated links file.

   You can also add, upload, or delete enumerated links files.
5. Load the resource.

   The load process uses the links in the enumerated links file to create links between matching objects.

Note: You can also use mmcmd to create and load a custom resource that uses enumerated links. If you use mmcmd commands, you must modify the resource configuration file and add enumerated links elements.

Enumerated Links File

An enumerated links file is a CSV file that contains pairs of objects you want to link.

The following table describes the columns for each row in the enumerated links file:

Column	Description
sourcePath	Location of the source element identified by its name and path.
sourceClass	Class of the source element. Optional.
destinationPath	Location of the target element identified by its name and path.
destinationClass	Class of the target element. Optional.
expression	The link expression that appears in the data lineage diagram when you hover over the target element. Optional.

If the enumerated links file has a column header, the optional columns do not need to be included in the file. If the file does not have a column header, all columns, including the optional columns, must be included.

Each row in the enumerated links file represents one link. The actual links in the custom resource might exceed the number of rows in the file. Link replication can occur when Metadata Manager links an object in one resource to an endpoint or to a non-endpoint object in another resource.

If an object has multiple locations, Metadata Manager creates links to this object for all paths.

If different objects share the same location, Metadata Manager uses the class name to distinguish the location. If Metadata Manager cannot distinguish objects by class name, it creates links to all objects in that location.

Sample Enumerated Links File

The following code shows a sample of an enumerated links file with headers:

```
sourcePath,sourceClass,destinationPath,destinationClass,expression
MM/AS400/GDWIDAT/Tables/GCOLMFACR/COLMR_COLL_SUBTYP_CD,"",MM/PC/Stress_S72277/
Transformations/LKP_having_multiple_inst,"",""
MM/AS400/GDWIDAT/Tables/GCOLMFACR/COLMR_COLL_TYP_C,"Column",MM/P/Stress_S72277/
Transformations/LKP_having_multiple_inst,"Transformation","Select * from GCOLMFACR"
MM/AS400/GDWIDAT/Tables/GCOLMFACR/COLMR_COLM_ID,"",MM/P/Stress_S72277/Transformations/
LKP_having_multiple_inst,"",""
MM/AS400/GDWIDAT/Tables/GCOLMFACR/COLMR_CTRY_CCY_CD,"",MM/P/Stress_S72277/
Transformations/LKP_having_multiple_inst,"",""
MM/AS400/GDWIDAT/Tables/GCOLMFACR/COLMR_CUS_ID,"",MM/PC/Stress_S72277/Transformations/
LKP_having_multiple_inst,"",""
MM/AS400/GDWIDAT/Tables/GCOLMFACR/COLMR_CUS_SRC_KEY,"",MM/PC/Stress_S72277/
Transformations/LKP_having_multiple_inst,"",""
MM/AS400/GDWIDAT/Tables/GCOLMFACR/COLMR_FAC_CD,"",MM/PC/Stress_S72277/Transformations/
LKP_having_multiple_inst,"",""
```

Note: The following columns are required:

- `sourcePath`
- `destinationPath`

The following columns are optional:

- `sourceClass`
- `destinationClass`
- `expression`

Enumerated Links File Properties

The enumerated links file includes properties such as the file path and file name, whether the file has a header, and whether to use the latest source files.

The following list describes the enumerated links file properties:

File

File path and file name for the enumerated links file.

Has header

Specifies whether the file has a header row. Default is disabled.

Always use latest source files

Uses the enumerated links file in the location you provide each time you load the resource. Enable this option if you use an extraction command file.

If you enable this option, the path to the file must include an absolute path that is accessible from the Metadata Manager web application. If you disable this option, Metadata Manager copies the file to the Metadata Manager application directory when you finish configuring the resource. Each time you load the resource, Metadata Manager uses the copied file in the Metadata Manager application directory.

Default is enabled for enumerated links files that you add and disabled for enumerated links files that you upload.

Creating Enumerated Links for a Custom Resource

Create enumerated links for a custom resource to link objects in the resource to objects in other resources based on the links in the enumerated links file. You can specify enumerated links files when you create or edit the custom resource.

1. On the **Load** tab, edit a custom resource.
2. Click the **Enumerated Links** tab.
3. Add or upload the files that contain enumerated links information:
   - Add enumerated links files when you store the files in a directory that the Metadata Manager web application can access and the files change.
   - Upload enumerated links files when the files do not change. Metadata Manager uploads the files into the Metadata Manager repository.
4. Update the file properties for each enumerated links file, if required.
5. Click **OK**.

Metadata Manager creates the links when you load the resource.

CHAPTER 8

Custom Metadata

This chapter includes the following topics:

Custom Metadata Overview

You can create custom metadata in the Metadata Manager warehouse. You can also edit the values for custom properties.

You can edit metadata object properties that you created for custom classes and properties that you created for classes in packaged and universal models. For example, you created a custom attribute in a report class for Cognos ReportNet. You can edit the value for the report property.

You can create and edit metadata in the Metadata Manager warehouse using the following methods:

Create custom metadata objects.

Use the metadata catalog to add metadata objects defined by classes in the model.

After you create metadata objects, you can edit the properties and relationships for the metadata objects.

Edit custom properties.

Edit the values of custom properties for metadata objects.

Creating Custom Metadata Objects

Create a model with classes, properties, and relationships on the **Model** tab. You can create classes and subclasses. After you create the model, you can add metadata objects based on the classes you created.

Create metadata objects through the metadata catalog on the **Browse** tab in Metadata Manager. You can edit or delete resources and metadata objects.

Create Custom Metadata Objects

You can create custom metadata objects and add them to a custom resource. Add custom metadata objects based on the classes for the model.

When you create the custom metadata object, select the class for which you want to create the metadata object. Add metadata objects based on the model hierarchy you created on the **Model** tab. For example, when you add a metadata object to the AccessDB resource, you add metadata objects of type AccessSchema, because the AccessSchema class is the root class of the model.

Create custom metadata objects in the **Create Custom Objects** window. You can create multiple metadata objects through this window. If the Metadata Manager repository is in a database that uses case insensitive collation, you cannot create multiple metadata objects with the same name but different cases. For example, you cannot create objects Customer and CUSTOMER.

By default, Metadata Manager displays `Untitled` as the name for a metadata object before you configure it. The properties you configure depend on the properties you created for the class. For example, the AccessSchema class includes the Name, Label, and Description properties.

1. In the **Catalog** view, select the metadata object or resource for which you want to create child objects.
2. Click **Actions** > **New** and select the name of the class for which you want to create the metadata object.

   The **Create Custom Objects** window appears.
3. Configure the metadata object properties.

   The name cannot contain the following characters:

   `/ \ : * ' ? " < > | [ ]`
4. Click **Add**.

   Metadata Manager adds the metadata object to the **Create Custom Objects** window.
5. If you want to add additional metadata objects, configure the metadata object properties and click **Add** for each additional object.
6. Click the arrow icons to navigate between metadata objects you created to configure the properties.
7. Optionally, select a metadata object you created and click **Delete** to delete the object.
8. Click **OK**.

Metadata Manager adds the metadata objects to the metadata catalog.

Deleting Custom Resources and Metadata Objects

You can delete any custom resource or custom metadata object from the metadata catalog. You can delete resources and metadata objects based on the permissions for the resource and metadata objects.

You can delete resources or metadata objects for which you have write permission on the resource or metadata object and all the child objects. If you do not have write permission on all the child objects, Metadata Manager deletes any child objects on which you have write permission that do not have any child objects.

For example, you created a custom resource, AccessDB, with a schema object ACCESS_DB_SOURCE. ACCESS_DB_SOURCE contains a child table object named CUSTOMERS. CUSTOMERS has no child objects. If you have write permissions on the resource and all objects and delete the resource, Metadata Manager deletes the resource and all child objects. However, if you have read permission on ACCESS_DB_SOURCE and write permission on CUSTOMERS, and delete the resource, Metadata Manager only deletes CUSTOMERS.

You can use the **Actions** menu or right-click menu to delete a resource or object from the following **Catalog** view on the **Browse** tab.

1. Select the metadata object or resource you want to delete.
2. Click **Actions** > **Delete**.

Editing Metadata Object Properties

You can edit custom metadata object properties or metadata object properties that you added to packaged or universal models on the **Model** tab. Edit custom metadata object properties and business name properties in the **Edit Metadata** window.

You can also edit custom metadata object properties and business name properties for packaged or universal resource types by exporting the properties to an Excel file. Edit the properties in the Excel file, and then import the properties into the metadata catalog.

You can use the **Actions** menu or right-click menu to edit properties for a single object from the following areas on the **Browse** tab in Metadata Manager:

- **Shortcuts** view
- **Catalog** view
- **Glossary** view
- Details panel
- Data lineage analysis

1. Select the object whose properties you want to edit.
2. Click **Actions** > **Edit Properties**.

   The **Edit Properties** window appears.
3. Edit the applicable properties.
4. Click **OK**.

Exporting and Importing Custom Properties

You can edit the values for custom properties that you add to packaged or universal model classes and business name properties using Microsoft Excel. Export custom properties and business name properties from the metadata catalog to an Excel file. Use the Excel file to edit values for the properties, and then import the properties from the Excel file into the metadata catalog.

The Excel file contains a worksheet for each object type you export. Each worksheet contains properties for all metadata objects for a specific object type.

The following table describes the content of each worksheet:

Row or Column Name	Description
Export Root Path	Metadata catalog root path for the metadata objects in the worksheet. Do not edit the Export Root Path.
Class Identifier	Name and path of the class in the Metadata Manager repository for the metadata objects in the worksheet. Do not edit the Class Identifier.
Element ID	Resource and object name for a metadata object. Metadata Manager displays the Element ID in the following format: <resource name>.<object name>.
Element Path	Hierarchical path for a metadata object in the metadata catalog.
Business Name	Business name property for a metadata object.
Custom Attribute Name	Custom property for a metadata object.

Exporting Custom Properties

When you export custom and business name properties, Metadata Manager exports the property values for the selected metadata objects and any child objects to the Excel file.

To limit the custom and business name properties that Metadata Manager exports, configure catalog preferences. Metadata Manager only exports the resources and object types you configure in catalog preferences.

To export custom and business name properties:

1. On the Browse tab, configure preferences to limit the object types you want to export.
2. In the Catalog view, select the resource, logical group, or metadata object for which you want to export properties.
3. Click Actions > Export Metadata > Excel.
4. Click Yes in the Note window to include the business name property.
5. Open or save the Excel file.

   The options to save or download in the Excel file depend on your browser.

Editing Custom Properties

Open the Excel file to edit custom and business name properties.

To edit custom and business name properties:

1. In Microsoft Excel, open the Excel file that contains the exported properties.
2. Select the worksheet that contains the properties for the class type of the objects you want to edit.
3. In the row that contains the object for which you want to edit properties, enter the property value in the appropriate column.
4. Repeat steps 2 to 3 for all object types and properties that you want to edit.
5. Save the Excel file.

Importing Custom Properties

Metadata Manager updates the properties for the objects in the metadata catalog with the custom or business name properties in the Excel file.

Metadata Manager does not import properties for objects that no longer exist in the catalog. If you delete an object from the catalog, Metadata Manager ignores the properties for the object when you import properties.

To import custom and business name properties:

1. In the Catalog view, click Actions > Import Metadata > Excel.

   The Import Catalog Metadata window appears.
2. Click Browse and select the Excel file.
3. Click Import.

   Metadata Manager imports the properties and displays the number of metadata objects that were updated, not changed, not found, or not valid.

Rules and Guidelines for Exporting and Importing Custom Properties

Use the following rules and guidelines when you work with Excel files:

- If you export a large number of objects for which there are custom or business name properties, you cannot perform any operation in Metadata Manager until all the properties are exported.
- If the number of metadata objects in a class is greater than the number of rows in an Excel spreadsheet, Metadata Manager does not export all objects for the object type. Metadata Manager can export a maximum of 65,536 objects for each worksheet.

CHAPTER 9

Custom Resource Migration

This chapter includes the following topics:

Custom Resource Migration Overview

You can migrate custom resources between Metadata Manager instances. Use Metadata Manager to migrate the models, load templates, resource configurations, and linking rule sets associated with a custom resource from one Metadata Manager repository to another. Migrate custom resources so that you do not have to re-create them in the target environment.

You might migrate custom resources when you move from a development to a production environment. Migrate the resources, and then load them to import the metadata and create the data lineage links in the target environment.

Note: The information in this chapter applies to migrating custom resources within the current Metadata Manager version. For information about migrating packaged or universal resources, see the *Metadata Manager Administrator Guide*.

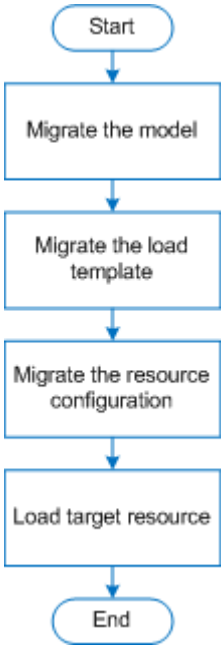
Custom Resource Migration Steps

To migrate a custom resource between Metadata Manager instances of the same version, you must migrate the models and resources to the target environment. To migrate objects, you export them from the source repository and import them into the target repository.

The method that you use to migrate a custom resource varies based on whether the resource uses a load template. If the resource uses a load template, you migrate the model, load template, and resource configuration, and then load the resource in the target environment. If the resource does not use a load template, you migrate the model, create the resource in the target environment, and then migrate the resource metadata.

Migrating a Custom Resource that uses a Load Template

The following image shows the process to migrate a custom resource that uses a load template:

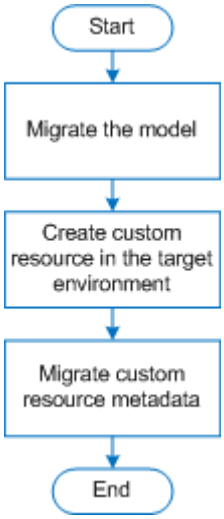


1. Migrate the model:
   a. Export the model from the source repository. On the **Model** tab, select **Actions** > **Export Models**. If the model has associated rule set definitions, include them in the export file.
   b. Import the model into the target repository. On the **Model** tab, select **Actions** > **Import Models**.
2. Migrate the load template:
   a. Export the load template from the source repository. On the **Model** tab, right-click the model, and select **Get Load Template**.
   b. Upload the load template into the target repository. On the **Model** tab, right-click the model, and select **Upload Load Template**.
3. Migrate the resource configuration:
   a. Export the resource configuration from the source repository. On the **Load** tab, select **Actions** > **Export Resource Configuration**. If the resource has associated rule sets or parameter definitions, include them in the export file.
   b. Import the resource configuration into the target repository. On the **Load** tab, select **Actions** > **Import Resource Configuration**.
4. Load the resource in the target environment.

   Load the resource to import the metadata and create the data lineage links.

Migrating a Custom Resource that does not use a Load Template

The following image shows the process to migrate a custom resource that does not use a load template:



1. Migrate the model:
   a. Export the model from the source repository. On the **Model** tab, select **Actions** > **Export Models**. If the model has associated rule set definitions, include them in the export file.
   b. Import the model into the target repository. On the **Model** tab, select **Actions** > **Import Models**.
2. In the target environment, create a custom resource with the same name as the resource in the source environment.
3. Migrate the custom resource metadata:
   a. Export the resource metadata to an XML file. On the **Browse** tab, click **Actions** > **Export Metadata** > **XML**.
   b. Import the resource metadata into the target repository. On the **Browse** tab, click **Actions** > **Import Metadata** > **XML**.

Model Migration

To migrate a model, you export the model from the source Metadata Manager repository to an XML file. You then import the XML file into the target Metadata Manager repository.

The model export XML file contains all the classes, attributes, and relationships for the model. When you import a model, Metadata Manager analyzes the contents of the XML file and compares it to the existing models. If the model does not exist, Metadata Manager creates the model. If the model exists, Metadata Manager imports the new or changed classes and relationships.

For example, you export a custom model from the development environment and then import it into the production environment. If you add a class to the custom model in the development environment, export it, and re-import it into the production environment, Metadata Manager imports the class that you added.

Exporting a Model

Export models to a model export XML file. You can include rule set definitions in the export file.

You can export packaged, universal, or custom models. You can export one packaged or universal model to an export file. You can also export one or multiple custom models to an export file. However, you cannot export a combination of packaged and custom models to one export file.

Note: You cannot export the Business Glossary model from Metadata Manager. To export business glossary assets and templates, use the Analyst tool.

1. On the **Model** tab, click **Actions** > **Export Models**.

   The **Export Model** window appears.

2. Select the models to export and add them to the **Selected models** list.

   Note: You can export multiple models to one export file only if all of the selected models are custom models.

3. To include the rule set definitions associated with the selected models in the export file, select **Include rule set definitions**.

   If you select this option, Metadata Manager creates a .zip file that contains the model export XML file plus an XML file for each rule set definition. If you do not select this option, Metadata Manager creates an XML file for the selected models.

4. Click **Export**.

   The options to save the XML file vary based on the browser.

Importing a Model

Use the Import Model wizard to import models into a Metadata Manager repository. When you import a model, you select the XML or .zip file that contains models and select the models you want to import.

The Import Model wizard analyzes and validates the file that you select. If the model does not exist in the Metadata Manager repository, Metadata Manager imports the entire model. If the model exists, Metadata Manager imports the new and changed classes and relationships. If the model in the file matches the model in the Metadata Manager repository, Metadata Manager does not import the model.

Note: You cannot import a Business Glossary model in Metadata Manager. To import business glossary assets and templates, use the Analyst tool.

1. On the **Model** tab, click **Actions** > **Import Models**.

   The **Import Model** window appears.

2. Select the XML or .zip file that contains the models that you want to import.
3. Click **Next**.
4. Select the models that you want to import, and click **Next**.

   The wizard analyzes the file and validates it against the models in the Metadata Manager repository. The wizard lists the classes and relationships that Metadata Manager will create and update, and the classes and relationships that are not affected by the import process.

5. Click **Import**.

Load Template Migration

If the custom resource that you migrate uses a load template, migrate the load template that is associated with the model. To migrate the load template, export it from the source repository and upload it into the target repository. Export and upload a load template on the **Model** tab.

To export a load template, open the Metadata Manager instance in the source environment. On the **Model** tab, right-click the model, and select **Get Load Template**.

To upload a load template, open the Metadata Manager instance in the target environment. On the **Model** tab, right-click the model, and select **Upload Load Template**. When you upload the load template, Metadata Manager validates the metadata against the model to prevent repository inconsistencies.

Resource Configuration Migration

If the custom resource that you migrate uses a load template, migrate the resource configuration after you migrate the load template. To migrate the resource configuration, you export the resource configuration from the source Metadata Manager repository to a file. You then import the file into the target Metadata Manager repository.

You can include rule sets and parameter definitions in the export file. If you export the resource configuration without the rule sets and parameter definitions, Metadata Manager creates a resource configuration file with the .rcf extension. If you include the rule sets and parameter definitions, Metadata Manager creates a compressed file with the .rcz extension. The compressed file contains the resource configuration file, rule set files, rule set parameter files, and enumerated links files.

When you import the .rcf or .rcz file, Metadata Manager creates the resource if it does not exist. If the resource exists, Metadata Manager updates the resource configuration.

After you import the resource configuration into the target environment, load the resource to import the metadata and create the data lineage links.

Related Topics:

- "Resource Configuration File" on page 152

Exporting a Resource Configuration

Export the resource configuration from the source repository so that you can import it into the target repository. If the resource includes rule sets or parameter definitions, include them in the export file.

1. On the **Load** tab, select a resource.
2. Click **Actions** > **Export Resource Configuration**.
3. Metadata Manager prompts you to include rule sets and parameter definitions in the export file.

   Select one of the following options:

Option	Description
Yes	Export all source files associated with the resource, including rule sets and parameter definitions.

Option	Description
No	Export the resource configuration only.

4. Metadata Manager prompts you to include the resource password in the export file.

   Custom resources do not include passwords, so select **No**.

   The options to save the export file vary based on the browser.

If you include rule sets and parameter definitions in the export file, Metadata Manager creates a file with the .rcz extension in the location you specify. If you export the resource configuration only, Metadata Manager creates a file with the .rcf extension in the location you specify.

Importing a Resource Configuration

Import the resource configuration into the target repository. If you included rule sets and parameter definitions when you exported the resource from the source repository, Metadata Manager imports the resource, rule set files, rule set parameter files, and all data and parameter files that are associated with the resource.

1. On the **Load** tab, click **Actions** > **Import Resource Configuration**.

   The **Import Resource Configuration** dialog box appears.

2. Select the import file.

   If you import a resource configuration file with the .rcf extension, Metadata Manager imports the resource configuration. If you import a resource configuration file with the .rcz extension, Metadata Manager imports the resource configuration plus all rule sets, parameter definitions, and enumerated links associated with the resource.

3. Custom resources do not include passwords or secure JDBC parameters, so leave the **Password** and **Secure JDBC Parameters** properties empty.

4. Click **OK**.

   If the resource does not exist, Metadata Manager creates it. If the resource already exists, Metadata Manager updates the resource configuration.

After you import the resource configuration, load the resource to import the metadata and create the data lineage links.

Custom Resource Metadata Migration

If the resource that you migrate does not use load templates, migrate the resource metadata. Migrate the resource metadata after you migrate the model and create the resource in the target environment.

To migrate custom resource metadata, export it from the source Metadata Manager repository to an XML file. Then import the XML file into the target repository.

The custom metadata XML file contains all metadata objects for the resource. When you import the XML file for a custom resource, Metadata Manager imports all metadata objects for the resource from the XML file.

Export and import custom metadata between Metadata Manager repositories with the same version. You cannot import custom metadata from a different version.

Exporting Custom Resource Metadata

You export custom resource metadata to an XML file.

1. On the **Browse** tab, select the resource for which you want to export metadata.
2. Click **Actions** > **Export Metadata** > **XML**.

   The options to save the XML file vary based on the browser.

Importing Custom Resource Metadata

You import custom resource metadata from an XML file.

Before you import metadata for a resource, verify that the model and resource exist.

1. On the **Browse** tab, select the custom resource for which you want to import metadata.
2. Click **Actions** > **Import Metadata** > **XML**.

   The **Import Catalog Metadata** window appears.
3. Select the XML file that contains the resource metadata, and click **Import**.

Metadata Manager imports all metadata objects for the resource from the XML file

Rules and Guidelines for Importing Custom Metadata

Review the following rules and guidelines before you import custom metadata:

- If you export comments and links, verify that the users who added the comments and links are registered in the target Metadata Manager instance.

  Metadata Manager does not import a comment or link if the user is not registered in the target instance.
- Metadata Manager writes an entry in the log when it does not import a custom object or custom attribute value.

  For example, Metadata Manager writes a log entry if you try to import custom metadata when the source and target models differ.

Part III: Custom XConnect Created with the Custom Metadata Configurator

This part contains the following chapters:

CHAPTER 10

Custom Metadata Configurator

This chapter includes the following topics:

Custom Metadata Configurator Overview

Use the Custom Metadata Configurator to load metadata for a custom model from metadata source files.

Use the Custom Metadata Configurator to configure the custom resource template that contains the metadata source file format and to generate the custom PowerCenter objects in the PowerCenter repository. The PowerCenter objects include the sessions, mappings, and workflows that Metadata Manager uses to load the metadata from the metadata source files.

Complete the following tasks to create a template and the custom PowerCenter objects:

1. Create the element and association metadata files.
2. Log in to the Custom Metadata Configurator.
3. Configure the custom resource template.
4. Configure delimiters for the metadata files.
5. Map attributes in the element metadata file to the attributes in the custom model.
6. Map relationships in the association metadata file to the relationships for the classes in the custom model.
7. Add class rules to select particular records in an element or association metadata.
8. Generate the PowerCenter objects required to load metadata from the metadata source files into the Metadata Manager warehouse.

Note: The Custom Metadata Configurator refers to relationships as associations.

After you create the custom resource template and generate the PowerCenter objects, you configure the custom resource on the **Load** tab. You can configure the template name and the metadata source files.

Step 1. Create Metadata Source Files

Before you log in to the Custom Metadata Configurator, create the element metadata file and the association metadata files that contain the metadata source information. You use the files to map attributes and associations to the classes you created for the model on the **Load** tab.

Create the following metadata source files:

Element metadata file

Contains metadata object names and attributes. The metadata objects are instances of the classes you defined in the model. You map the object attributes in the element metadata file to the class attributes defined in the corresponding model you created on the **Model** tab.

Association metadata file

Contains relationships between metadata objects in an association metadata file. The relationships between metadata objects are instances of the class associations you defined in the model. You use the Custom Metadata Configurator, which exposes the information in the association metadata file, to map the associations between the objects.

If you want to preview the data in the element and association metadata files in the Custom Metadata Configurator, you must map the PATH environment variable to the location of the Microsoft Excel excel.exe executable.

Metadata Source File Rules and Guidelines

Certain rules and guidelines apply when you create the element and association metadata source files.

Use the following rules and guidelines to create the element and association metadata source files:

- You can store the metadata object attributes and associations together in the same file or in separate files.
- The association metadata file must identify the From and To objects in each association. It should store one or more object attributes to enable you to uniquely identify each From and To object in the file and store object attributes in separate columns.
- When you specify an association between custom objects and objects in a packaged or universal resource, use the ELEMENT_ID attribute to identify the objects that you want to associate. The ELEMENT_ID attribute is in view IMA_ELEMENT in the database schema that stores the Metadata Manager repository. You can look up ELEMENT_ID by NAME_PATH in the IMA_ELEMENT view.
- Date values in a metadata source file must be in the format `yyyy/MM/dd`.
- The custom resource fails to load some metadata objects, attributes, and relationships if the metadata sources file names contain any spaces or the following characters:

  ```
  ~ ' ! % ^ & * ( ) - + = { } [ ] | \ : ; " ' < > , . ? /
  ```

  If the column names contain spaces or any of these characters, the Custom Metadata Configurator converts them to underscores when you generate the PowerCenter objects. After converting the characters and spaces to underscores, columns might have the same modified name. The Custom Metadata Configurator ignores one of the columns when you generate the workflows.

- The association and element metadata file names cannot start with a number. If the file name starts with a number, PowerCenter object generation fails.

Step 2. Log In to the Custom Metadata Configurator

To log in to the Custom Metadata Configurator, start the application and connect to the Metadata Manager warehouse database that contains the custom model for the metadata source files you want to load.

1. Click **Start** > **Programs** > **Informatica <version>** > **Client** > **PowerCenter Client** > **Custom Metadata Configurator**.

   The **Informatica Custom Metadata Configurator** login window appears.
2. Enter the connection properties.
3. Click **OK**.

Custom Metadata Configurator Connection Properties

To log in to the Custom Metadata Configurator, enter the connection properties.

The following table describes the connection properties:

Property	Description
Previous Connections	List of connection strings used to connect to Metadata Manager warehouses. Select a Metadata Manager warehouse from the list, or specify connection information for another warehouse.
User ID	User account for the Metadata Manager warehouse database.
Password	Password for the Metadata Manager warehouse database user account.

Property	Description
Database Type	Type of database for the Metadata Manager warehouse database.

Connection String	JDBC connection string for the Metadata Manager warehouse database. The connection string depends on the database type you select:
	- For IBM DB2, use the following connection string:
	`jdbc:informatica:db2://[host name]:1521;DatabaseName=[database name]`
	- For Microsoft SQL Server, use the following connection string:
	`jdbc:informatica:sqlserver://[host name]:1521;DatabaseName=[database name]`
	To authenticate the database user credentials using Windows authentication and establish a trusted connection to a Microsoft SQL Server database, append `;AuthenticationMethod=ntlm` to the connection string.
	- For Oracle, use the following connection string:
	`jdbc:informatica:oracle://[host name]:1521;SID=[sid]`
	You can enter the SID or use the full service name. For example:
	`jdbc:informatica:oracle://[host name]:1521;ServiceName=[service name]`
	If the Oracle database uses the Advanced Security Option, use the following connection string:
	`jdbc:informatica:oracle://[host name]:1521;SID=[SID];EncryptionLevel=[encryption level];EncryptionTypes=[encryption types];DataIntegrityLevel=[data integrity level];DataIntegrityTypes=[data integrity types]`
	Note: If secure communication is enabled for the Metadata Manager warehouse database, you must configure additional JDBC parameters in the connection string.

Note: You can override the default database code page for the Metadata Manager warehouse database when you log in. Override the code page if the custom resource templates contain characters that the database code page does not support. For example, the Custom Metadata Configurator does not retrieve saved templates correctly. To override the code page, append the CODEPAGEOVERRIDE parameter to the connection string and specify a compatible code page.

For example, use the following JDBC URL to override the default code page with MS932:

```
jdbc:informatica:sqlserver://myhost:1433;DatabaseName=mm861;CODEPAGEOVERRIDE=MS932;
```

JDBC Parameters for Secure Databases

If secure communication is enabled for the Metadata Manager warehouse database, you must append additional JDBC parameters to the connection string.

Append the following parameters to the connection string:

```
;EncryptionMethod=SSL;TrustStore=<truststore
location>;TrustStorePassword=<password>;HostNameInCertificate=<host
name>;ValidateServerCertificate=<true|false>;KeyStore=<keystore
location>;keyStorePassword=<password>
```

Configure the parameters as follows:

EncryptionMethod

Encryption method for data transfer between Metadata Manager and the database server. Must be set to SSL.

TrustStore

Path and file name of the truststore file that contains the security certificate of the database server.

TrustStorePassword

Password used to access the truststore file.

HostNameInCertificate

Host name of the machine that hosts the secure database. If you specify a host name, the Metadata Manager Service validates the host name included in the connection string against the host name in the security certificate.

ValidateServerCertificate

Indicates whether the Metadata Manager Service validates the certificate that the database server presents. If you set this parameter to true, the Metadata Manager Service validates the certificate. If you specify the HostNameInCertificate parameter, the Metadata Manager Service also validates the host name in the certificate.

If you set this parameter to false, the Metadata Manager Service does not validate the certificate that the database server presents. The Metadata Manager Service ignores any truststore information that you specify.

KeyStore

Path and file name of the keystore file that contains the security certificates that the Metadata Manager Service presents to the database server.

KeyStorePassword

Password used to access the keystore file.

Step 3. Configure the Custom Resource Template

A custom resource template stores information about how to map the metadata object attributes to the class attributes and can also store the class relationships between metadata objects. Create a template using attribute and association files.

Map the object attributes in an element metadata file to the class attributes configured in the model. Map class relationships defined in the model between objects in the custom source repository. Save the associations in a template.

You can edit or delete a custom resource template after you create it.

Edit a template to perform the following tasks:

- Add, edit, or delete class attribute maps to metadata objects in the element metadata file.
- Add, edit, or delete association maps between metadata objects.
- Delete classes from a template.
- Configure delimiters for the element and association metadata files.
- Change a class rule.

If you edit classes for a template, the changes occur in the Metadata Manager warehouse when you load the resource. Because of this, the metadata in the Metadata Manager warehouse matches the metadata in the element and association metadata files.

Delete a template when it becomes obsolete. If you delete a template, you can purge the metadata loaded by the custom resource from the Metadata Manager warehouse.

Custom Resource Template Properties

When you create a template, enter the configuration information, select the classes to map, and select the metadata element and association files.

The following table describes the template properties that you configure:

Property	Description
Template Name	Name of the template. The name must contain alphanumeric characters and cannot contain spaces. Maximum length is 255 characters. If you create multiple templates, each template name must be unique.
Repository Name	Custom resource you create on the **Load** page in Metadata Manager. If you load metadata objects from multiple custom metadata sources, select one of the custom resources.
Classes	Classes created in the model for the selected custom source repository. Use this list of classes to map the class attributes to the element metadata file.
Element Metadata File	Contains the attributes for each metadata object. Click **Configure** to configure the delimiters for the element metadata file. Click **Preview** to preview the element metadata file in Microsoft Excel.
Association Metadata File	Contains the information required to establish the associations between metadata objects in the metadata file. Click **Configure** to configure delimiters for the associations file. Click **Preview** to preview the metadata file in Microsoft Excel.

Note: You must create another template if the format of the metadata source files change after you create the template.

Creating a Custom Resource Template

Create a template on the **Configure Maps** tab of the Custom Metadata Configurator.

1. Click the **Template Summary** tab.
2. Click **Configure New Template**.

   The **Configure Maps** tab appears.
3. Enter the template properties.
4. To select the classes from the element metadata file to include in the template, click **Select**.
5. Optionally, to preview the metadata source file data in Microsoft Excel, click **Preview**.
6. Optionally, to delete all values set in the template, click **Clear All**.

   To delete all information in the template, including the attribute and association maps, click **Clear All**. This option is available if you have not saved the template.
7. Click **Save**.

Editing or Deleting a Custom Resource Template

Edit or delete a custom resource on the **Template Summary** tab of the Custom Metadata Configurator.

1. Select the Metadata Manager warehouse you used to create the template.

2. Click the **Template Summary** tab.
3. To edit the template, click **View/Edit** for the template that you want to edit.

   The **Configure Maps** tab appears.
4. Edit the template.

   You can edit the template properties, the attribute maps, and the relationship maps.
5. Click **Save**.

To delete the template, click **Delete** in the **Template Summary** tab.

Viewing a Custom Resource Template Summary

After you create and save a template, you can view the template under the corresponding custom resource on the **Template Summary** tab of the Custom Metadata Configurator.

1. Log in to the Custom Metadata Configurator.

   The **Template Summary** tab appears. It displays one of the following statuses for each template:

Status	Description
Saved	Indicates you saved the template, but did not generate the PowerCenter objects.
Last generated date	The date you last generated the PowerCenter objects

2. Select a template, and then click **View/Edit**.

   The template appears on the **Configure Maps** tab.

Step 4. Configure Delimiters for Files

To ensure that the Custom Metadata Configurator reads the element and association metadata files, configure the delimiters in the files. Use the **Configure Delimiters** window to configure the metadata source file delimiters.

The following table describes the properties you configure:

Option	Description
Start in Row	First row in the file that contains column headers, metadata, or relationships. Exclude preliminary rows that contain header information or no information. You must enter an integer greater than 0. Default is 1.
Header Row	File has a header row. Header rows of element metadata files are mapped to class attributes. Header rows of association metadata files are mapped to relationship attributes. Default is enabled.
Delimiter	Character that separates entries in the file. Default is a comma (,).
Text Qualifier	Character used to enclose text that should be treated as one entry. Use a text qualifier to disregard the delimiter character within text. Default is quotes (“).

Some element or association metadata files can have extra records that the Custom Metadata Configurator ignores. For example, the first five records in an element metadata file provide a description about the type of information in the file. Record six is the first record that contains object attributes. You can direct the Custom Metadata Configurator to start in row six.

Specify whether the element and association metadata files contain column headers. If the file does not have column headers, the Custom Metadata Configurator displays generic headers for each column, such as Column 1, Column 2, and Column 3.

Since you provide the object attribute and association information in flat files, you must specify the delimiter between records.

You can also specify the text qualifier to escape the delimiter character if it is used within text. For example, you specify a comma (,) as the delimiter in an element metadata file.

Configuring Delimiters for Files

Configure the metadata source file delimiters on the **Configure Maps** tab of the Custom Metadata Configurator.

1. On the **Configure Maps** tab, click **Configure** for the element or association metadata file.

   The **Configure Delimiters** window appears.
2. Configure the delimiter properties.
3. To set the current settings as the default for all element and association metadata files that have no settings, click **Set Default**.
4. Click **OK**.
5. Click **Save**.

Step 5. Map Class Attributes

The element metadata file stores the attributes for each metadata object. When you map the object attributes to the class attributes, you map the column of the element metadata file to a class attribute.

To map an attribute of an object to a class attribute, identify the class of the object. Then, map the attribute of the object to the attribute defined for the selected class.

Each column in the element metadata file contains an object attribute. To map the object attribute to the class attribute, map the header of each column in the element metadata file to a class attribute.

If the file does not contain column headers, the Custom Metadata Configurator displays generic header names, such as Column 1, Column 2, and Column 3.

Each model class has a Name attribute. Map an object attribute to the Name attribute for every class. You do not need to map object attributes to other class attributes.

You must also establish the primary key of the element metadata file. The Custom Metadata Configurator uses the primary key to identify each record in the element metadata file for a given class. The key can be a composite value if you need to specify multiple columns in the element metadata file to uniquely identify each record.

One element metadata file can contain object attributes for multiple classes of objects. You can map a column in the element metadata file to a class attribute that is common to multiple classes.

The following table shows an example of metadata elements in the element metadata file:

Object	Object_Description	Object_Type
Customer	Customer information.	Database Table
Address	Customer address.	Database Column
Product	Product information.	Database Table

The Object_Description column in the element metadata file describes all column and table metadata objects. In the Custom Metadata Configurator, you map the Object_Description column to the Description class attribute for the AccessTableColumn class. Next, you click **Apply to All**. The Custom Metadata Configurator also maps the Object_Description column to the Description class attribute for the AccessTable class.

Mapping Class Attributes

Map class attributes on the **Configure Maps** tab of the Custom Metadata Configurator.

1. On the **Configure Maps** tab, click **Map Information**.

   The **Attributes Map** tab appears.
2. For each class, map the columns in the element metadata file to the class attributes.

   To map an object attribute to the class attribute, select the class attribute from the **Class Attributes** column for each applicable column in the **File Columns** column.

   To display the list of possible values for the **Class Attributes** columns, click the field.

   Note: You must map the Name attribute for all classes.
3. Select the **Key** option for all element metadata file columns that are used to identify each record in the element metadata file.
4. To apply the common class attribute settings to all other classes, click **Apply to All**.
5. For each class, to select the records in the element metadata file that apply to the attribute map for a particular class, click **Add Rule**.

   If you map multiple classes, you must create a rule for each class.

Step 6. Map Class Relationships

The association metadata file stores relationships between objects. You can establish relationships between metadata objects by specifying an association between them. In the Custom Metadata Configurator, you can choose any relationship that belongs to the classes of the two objects involved in the relationship.

When you set the class relationship between two metadata objects, you specify the following information:

- The metadata source and class of each object in the relationship
- The From and To objects that participate in the relationship

Select all columns required to uniquely identify each object in the association metadata file.

You can create a relationship between two objects from the same metadata source. You cannot create relationships between a class in a custom metadata source and a class in a resource for which Metadata

Manager packages a resource type. Use the **Model** page in Metadata Manager to create a class-level relationship between a custom class for which Metadata Manager packages a resource type, and then create the object-level relationship on the **Browse** page.

You configure the relationship maps on the **Associations Map** tab.

The following table describes the relationship properties that you configure:

Property	Description
From Repository	Metadata source containing the From element.
From Class	Class of the From element.
From Element	Metadata object. Select the columns in the associations file that uniquely identify the From element in the association.
Association Type	Class association between the From class and To class.
Struct	Displays a link between the two associated objects in the lineage diagram. You can select either **Struct** or **Field**. Use if both objects are similar to data structures. The associated objects contain child objects.
Field	Displays a link between the two associated objects in the lineage diagram. You can select either **Struct** or **Field**. Use if both objects are fields. The associated objects do not contain child objects.
To Repository	Metadata source that contains the To element. Either the From repository or the To repository must refer to the metadata source that contains the custom metadata.
To Class	Class of the To element.
To Element	Metadata object. Select the columns in the associations file that identify the To element in the relationship.
Rule	Adds a rule to select particular records in the associations file.

Mapping Class Relationships

Map class relationships on the **Configure Maps** tab of the Custom Metadata Configurator.

1. On the **Configure Maps** tab, click **Map Information**.
2. Click the **Associations Map** tab.

   The grid appears for the association map.
3. To add an association, click **Add**.
4. Map the associations between each related From and To class.
5. To display values for a cell in the grid, click the cell.
6. To delete a relationship, highlight the relationship and click **Delete**.
7. To remove all relationships from the grid, click **Clear Table**.

   Note: If you click **Clear Table**, you cannot undo it.

Step 7. Add Class Rules to Files

Use class rules to select specific records in an element or association metadata file for an attribute or association map.

The following table describes the class rule properties you can configure:

Property	Description
File Column	Column in the element or association metadata file that you use to filter the records in the attribute or association map.
Operator	Operator you want to apply between the selected file column and the entered value.
Value	Value used to evaluate the operation.

If a file has different classes or an associations file has more than one association, use at least one column to identify each record in the file. Use a class rule to filter records in the file that do not apply to the class attribute or association map.

For example, an element metadata file contains objects for the following Microsoft Access database classes:

- AccessTableColumns
- AccessTables
- AccessSchema

The following table shows some of the records in the element metadata file:

Repository Name	Type	Object Name	Description
ACCESS_DB_SOURCE	Column	SRC_PRODUCT_ID_FK	Foreign key to the SRC_PRODUCTS table.
ACCESS_DB_SOURCE	Column	SRC_ORDER_QTY	Number of items sold for a particular sales order.
ACCESS_DB_SOURCE	Table	SRC_CUSTOMERS	Contains information about customers from website registry.
ACCESS_DB_SOURCE	Table	SRC_PRODUCTS	Contains information about products.
ACCESS_DB_SOURCE	Table	SRC_ORDERS	Contains information about orders made by customers that use the website.
ACCESS_DB_SOURCE	Repository Name	ACCESS_DB_SOURCE	Contains operational data store records for customers, sales, and products.

Create a rule using the Type column in the element metadata file to select the records that apply to each class. For example, the AccessTables class should only contain Microsoft Access database table objects. You can create a rule for the AccessTables class that selects the records based on the following condition:

```
Type = 'Table'
```

Create complex filter statements that involve more than one filter condition. To create a complex filter statement, use AND and OR to join two different conditions, and use parenthesis to group conditions. For example, you can create the following complex filter statement:

```
(FileColumnA = 'MappingInstances' OR FileColumnA = 'MappingShortcuts')
AND (FileColumnB = 'PowerCenterDemoRepository')
```

If you map multiple classes for a custom resource template, specify a rule for each class. If you do not specify a class rule when mapping multiple classes for a template, the following error displays when you save the template:

```
Class Rule is not defined.
```

If you map one class for a template, you do not have to provide a class rule.

You can create one rule for each class or association in a template.

Adding Class Rules to Files

Add class rules to files on the **Configure Maps** tab of the Custom Metadata Configurator.

1. On the **Configure Maps** tab, click **Map Information**.
2. Click the **Attributes Map** tab.

   The grid appears for the attributes map.
3. Click **Add Rules**.

   The **Rules Setup** window appears.
4. Configure the rule properties.
5. To add the selected filter, click **Add to Filter**.
6. To create complex filter statements, use the following options:

Option	Description
AND	Creates an intersection statement.
OR	Creates a union statement.
()	Groups statements.

7. Click **OK**.

Step 8. Generate the PowerCenter Objects

After you finalize the custom resource template, use the Custom Metadata Configurator to create the PowerCenter objects. The PowerCenter objects include the mappings, sessions, and workflows that extract metadata from the metadata files and load it into the Metadata Manager warehouse.

When you generate the PowerCenter workflows, the Custom Metadata Configurator creates the following PowerCenter objects:

PowerCenter mappings

For each template, the Custom Metadata Configurator creates one mapping for the class attribute map and one mapping for the association map.

PowerCenter sessions

The Custom Metadata Configurator creates one session for each PowerCenter mapping. It includes all sessions in a PowerCenter workflow.

PowerCenter workflows

The Custom Metadata Configurator creates one PowerCenter workflow for each template.

The Custom Metadata Configurator stores the PowerCenter objects in an XML file and then imports the XML file into the PowerCenter repository.

The following table describes the naming convention for the generated PowerCenter objects:

Object	Naming Convention
Mapping	M_<repository_name>_<template_name>_Element_Elmnt_Attr M_<repository_name>_<template_name>_Elmnt_Assoc
Session	S_<mapping_name>
Workflow	WF_<repository_name>_<template_name>_Custom_Metadata

Generating the PowerCenter Objects

Generate the PowerCenter mappings, sessions, and workflows on the **Template Summary** tab of the Custom Metadata Configurator.

1. Click the **Template Summary** tab.
2. Click **View/Edit** for the template you want to view.

   The template appears in the **Configure Maps** tab.
3. Click **Generate Workflow**.

   The Custom Metadata Configurator prompts you for the PowerCenter repository connection information.
4. Enter the user name and password that you use to connect to the PowerCenter repository, and click **OK**.

   The Custom Metadata Configurator generates the PowerCenter objects and imports the objects into the PowerCenter repository.

CHAPTER 11

Custom Metadata

This chapter includes the following topics:

Custom Metadata Overview

You can create custom metadata in the Metadata Manager warehouse. After you create a model, create a resource that represents the model and create metadata objects for the resource using the metadata catalog.

You can also edit the values for custom properties. You can edit metadata object properties that you created for custom classes and properties that you created for classes in packaged and universal models. For example, you created a custom attribute in a report class for Cognos ReportNet. You can edit the value for the report property.

You can create and edit metadata in the Metadata Manager warehouse using the following methods:

Create a custom resource and custom metadata objects.

Use the metadata catalog to add a resource based on a custom model, and add metadata objects defined by classes in the model.

After you create the resource and metadata objects, you can edit the properties and relationships for the metadata objects.

Edit custom properties.

Edit the values of custom properties for metadata objects.

You can also create object-level relationships for metadata objects after you create class-level relationships for custom classes on the **Model** tab.

Note: You can also use the Custom Metadata Configurator and the Metadata Manager **Load** tab to create a custom resource, create a template for a custom model and generate the PowerCenter objects required to load the metadata, and load the metadata for the resource.

Creating Custom Metadata Objects

Create a model with classes, properties, and relationships on the **Model** tab. You can create classes and subclasses. After you create the model, you can add metadata objects based on the classes you created.

Create metadata objects through the metadata catalog on the **Browse** tab in Metadata Manager. You can edit or delete resources and metadata objects.

Creating a Custom Resource

Create a custom resource based on a custom model. All child classes of the resource appear under the resource name in the metadata catalog. Create the resource in the **Create Custom Metadata** window.

The following table describes the Create Custom Metadata properties:

Property	Description
Name	Name of the custom resource.
Description	Description for the custom resource.
Preview of Model Hierarchy	Lists the child classes for the selected model. You can create the metadata objects based on these classes after you create the resource.

1. In the Catalog view on the Browse tab, click **Actions** > **New** > **Custom Metadata**.

   The **Create Custom Metadata** window appears.
2. On the left pane, select the model on which you want to base the custom resource.
3. Configure the resource properties.
4. Optionally, view the model hierarchy and the classes included in the model on which you base the custom resource.
5. Click **OK**.

Metadata Manager creates the resource. The resource appears in the metadata catalog.

Create Custom Metadata Objects

You can create custom metadata objects and add them to a custom resource. Add custom metadata objects based on the classes for the model.

When you create the custom metadata object, select the class for which you want to create the metadata object. Add metadata objects based on the model hierarchy you created on the **Model** tab. For example, when you add a metadata object to the AccessDB resource, you add metadata objects of type AccessSchema, because the AccessSchema class is the root class of the model.

Create custom metadata objects in the **Create Custom Objects** window. You can create multiple metadata objects through this window. If the Metadata Manager repository is in a database that uses case insensitive collation, you cannot create multiple metadata objects with the same name but different cases. For example, you cannot create objects Customer and CUSTOMER.

By default, Metadata Manager displays `Untitled` as the name for a metadata object before you configure it. The properties you configure depend on the properties you created for the class. For example, the AccessSchema class includes the Name, Label, and Description properties.

1. In the **Catalog** view, select the metadata object or resource for which you want to create child objects.

2. Click **Actions** > **New** and select the name of the class for which you want to create the metadata object.

   The **Create Custom Objects** window appears.

3. Configure the metadata object properties.

   The name cannot contain the following characters:

   `/ \ : * ' ? " < > | [ ]`

4. Click **Add**.

   Metadata Manager adds the metadata object to the **Create Custom Objects** window.

5. If you want to add additional metadata objects, configure the metadata object properties and click **Add** for each additional object.
6. Click the arrow icons to navigate between metadata objects you created to configure the properties.
7. Optionally, select a metadata object you created and click **Delete** to delete the object.
8. Click **OK**.

Metadata Manager adds the metadata objects to the metadata catalog.

Deleting Custom Resources and Metadata Objects

You can delete any custom resource or custom metadata object from the metadata catalog. You can delete resources and metadata objects based on the permissions for the resource and metadata objects.

You can delete resources or metadata objects for which you have write permission on the resource or metadata object and all the child objects. If you do not have write permission on all the child objects, Metadata Manager deletes any child objects on which you have write permission that do not have any child objects.

For example, you created a custom resource, AccessDB, with a schema object ACCESS_DB_SOURCE. ACCESS_DB_SOURCE contains a child table object named CUSTOMERS. CUSTOMERS has no child objects. If you have write permissions on the resource and all objects and delete the resource, Metadata Manager deletes the resource and all child objects. However, if you have read permission on ACCESS_DB_SOURCE and write permission on CUSTOMERS, and delete the resource, Metadata Manager only deletes CUSTOMERS.

You can use the **Actions** menu or right-click menu to delete a resource or object from the following **Catalog** view on the **Browse** tab.

1. Select the metadata object or resource you want to delete.
2. Click **Actions** > **Delete**.

Editing Metadata Object Properties

You can edit custom metadata object properties or metadata object properties that you added to packaged or universal models on the **Model** tab. Edit custom metadata object properties and business name properties in the **Edit Metadata** window.

You can also edit custom metadata object properties and business name properties for packaged or universal resource types by exporting the properties to an Excel file. Edit the properties in the Excel file, and then import the properties into the metadata catalog.

You can use the **Actions** menu or right-click menu to edit properties for a single object from the following areas on the **Browse** tab in Metadata Manager:

- **Shortcuts** view
- **Catalog** view
- **Glossary** view
- Details panel
- Data lineage analysis

1. Select the object whose properties you want to edit.
2. Click **Actions** > **Edit Properties**.

   The **Edit Properties** window appears.
3. Edit the applicable properties.
4. Click **OK**.

Exporting and Importing Custom Properties

You can edit the values for custom properties that you add to packaged or universal model classes and business name properties using Microsoft Excel. Export custom properties and business name properties from the metadata catalog to an Excel file. Use the Excel file to edit values for the properties, and then import the properties from the Excel file into the metadata catalog.

The Excel file contains a worksheet for each object type you export. Each worksheet contains properties for all metadata objects for a specific object type.

The following table describes the content of each worksheet:

Row or Column Name	Description
Export Root Path	Metadata catalog root path for the metadata objects in the worksheet. Do not edit the Export Root Path.
Class Identifier	Name and path of the class in the Metadata Manager repository for the metadata objects in the worksheet. Do not edit the Class Identifier.
Element ID	Resource and object name for a metadata object. Metadata Manager displays the Element ID in the following format: <resource name>.<object name>.
Element Path	Hierarchical path for a metadata object in the metadata catalog.
Business Name	Business name property for a metadata object.
Custom Attribute Name	Custom property for a metadata object.

Exporting Custom Properties

When you export custom and business name properties, Metadata Manager exports the property values for the selected metadata objects and any child objects to the Excel file.

To limit the custom and business name properties that Metadata Manager exports, configure catalog preferences. Metadata Manager only exports the resources and object types you configure in catalog preferences.

To export custom and business name properties:

1. On the Browse tab, configure preferences to limit the object types you want to export.
2. In the Catalog view, select the resource, logical group, or metadata object for which you want to export properties.
3. Click Actions > Export Metadata > Excel.
4. Click Yes in the Note window to include the business name property.
5. Open or save the Excel file.

   The options to save or download in the Excel file depend on your browser.

Editing Custom Properties

Open the Excel file to edit custom and business name properties.

To edit custom and business name properties:

1. In Microsoft Excel, open the Excel file that contains the exported properties.
2. Select the worksheet that contains the properties for the class type of the objects you want to edit.
3. In the row that contains the object for which you want to edit properties, enter the property value in the appropriate column.
4. Repeat steps 2 to 3 for all object types and properties that you want to edit.
5. Save the Excel file.

Importing Custom Properties

Metadata Manager updates the properties for the objects in the metadata catalog with the custom or business name properties in the Excel file.

Metadata Manager does not import properties for objects that no longer exist in the catalog. If you delete an object from the catalog, Metadata Manager ignores the properties for the object when you import properties.

To import custom and business name properties:

1. In the Catalog view, click Actions > Import Metadata > Excel.

   The Import Catalog Metadata window appears.
2. Click Browse and select the Excel file.
3. Click Import.

   Metadata Manager imports the properties and displays the number of metadata objects that were updated, not changed, not found, or not valid.

Rules and Guidelines for Exporting and Importing Custom Properties

Use the following rules and guidelines when you work with Excel files:

- If you export a large number of objects for which there are custom or business name properties, you cannot perform any operation in Metadata Manager until all the properties are exported.

- If the number of metadata objects in a class is greater than the number of rows in an Excel spreadsheet, Metadata Manager does not export all objects for the object type. Metadata Manager can export a maximum of 65,536 objects for each worksheet.

CHAPTER 12

Rule-Based Links

This chapter includes the following topics:

Rule-Based Links Overview

To run data lineage across metadata sources, you must create links between matching objects in different sources. Use rule-based links to define the rules that Metadata Manager uses to link objects. You define rule-based links in a linking rules XML file.

Create rule-based links to define the rules that Metadata Manager uses to link matching objects in a pair of resources. For example, you might create a linking rule that links a business glossary term to an Oracle column when the business term "Technical Name" field matches the Oracle column name.

Use rule-based links to create links between the following resource types:

- Custom resource to another custom resource
- Custom resource to a packaged resource
- Custom resource to a universal resource
- Custom resource to a business glossary resource
- Business glossary resource to a packaged resource
- Business glossary resource to a universal resource

You can also use rule-based links to create links between packaged resources, between universal resources, or between packaged and universal resources when connection assignments do not create all of the required links.

To use rule-based links, you create linking rules as expressions that Metadata Manager uses to create link relationships between matching objects across different resources.

When you define a linking rule, you specify the following information:

- Set of possible objects in a source resource
- Set of possible objects in a target resource
- Direction that indicates whether the link originates from the source object or from the target object
- Expression that defines which source and target objects match

You can create multiple linking rules for the pair of resources that you want to link. You group linking rules that apply to the same pair of resources into a linking rule set. A linking rule set is a group of rules that links objects between two resources.

You define a linking rule set in a linking rules XML file. You create different types of linking rules files based on how you want to apply a linking rule set. You can apply a linking rule set to a pair of models or a pair of resources.

Linking Rules Files

You define a set of linking rules in a linking rules file. A linking rules file contains a set of linking rules that Metadata Manager uses to link objects in a pair of resources.

The type of linking rules file that you create depends on whether you want to create linking rules for a pair of models or a pair of resources.

Define a linking rule set for a pair of models when you want to apply the same linking rules to different resources associated with the models. For example, you develop a custom model, "CustomETL." You need to link objects in CustomETL resources with objects in Oracle resources. You can create a set of linking rules that allow you to link objects in any CustomETL resource with objects in any Oracle resource.

To define a linking rule set for a pair of models, create the following types of linking rules files:

Rule set definition file

An XML file that defines a linking rule set for a pair of models. A rule set definition file takes a parameter file that identifies the pair of resources to which to apply the linking rules. The rule set definition file must conform to the structure of the rule set definition file XML schema definition (XSD).

After you create a rule set definition file, you upload it on the **Model** tab. Metadata Manager associates the rule set definition with the source and target model identified in the file.

Rule set parameter file

An XML file that specifies the pair of resources to which to apply a rule set definition. It also contains parameter values for resource-specific attributes such as connection names and table names. The rule set parameter file must conform to the structure of the rule set parameter file XSD.

After you create a rule set parameter file, you upload it to the source resource or the target resource on the **Load** tab. When you upload the file, Metadata Manager creates a rule set for the resources. To create the rule set, Metadata Manager substitutes the parameters defined in the rule set definition file with the parameter values defined in the rule set parameter file.

Define a linking rule set for a pair of resources when you want to apply the linking rules to specific resources. For example, you want to link objects in resource "BusinessGlossary1" with objects in resource "Oracle1."

To define a linking rule set for a specific pair of resources, create the following type of linking rules file:

Rule set file

An XML file that defines a linking rule set for pair of resources. The rule set file must conform to the structure of the rule set file XSD.

After you create a rule set file, you upload it to the source resource or the target resource on the **Load** tab.

You define linking rule sets in rule set definition files and in rule set files. Rule syntax is identical in both file types except that rules in rule set definition files can contain parameters. The parameters represent resource-specific attributes such as connection names, package names, and string literals such as table names.

Process to Use Rule-Based Links

Use rule-based links to define the rules that Metadata Manager uses to create links between resources.

The process to use rule-based links varies based on whether you create linking rules for a pair of models or a pair of resources.

Process to Use Rule-Based Links for Models

You can use rule-based links to create linking rules for a pair of models.

To create linking rules for a pair of models, complete the following steps:

1. Create a rule set definition file.

   Configure the rule set in the file. Also define parameters for connections, packages, and string literals that are specific to the source or target resource.

2. Upload the rule set definition file to the source model or the target model on the **Model** tab.

   When you upload the file, Metadata Manager creates the rule set definition in the Metadata Manager repository. Metadata Manager associates the rule set definition with the models that you specify in the rule set definition file.

3. For each pair of resources to which you want to apply the rule set definition, create a rule set parameter file.

   Specify the source resource and the target resource in the file. Also specify a value for each parameter that you define in the rule set definition file.

4. Upload the rule set parameter file to the source resource or the target resource on the **Load** tab.

   When you upload the file, Metadata Manager creates the rule set in the Metadata Manager repository with the same name as the rule set parameter name. Metadata Manager associates the rule set with the resources that you specify in the rule set parameter file.

5. Link the objects in the resources by completing one of the following tasks:

   - Load the resources. The load process uses the rules to create the links between matching objects.
   - Use the **Resource Link Administration** window in the **Load** tab to create the links. If you loaded the resources, direct Metadata Manager to use the rules to create the links between matching objects.

Process to Use Rule-Based Links for Resources

You can use rule-based links to create linking rules for a pair of resources.

To create linking rules for a pair of resources, complete the following steps:

1. Create a rule set file and configure the rule set in the file.
2. If the rule set file contains endpoint links for a PowerCenter resource, load the PowerCenter resource to ensure that Metadata Manager configures connection assignments for the resource.
3. Upload the rule set file to the source resource or the target resource on the **Load** tab.

   When you upload the file, Metadata Manager creates the rule set in the Metadata Manager repository. Metadata Manager associates the rule set with the resources that you specify in the rule set file.
4. Link the objects in the resources by completing one of the following tasks:
   - Load the resources. The load process uses the rules to create the links between matching objects.
   - Use the **Resource Link Administration** window in the **Load** tab to create the links. If you loaded the resources, direct Metadata Manager to use the rules to create the links between matching objects.

Endpoint and Non-Endpoint Links

When you define a linking rule, you specify whether you are linking an object in one resource to an endpoint or to a non-endpoint in another resource. An endpoint is an object that has a connection to another object in a different resource.

Business Intelligence, Data Integration, and Data Modeling resource types contain endpoints. For example, in a PowerCenter resource, a Source Qualifier port is an endpoint because it can have a cross-resource relationship to a table column in a database management resource. In a PowerCenter resource, a Filter port is not an endpoint because it cannot have a relationship to an object in another resource.

When you link another resource to a Business Intelligence, Data Integration, or Data Modeling resource, you usually link to endpoints. However, you can link to a non-endpoint object in these resources. The rule syntax required to link endpoints differs from the syntax required to link non-endpoints.

Custom, SAP R/3, database management, and business glossary resource types do not contain endpoints. When you use linking rules to link objects in a custom resource to an object in these resource types, use non-endpoint links.

Endpoints in PowerCenter Resources

A PowerCenter resource contains endpoint and non-endpoint objects. In the PowerCenter model, endpoint classes are classes that can have relationships to classes in another model.

The following table lists the endpoint classes in the PowerCenter model:

Parent Class	Endpoint Class
Source Qualifier Instance	Source Qualifier Port
Target Definition Instance	Target Definition Port

Parent Class	Endpoint Class
Lookup Procedure Instance	Lookup Transformation Port
Mapping	Stored Procedure Instance

All other PowerCenter classes are non-endpoint classes.

Endpoints in Business Intelligence, Data Modeling, and Informatica Platform Resources

Business Intelligence, Data Modeling, and Informatica Platform resources contain endpoints and non-endpoint classes. Endpoint classes are classes that can have relationships to classes in another model.

You can determine which classes in a model are endpoint classes. Metadata Manager includes text files that list the endpoint classes for Business Intelligence, Data Modeling, and Informatica Platform models. The endpoint class files are named `<model type>.endpoint.txt`.

The endpoint class files are located in the following directory:

```
<Installation directory>\services\MetadataManagerService\md-repo\xconnects
```

You can also determine whether an individual class is an endpoint. To determine whether an individual class is an endpoint, check the **Properties** panel on the **Model** tab. The **Is an endpoint** property indicates whether the class is an endpoint.

Upload a Rule Set Definition to a Model

After you configure a rule set definition file, upload the file to the source or target model to create the rule set definition in the Metadata Manager repository.

When you upload a rule set definition file, Metadata Manager associates the rule set definition with the source and target models configured in the file. If either the source or the target model does not exist, Metadata Manager still creates the rule set definition. When you create the source or target model, Metadata Manager associates the saved rule set definition with the model.

Uploading a Rule Set Definition

Upload a rule set definition file to the source model or the target model. Upload a rule set definition to create or update a rule set definition in the Metadata Manager repository. You can upload multiple rule set definition files for a model.

1. On the **Model** tab, select a model.
2. Click **Actions** > **Upload Rule Set Definition**.

   The **Upload Rule Set Definition** dialog box appears.
3. Click **Browse**, select the rule set definition XML file, and click **Open**.
4. Click **OK**.

   Metadata Manager uploads the file and creates or updates the rule set definition. It also validates the rule set definition. If the rule set definition is not valid, check the service log for more information.

The **Rule Set Definitions** tab in the **Properties** panel displays the rule set definition information for the model. To remove a rule set definition, select the rule set definition, and click **Delete**.

Upload a Rule Set to a Resource

After you configure a rule set file or a rule set parameter file, upload the file to the source resource or the target resource. Upload the file to create or update the rule set in the Metadata Manager repository.

When you upload a rule set file or a rule set parameter file, Metadata Manager associates the rule set with the source and target resources configured in the XML file. If either the source or the target resource does not exist, Metadata Manager still creates the rule set. When you create the source or target resource, Metadata Manager associates the saved rule set with the resource.

When you create or update a rule set, Metadata Manager does not create the links between the resources. To use the rules to link metadata objects between resources, you must load the resources or create the links in the **Resource Link Administration** window.

Uploading a Rule Set

Upload a rule set file or a rule set parameter file to the source resource or the target resource to create or update the rule set in the Metadata Manager repository. You can upload multiple rule set files and multiple rule set parameter files to a resource.

Before you upload a rule set file that contains endpoint links for a PowerCenter resource, load the resource. Before you upload a rule set parameter file for a resource, create and upload the rule set definition file to the source or target model.

1. On the **Load** tab, select a resource in the **Resources** panel.
2. In the **Properties** panel, click **Edit**.

   The **Edit Resource** window appears.
3. Click the **Linking Rules** tab.
4. Click **Upload**.

   The **Upload** dialog box appears.
5. Click **Browse**, select the rule set file or the rule set parameter file, and click **Open**.
6. Click **OK**.

   Metadata Manager uploads the file and creates or updates the rule set. It also validates the rule set. If the rule set is not valid, check the service log for more information.

The **Linking Rules** tab in the **Properties** panel displays the rule set information for the resource.

Removing a Rule Set

To remove a rule set from a resource, edit the resource and delete the rule set.

1. On the **Load** tab, select the resource in the **Resources** panel.
2. In the **Properties** panel, click **Edit**.

   The **Edit Resource** window appears.
3. Click the **Linking Rules** tab.

4. Select the rule set that you want to remove, and click **Delete**.
5. Click **OK** to close the **Edit Resource** window.

   Metadata Manager removes the rule set from the resource.

Link Objects Using Rules

After you define linking rules in a linking rules file and create the rule set in the Metadata Manager repository, create the links between the resources. After you create links, you can run data lineage analysis across the metadata sources.

To link the objects in the resources, complete one of the following tasks:

Load the resources.

The load process creates the links between matching objects.

Use the Resource Link Administration window.

Use the **Resource Link Administration** window in the **Load** tab to create the links. If you loaded the resources, direct Metadata Manager to create the links between matching objects.

Creating Links through the Resource Link Administration Window

When you create links through the **Resource Link Administration** window, Metadata Manager links objects in the resource to objects in other resources based on all of the linking rules.

1. On the **Load** tab, click **Actions** > **Resource Link Administration**.

   The **Resource Link Administration** window appears.
2. Select the resources that you want to link, and click **Create Links**.

   Metadata Manager adds the resource to the link queue, and then starts the link process.
3. To cancel a link process, select the resource, and click **Actions** > **Cancel** in the **Load** tab.

When the linking completes, Metadata Manager updates the Last Status Date and Last Status for the resource.

CHAPTER 13

Linking Rules File Configuration

This chapter includes the following topics:

Linking Rules File Configuration Overview

Define linking rule definitions or linking rule sets for a pair of models or a pair of resources in a linking rules file. A linking rules file is an XML file that contains a set of linking rules or that defines parameters associated with a rule set definition.

You can create the following types of linking rules files:

Rule Set Definition File

An XML file that defines a linking rule set for a pair of models. A rule set definition file takes a parameter file that identifies the pair of resources to which to apply the linking rules. A rule set definition file must conform to the structure of the rule set definition file XML schema definition (XSD).

Rule Set Parameter File

An XML file that specifies the pair of resources to which to apply a rule set definition. It also contains parameter values for connection names, package names, and string literals such as table names. A rule set parameter file does not contain linking rules. A rule set parameter file must conform to the structure of the rule set parameter file XML schema definition (XSD).

Rule Set File

An XML file that defines a linking rule set for pair of resources. A rule set file cannot contain parameters or parameter definitions. A rule set file must conform to the structure of the rule set file XML schema definition (XSD).

You define linking rule sets in rule set definition files and in rule set files. Rule syntax is identical in both file types except that the rules in rule set definition files can contain parameters. Create linking rules files with a text editor or XML editor.

Rule Set Definition File

A rule set definition file is an XML file that defines linking rules for a pair of models. You create a rule set definition file based on the rule set definition file XML schema definition (XSD).

To create a rule set definition file, create an XML file that includes the following XML elements:

```
<?xml version="1.0" encoding="UTF-8"?>

<ruleSetDefinition>

    <sourceModel/>
    <targetModel/>

    <param/>

    <rule>
        <sourceFilter>
            …
        </sourceFilter>
        <targetFilter>
            …
        </targetFilter>

        <link/>
    </rule>

</ruleSetDefinition>
```

The XML attributes and elements that these elements contain depend on whether you configure rules to link to endpoints or to link to non-endpoints.

Rule Set Definition File Elements

The XML elements in the rule set definition file define the rules that Metadata Manager uses to link objects in the source resource to matching objects in the target resource.

Use the following XML elements:

ruleSetDefinition

Contains a group of rules to link objects between two models. A ruleSetDefinition element contains a required name attribute. The ruleSetDefinition name must be unique in the Metadata Manager repository.

A ruleSetDefinition element must contain one sourceModel element, one targetModel element, and at least one rule element. A ruleSetDefinition element can also contain one or more param elements.

sourceModel

If you link endpoints, this element defines the name of the Business Intelligence, Data Integration, or Data Modeling model that contains endpoints. If you link non-endpoints, this element defines the name of one model that you want to link. Contains a required name attribute.

targetModel

If you link endpoints, this element defines the name of the model that does not contain endpoints. If you link non-endpoints, this element defines the name of the other model that you want to link. Contains a required name attribute.

param

Defines a parameter that represents a resource-specific attribute. A parameter can define a connection name, a package name, or a string literal. Contains a required name attribute and optional description, defaultValue, and type attributes.

The following table describes the values for the type attribute:

Value	Description
connection	The parameter defines a connection name for endpoint links.
package	The parameter defines a package name for endpoint links.
string	The parameter defines a string literal in an expression, for example, a table name. If you do not specify a value for the type attribute, the parameter type is string.

rule

Defines the linking rule. The XML attributes and elements that this element contains depends on whether you configure rules to link endpoint classes or to link non-endpoint classes.

Rule Set Parameter File

A rule set parameter file is an XML file that specifies the resources to which to apply a rule set definition. It also contains parameter values for resource-specific attributes. You create a rule set parameter file based on the rule set parameter file XML schema definition (XSD).

To create a rule set parameter file, create an XML file that includes the following required XML elements:

```
<?xml version="1.0" encoding="UTF-8"?>

<ruleSetParams>

    <sourceResource/>
    <targetResource/>

    <param/>

</ruleSetParams>
```

The XML attributes and elements that these elements contain depend on whether you configure rules to link to endpoints or to link to non-endpoints.

Rule Set Parameter File Elements

The XML elements in the rule set parameter file define the resource-specific attributes for a rule set definition. When you upload the rule set parameter file, Metadata Manager creates a rule set for the resource by substituting the parameter values for the parameters defined in the rule set definition.

Use the following required XML elements:

ruleSetParams

Contains the names of the source and target resource and the values for all parameters defined in a rule set definition.

The following table describes the attributes for the ruleSetParams element:

Attribute	Description
definition	Required. Identifies the rule set definition to which the rule set parameter file applies.
name	Required. A string that you specify so that you can update or delete the rule set. The name must be unique in the Metadata Manager repository.
description	Optional. Element description.

A ruleSetParams element must contain one sourceResource element, one targetResource element, and one param element for each parameter defined in the rule set definition file.

sourceResource

If you link endpoints, this element defines the name of the Business Intelligence, Data Integration, or Data Modeling resource that contains endpoints. If you link non-endpoints, this element defines the name of one resource that you want to link. Contains a required name attribute.

targetResource

If you link endpoints, this element defines the name of the resource that does not contain endpoints. If you link non-endpoints, this element defines the name of the other resource that you want to link. Contains a required name attribute.

param

Identifies the parameter and defines the parameter value. Contains a required name attribute and a required value attribute.

Rule Set File

A rule set file is an XML file that defines linking rules for a pair of resources. You create a rule set file based on the rule set file XML schema definition (XSD).

To create a rule set file, create an XML file that includes the following required XML elements:

```
<?xml version="1.0" encoding="UTF-8"?>

<ruleSet>

    <sourceResource/>
    <targetResource/>

    <rule>
        <sourceFilter>
            …
        </sourceFilter>
        <targetFilter>
            …
        </targetFilter>

        <link/>
    </rule>

</ruleSet>
```

The XML attributes and elements that these required elements contain depend on whether you configure rules to link to endpoints or to link to non-endpoints.

Rule Set File Elements

The XML elements in the rule set file define the rules that Metadata Manager uses to link objects in the source resource to matching objects in the target resource.

Use the following required XML elements:

ruleSet

Contains a group of rules to link objects between two resources. A ruleSet element contains a required name attribute. The ruleSet name must be unique in the Metadata Manager repository.

A ruleSet element must contain one sourceResource element, one targetResource element, and at least one rule element.

sourceResource

If you link endpoints, this element defines the name of the Business Intelligence, Data Integration, or Data Modeling resource that contains endpoints. If you link non-endpoints, this element defines the name of one resource that you want to link. Contains a required name attribute.

targetResource

If you link endpoints, this element defines the name of the resource that does not contain endpoints. If you link non-endpoints, this element defines the name of the other resource that you want to link. Contains a required name attribute.

rule

Defines the linking rule. The XML attributes and elements that this element contains depends on whether you configure rules to link endpoint classes or to link non-endpoint classes.

Rule Element Configuration for Endpoints

Use the rule element to define a linking rule that Metadata Manager uses to link endpoints in the source resource to matching objects in the target resource.

The following table describes the attributes for the rule element:

Attribute	Description
name	Required. Name of the rule. The rule name must be unique in the rule set.
direction	Optional. Indicates whether the link originates from the source object or from the target object. For endpoint links, the direction must be Auto. If you do not use the direction attribute, Metadata Manager uses Auto by default.

A rule element must contain the following elements:

sourceFilter

Filters the list of possible endpoints in the source resource that you want to link. A sourceFilter element must contain an endpoint element.

The following table describes the attributes for the endpoint element:

Attribute	Description
class	Optional. Name of the class that the endpoint belongs to. Use a pipe (|) to separate multiple class names. Either the class attribute or the type attribute is required.
type	Optional. Type of the class that the endpoint belongs to. Use a pipe (|) to separate multiple class types. Either the class attribute or the type attribute is required.
connection	Optional. Name of the connection that the endpoint uses to connect to an external source database. To view the list of all connections in the Business Intelligence, Data Integration, or Data Modeling resource, view the **Connection Assignment** tab in the resource properties. In a rule set definition file, you can use a parameter to represent the connection name. Enter the connection name in the following format: `connection="${<parameter_name>}"`
package	Optional. Name of the package or the database schema that the endpoint connects to in the external source database. To view the list of all schema names, view the **Connection Assignment** tab in the properties of the Business Intelligence, Data Integration, or Data Modeling resource. In a rule set definition file, you can use a parameter to represent the package name. Enter the package name in the following format: `package="${<parameter_name>}"`

targetFilter

Filters the list of possible objects in the target resource that you want to link. A targetFilter element must contain an XML element named element.

The following table describes the attributes for the XML element named element:

Attribute	Description
class	Optional. Name of the class that the object belongs to. Use a pipe (|) to separate multiple class names. Either the class attribute or the type attribute is required.
type	Optional. Type of the class that the object belongs to. Use a pipe (|) to separate multiple class types. Either the class attribute or the type attribute is required.
condition	Optional. Expression that filters the list of possible objects to link.

An element can contain another element. The objects selected through the class, type, and condition attributes for this element must be immediate children of the objects selected through the containing element.

To specify a feature in a packaged or universal resource, the structure or the parent class must also be selected in the targetFilter. For example, the following targetFilter element includes the parent table as a containing element of an Oracle column:

```
<targetFilter>
   <element class="Oracle Table">
      <element type="Column"/>
```

```
    </element>
</targetFilter>
```

link

Defines the expression that specifies which filtered source and target objects Metadata Manager links. Contains a required condition attribute.

Expressions to Link Endpoints

When you define a linking rule, you enter expressions in condition attributes. Expressions filter the list of possible objects to link and define which filtered objects Metadata Manager links.

You can include an expression in a condition attribute for the following elements in a rule set definition file or a rule set file configured to link endpoints:

- targetFilter elements use the condition attribute as a selection expression. A selection expression filters the list of possible objects to link in the target resource.
- link elements use the condition attribute as a link expression. A link expression defines which filtered objects are linked. Metadata Manager links the objects that meet the condition defined in the link expression.

You can use the following operators in selection expressions and link expressions:

- Logical operators AND and OR.
- Comparison operators =, !=, IS NULL, and IS NOT NULL.
- Parentheses to group multiple conditions.

In a rule set definition file, you can use parameters to represent string literals in selection expressions and link expressions. Enter each parameter in the format `${<parameter_name>}`. For example:

```
<link condition="source.structName=${ref.name}" />
```

Selection Expressions for Endpoints

You can include class attributes in the selection expression. Use string literal values to define the values of attributes.

You can use any class attribute in a selection expression except for the following attributes:

- Object Class
- Location
- Source Creation Date
- Source Update Date
- MM Creation Date
- MM Update Date

If the attribute name contains spaces, enclose it in the XML character entity for quotation marks, `&quot;`. For example, if the attribute name is "Business Name," enter the name as follows:

```
&quot;Business Name&quot;
```

Literal values for attributes can use the following special characters:

```
! @ # $ % ^ { } | ?
```

Example

The following selection expression filters the list of possible target objects to objects in the class MyCustomClass that have a Description attribute with a value of "MyDescription":

```
<targetFilter>
   <element class="MyCustomClass" condition="Description='MyDescription'">
   </element>
</targetFilter>
```

Link Expressions for Endpoints

A link expression for endpoints must refer to both the source and target resources. When you create link expressions for endpoints, you can use keywords and specific endpoint attributes.

To refer to an object attribute in a link expression, use the following format:

```
<keyword>.<attribute>
```

The following table describes the keywords that you can use in link expressions:

Keyword	Description
source	Represents the object selected in the sourceFilter.
target	Represents the object selected in the targetFilter.
parent	Represents the parent of the object selected in the sourceFilter or targetFilter.

You can use the structName, featureName, and packageName attributes for endpoints in a link expression. You cannot use any other class attribute.

The following table describes the endpoint attributes that you can use in link expressions that link endpoints:

Endpoint Attribute	Description
structName	Structure name. A structure is a metadata object that contains fields. For example, a structure can be a PowerCenter source definition instance or an Oracle table.
featureName	Feature name. A feature is a field in a metadata object. For example, a field can be a PowerCenter source definition port or an Oracle table column.
packageName	Package name. A package is the schema for a database resource that the connection is assigned to.

To refer to object attributes in the target resource, use the same class attributes that are valid for selection expressions.

If the attribute name contains spaces, enclose it in the XML character entity for quotation marks, `&quot;`. For example, if the attribute name is "Business Name," enter the name as follows:

```
&amp;quot;Business Name&amp;quot;
```

When you include multiple conditions in a link expression, Metadata Manager evaluates the conditions from left to right. For better performance during linking, write a condition that checks for parent attributes before a condition that checks for child attributes.

Example

The following link expression links a source endpoint to a target object when the parent names and the object names match:

```
<link condition="source.structName = target.parent.Name AND source.featureName =
target.Name"/>
```

Sample Rule Set Definition Files to Link Endpoints

The following code shows a sample rule set definition file that defines linking rules between a PowerCenter model and a Microsoft SQL Server model:

```
<?xml version="1.0" encoding="UTF-8"?>
<ruleSetDefinition name="test_ep_to_nep_def" description="Link on ref table name">

    <sourceModel name="PowerCenter"/>
    <targetModel name="SQLServer"/>

    <param name="connection" description="PowerCenter connection name"
type="connection" />
    <param name="ref.name" description="Source qualifier reference table name" />
    <param name="table.name" description="Table name" />

    <rule name="source target">

        <sourceFilter>
            <endPoint connection="${connection}" class="Source Qualifier Instance"/>
        </sourceFilter>

        <targetFilter>
            <element class="Sqlserver Table" condition="Name=${table.name}">
                <element class="Sqlserver Column"/>
            </element>
        </targetFilter>

        <link condition="source.structName=${ref.name} AND target.parent.Name =
source.structName AND target.Name = source.featureName" />

    </rule>

</ruleSetDefinition>
```

The following code shows a sample of a rule set parameter file that defines the source and target resources and the parameter values for the rule set definition file:

```
<?xml version="1.0" encoding="UTF-8"?>
<ruleSetParams definition="test_ep_to_nep_def" name="INFA27003" description="Link rbl_pc
to RBLSchema">

    <sourceResource name="rbl_pc" />
    <targetResource name="RBLSchema" />

    <param name="connection" value="rbltest" />
    <param name="ref.name" value="CUSTOMER" />
    <param name="table.name" value="CUSTOMER" />

</ruleSetParams>
```

Sample Rule Set Files to Link Endpoints

The following code shows a sample rule set file that defines linking rules between custom resource objects and PowerCenter resource endpoints:

```
<?xml version="1.0" encoding="UTF-8"?>
<ruleSet name="Link custom objects to PowerCenter endpoints">
```

```
    <sourceResource name="MyPowerCenterResource"/>
    <targetResource name="MyCustomResource"/>

    <rule name="Link custom columns to PowerCenter Source Qualifier or Lookup ports">

        <sourceFilter>
            <endPoint connection="MyConnection" class="Source Qualifier Port|Lookup
Transformation Port"/>
        </sourceFilter>

        <targetFilter>
            <element class="Library" >
                <element class="Table">
                    <element class="TableColumn"/>
                </element>
            </element>
        </targetFilter>

        <!--Link the specified endpoints and objects when the parent names and the
object names match. -->
        <link condition="source.structName = target.parent.Name AND source.featureName =
target.Name"/>

    </rule>

</ruleSet>
```

The following code shows a sample rule set file that defines linking rules between custom resource objects and Informatica Platform resource endpoints:

```
<?xml version="1.0" encoding="UTF-8"?>
<ruleSet name="Link custom objects to Informatica Platform endpoints">

    <sourceResource name="MyInfaPlatformResource"/>
    <targetResource name="MyCustomResource"/>

    <rule name="Link custom columns to Informatica Platform relational or flat file data
object columns.">

        <sourceFilter>
            <endPoint connection="MyConnection" class="Column"/>
        </sourceFilter>

        <targetFilter>
            <element class="Library" >
                <element class="Table">
                    <element class="TableColumn"/>
                </element>
            </element>
        </targetFilter>

        <!--Link the specified endpoints and objects when the parent names and the
object names match. -->
        <link condition="source.structName = target.parent.Name AND source.featureName =
target.Name"/>

    </rule>

</ruleSet>
```

Rule Element Configuration for Non-Endpoints

Use the rule element to define a linking rule that Metadata Manager uses to link non-endpoint objects in the source resource to matching non-endpoint objects in the target resource.

The following table describes the attributes for the rule element:

Attribute	Description
name	Required. Name of the rule. The rule name must be unique in the rule set.
direction	Optional if linking to a business glossary resource. Required if linking between custom resources, between a custom and packaged resource, or between a custom and universal resource. Indicates whether the link originates from the source object or from the target object. Enter SourceToTarget, TargetToSource, or Auto. To link between custom resources, between a custom and packaged resource, or between a custom and universal resource, the direction must be SourceToTarget or TargetToSource. To link to terms in a business glossary, the direction must be Auto. If you do not use the direction attribute, Metadata Manager uses Auto by default.

A rule element must contain the following elements:

sourceFilter

Filters the list of possible objects in the source resource that you want to link. A sourceFilter element must contain an XML element named element.

The following table describes the attributes for the XML element named element:

Attribute	Description
class	Optional. Name of the class that the object belongs to. Use a pipe (|) to separate multiple class names. Either the class attribute or the type attribute is required.
type	Optional. Type of the class that the object belongs to. Use a pipe (|) to separate multiple class types. Either the class attribute or the type attribute is required.
condition	Optional. Expression that filters the list of possible objects to link.

An element can contain another element. The objects selected through the class, type, and condition attributes for this element must be immediate children of the objects selected through the containing element.

To specify a feature in a packaged or universal resource, the structure or the parent class must also be selected in the sourceFilter. For example, the following sourceFilter element includes the parent table as a containing element of an Oracle column:

```
<sourceFilter>
   <element class="Oracle Table">
      <element type="Column"/>
   </element>
</sourceFilter>
```

targetFilter

Filters the list of possible objects in the target resource that you want to link. A targetFilter element must contain an XML element named element.

Use the same syntax to configure elements in sourceFilters and targetFilters.

link

Defines the expression that specifies which filtered source and target objects Metadata Manager links. Contains a required condition attribute.

Expressions to Link Non-Endpoints

When you define a linking rule, you enter expressions in condition attributes. Expressions filter the list of possible objects to link and define which filtered objects Metadata Manager links.

You can include an expression in a condition attribute for the following elements in a rule set definition file or a rule set file configured for non-endpoint links:

- sourceFilter and targetFilter elements use the condition attribute as a selection expression. A selection expression filters the list of possible objects to link in the source or target resource.
- link elements use the condition attribute as a link expression. A link expression defines which filtered objects are linked. Metadata Manager links the objects that meet the condition defined in the link expression.

You can use the following operators in selection expressions and link expressions:

- Logical operators AND and OR.
- Comparison operators =, !=, IS NULL, and IS NOT NULL.
- Parentheses to group multiple conditions.

In a rule set definition file, you can use parameters to represent string literals in selection expressions and link expressions. Enter each parameter in the format `${<parameter_name>}`. For example:

```
<link condition="source.parent.Description=${Desc_string1}" />
```

Selection Expressions for Non-Endpoints

You can include class attributes in the selection expression. Use string literal values to define the values of attributes.

You can use any class attribute in a selection expression except for the following attributes:

- Object Class
- Location
- Source Creation Date
- Source Update Date
- MM Creation Date
- MM Update Date

If the attribute name contains spaces, enclose it in the XML character entity for quotation marks, `&quot;`. For example, if the attribute name is "Business Name," enter the name as follows:

```
&quot;Business Name&quot;
```

Literal values for attributes can use the following special characters:

```
! @ # $ % ^ { } | ?
```

Example

The following selection expression filters the list of possible target objects to objects in the class MyCustomClass that have a Description attribute with a value of "MyDescription":

```
<targetFilter>
   <element class="MyCustomClass" condition="Description='MyDescription'">
   </element>
</targetFilter>
```

Link Expressions for Non-Endpoints

A link expression for non-endpoints must refer to both the source and target resources. When you create link expressions for non-endpoints, you can use keywords and attributes that belong to the non-endpoint classes.

To refer to an object attribute in a link expression, use the following format:

```
<keyword>.<attribute>
```

The following table describes the keywords that you can use in link expressions:

Keyword	Description
source	Represents the object selected in the sourceFilter.
target	Represents the object selected in the targetFilter.
parent	Represents the parent of the object selected in the sourceFilter or targetFilter.

You can use the same class attributes that are valid for selection expressions.

If the attribute name contains spaces, enclose it in the XML character entity for quotation marks, `&quot;`. For example, if the attribute name is "Business Name," enter the name as follows:

```
&quot;Business Name&quot;
```

When you include multiple conditions in a link expression, Metadata Manager evaluates the conditions from left to right. For better performance during linking, write a condition that checks for parent attributes before a condition that checks for child attributes.

Example

The following link expression links non-endpoint objects when the parent names and object names match:

```
<link condition="source.parent.Name=target.parent.Name AND source.Name=target.Name"/>
```

Example

The following link expression links non-endpoint objects when the Label attribute for the source object or target object has a value of "MyLabel":

```
<link condition="source.Label='MyLabel' OR target.Label='MyLabel'"/>
```

Sample Rule Set Definition Files to Link Non-Endpoints

The following code shows a sample rule set definition file that defines two linking rules between a pair of custom models:

```
<?xml version="1.0" encoding="UTF-8"?>
<ruleSetDefinition name="MAL_NEP_def_custom_to_custom_string_link">

    <sourceModel name="Custom_Dataset_MAL"/>
    <targetModel name="Custom_Dataset_Rel_MAL"/>
```

```
    <param name="Desc_string1" description="The description string value"
type="string" />
    <param name="Desc_string2" description="The description string value"
type="string" />

    <rule name="SC_to_SC_Rel_RBL_NEP_12_down" direction="TargetToSource">

        <sourceFilter>
            <element class="Class_lin">
                <element class="SubClass_lin"/>
            </element>
        </sourceFilter>

        <targetFilter>
            <element class="Class_lin_Rel">
                <element class="SubClass_lin_Rel"/>
            </element>
        </targetFilter>

        <link condition="source.Name=target.Description AND source.parent.Description=$
{Desc_string1} AND source.parent.Name=target.parent.Description AND target.parent.Label=$
{Desc_string1}"/>

    </rule>

    <rule name="SC_to_SC_Rel_RBL_NEP_12_up" direction="SourceToTarget">

        <sourceFilter>
            <element class="Class_lin">
                <element class="SubClass_lin"/>
            </element>
        </sourceFilter>

        <targetFilter>
            <element class="Class_lin_Rel">
                <element class="SubClass_lin_Rel"/>
            </element>
        </targetFilter>

        <link condition="source.Name=target.Description AND source.parent.Description=$
{Desc_string2} AND source.parent.Name=target.parent.Description AND target.parent.Label=$
{Desc_string2}"/>

    </rule>

</ruleSetDefinition>
```

The following code shows a sample of a rule set parameter file that defines the source and target resources and the parameter values for the rule set definition file:

```
<?xml version="1.0" encoding="UTF-8"?>
<ruleSetParams definition="MAL_NEP_def_custom_to_custom_string_link"
name="param_custom_string_link_res1" description="testing string parameter">

    <sourceResource name="Custom_Test_MBL" />
    <targetResource name="Custom_Test_MBL_Rel" />

    <param name="Desc_string1" value="grp3" />
    <param name="Desc_string2" value="grp4" />

</ruleSetParams>
```

Sample Rule Set Files to Link Non-Endpoints

The following code shows a sample rule set file that defines two linking rules between custom resource objects and database management resource objects:

```
<?xml version="1.0" encoding="UTF-8"?>
<ruleSet name="Link custom objects to Oracle objects">

    <sourceResource name="MyCustomResource"/>
    <targetResource name="MyOracleResource"/>

    <rule name="Link custom table or view column to Oracle column"
direction="TargetToSource">

        <sourceFilter>
            <element class="Library" >
                <element class="Table|Synonym|View">
                    <element class="TableColumn|ViewColumn"/>
                </element>
            </element>
        </sourceFilter>

        <targetFilter>
            <element class="Oracle Schema" >
                <element type="Table|View|Synonym">
                    <element type="Column"/>
                </element>
            </element>
        </targetFilter>

        <!--Link the specified objects when two levels of the parent names and the
object names match. -->
        <link condition="source.parent.parent.Name=target.parent.parent.Name AND
source.parent.Name=target.parent.Name AND source.Name=target.Name"/>

    </rule>

    <rule name="Link custom procedure to Oracle procedure" direction="TargetToSource">

        <sourceFilter>
            <element class="Library" >
                <element class="Procedure">
                </element>
            </element>
        </sourceFilter>

        <targetFilter>
            <element class="Oracle Schema" >
                <element type="Procedure">
                </element>
            </element>
        </targetFilter>

        <!--Link the specified objects when the parent names and the object names match.
-->
        <link condition="source.parent.Name=target.parent.Name AND
source.Name=target.Name"/>

    </rule>

</ruleSet>
```

The following code shows a sample rule set file that defines a linking rule between custom resource objects and business glossary terms:

```
<?xml version="1.0" encoding="UTF-8"?>
<ruleSet name="Link custom objects to business terms">

    <sourceResource name="MyCustomResource"/>
    <targetResource name="MyBusinessGlossary"/>
```

```
    <rule name="Link custom objects to business terms">

        <sourceFilter>
            <element class="Library" >
                <element class="Table">
                    <element class="TableColumn"/>
                </element>
            </element>
        </sourceFilter>

        <targetFilter>
            <element class="Business Term" />
        </targetFilter>

        <!-- Link columns and business terms when the column and business term names
match. -->
        <link condition="source.Name = target.Name" />

    </rule>

</ruleSet>
```

Linking Rules Files Schema Definitions

A linking rules file must conform to the structure of the XML schema definition (XSD). If the linking rules file does not conform to the schema definition, Metadata Manager cannot create or update the rule set definition in the Metadata Manager repository.

Each type of linking rules file has its own schema definition.

Rule Set Definition File Schema Definition

A rule set definition file must conform to the structure of the rule set definition file XSD.

The following example shows the rule set definition file XML schema definition:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

    <xs:element name="ruleSetDefinition">
        <xs:annotation>
            <xs:documentation>Container of rules. This container should have a globally
unique name</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:sequence>
                <!--Identifiers for selecting the source and target resource. Later it
can be used to just select types and provide the actual resources at application time -->
                <xs:element name="sourceModel">
                    <xs:complexType>
                        <xs:attribute name="name" type="xs:string"></xs:attribute>
                    </xs:complexType>
                </xs:element>
                <xs:element name="targetModel">
                    <xs:complexType>
                        <xs:attribute name="name" type="xs:string"></xs:attribute>
                    </xs:complexType>
                </xs:element>
                <!--A rule set must have at least one rule -->
                <xs:element name="param" type="paramDefType" maxOccurs="unbounded"
minOccurs="0"></xs:element>
                <xs:element maxOccurs="unbounded" name="rule" type="rule" />
            </xs:sequence>
```

```
            <xs:attribute name="name" type="xs:string" use="required" />
            <xs:attribute fixed="1.0" name="version" type="xs:string" />
            <!--Mandatory attributes for rule set-->
            <xs:attribute name="description" type="xs:string" use="optional"></
xs:attribute>
        </xs:complexType>
    </xs:element>

    <xs:complexType name="rule">
        <xs:annotation>
            <xs:documentation>A rule in a ruleset. A rule is uniquely identified by its
name within a rule set</xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:element name="sourceFilter" type="sourceElementSelectorType"/>
            <xs:element name="targetFilter" type="targetElementSelectorType"/>
            <xs:element name="link" type="linkType"/>
        </xs:sequence>
        <xs:attribute name="name" type="xs:string" use="required"/>
        <xs:attribute name="direction" type="directionType"/>
        <!--A rule must have source, target, link in the sequence-->
    </xs:complexType>

    <xs:simpleType name="directionType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="SourceToTarget"/>
            <xs:enumeration value="TargetToSource"/>
            <xs:enumeration value="Auto"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:complexType name="sourceElementSelectorType">
        <xs:choice>
            <xs:element name="element" type="elementFilterType"/>
            <xs:element name="endPoint" type="endPointType"/>
        </xs:choice>
    </xs:complexType>

    <xs:complexType name="targetElementSelectorType">
        <xs:choice>
            <xs:element name="element" type="elementFilterType"/>
        </xs:choice>
    </xs:complexType>

    <!-- Right now we are not creating simple types that can define a pattern for
specifying identifiers (like class name, feature name, type etc). If we have leisure
time later :) we can do that -->

    <xs:complexType name="elementFilterType">
        <xs:choice>
            <xs:element maxOccurs="unbounded" minOccurs="0" name="element"
type="elementFilterType"/>
        </xs:choice>
        <xs:attribute name="class" type="xs:string"/>
        <xs:attribute name="type" type="xs:string"/>
        <xs:attribute name="condition" type="xs:string"/>
    </xs:complexType>

    <xs:complexType name="endPointType">
        <xs:attribute name="class" type="xs:string"/>
        <xs:attribute name="type" type="xs:string"/>
        <xs:attribute name="connection" type="xs:string"/>
        <xs:attribute name="package" type="xs:string"/>
    </xs:complexType>

    <xs:complexType name="linkType">
        <xs:attribute name="condition" type="xs:string" use="required"/>
    </xs:complexType>

    <xs:complexType name="paramDefType">
        <xs:attribute name="name" type="xs:string"></xs:attribute>
```

```
        <xs:attribute name="description" type="xs:string"></xs:attribute>
        <xs:attribute name="defaultValue" type="xs:string"></xs:attribute>
        <xs:attribute name="type" default="string">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="connection"></xs:enumeration>
                    <xs:enumeration value="package"></xs:enumeration>
                    <xs:enumeration value="string"></xs:enumeration>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
    </xs:complexType>

</xs:schema>
```

Rule Set Parameter File Schema Definition

A rule set parameter file must conform to the structure of the rule set parameter file XSD.

The following example shows the rule set parameter file XML schema definition:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

    <xs:element name="ruleSetParams">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="sourceResource" type="resourceType" maxOccurs="1"
minOccurs="1"></xs:element>
                <xs:element name="targetResource" type="resourceType" maxOccurs="1"
minOccurs="1"></xs:element>
                <xs:element name="param" maxOccurs="unbounded" minOccurs="0">
                    <xs:complexType>
                        <xs:attribute name="name" type="xs:string" use="required"></
xs:attribute>
                        <xs:attribute name="value" type="xs:string" use="required"></
xs:attribute>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
            <xs:attribute name="definition" type="xs:string" use="required"></
xs:attribute>
            <xs:attribute name="name" type="xs:string" use="required"></xs:attribute>
            <xs:attribute name="description" type="xs:string" use="optional"></
xs:attribute>
        </xs:complexType>
    </xs:element>

    <xs:complexType name="resourceType">
        <xs:attribute name="name" type="xs:string" use="required"></xs:attribute>
    </xs:complexType>

</xs:schema>
```

Rule Set File Schema Definition

A rule set file must conform to the structure of the rule set file XSD.

The following example shows the rule set file XML schema definition:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

    <xs:element name="ruleSet">
        <xs:annotation>
            <xs:documentation>Container of rules. This container should have a globally
unique name</xs:documentation>
        </xs:annotation>
```

```
        <xs:complexType>
            <xs:sequence>
                <!--Identifiers for selecting the source and target resource. Later it
can be used to just select types and provide the actual resources at application time -->
                <xs:element name="sourceResource" type="resourceType" />
                <xs:element name="targetResource" type="resourceType" />
                <!--A rule set must have at least one rule -->
                <xs:element maxOccurs="unbounded" name="rule" type="rule" />
            </xs:sequence>
            <xs:attribute name="name" type="xs:string" use="required" />
            <xs:attribute fixed="1.0" name="version" type="xs:string" />
            <!--Mandatory attributes for rule set-->
            <xs:attribute name="description" type="xs:string" use="optional"></
xs:attribute>
        </xs:complexType>
    </xs:element>

    <xs:complexType name="resourceType">
        <xs:attribute name="name" use="required"/>
        <!--Mandatory attribute in this release. Later we will allow type also which
will make this attribute non-mandatory. Any one of them would be sufficient-->
    </xs:complexType>

    <xs:complexType name="rule">
        <xs:annotation>
            <xs:documentation>A rule in a ruleset. A rule is uniquely identified by its
name within a rule set</xs:documentation>
        </xs:annotation>
            <xs:sequence>
                <xs:element name="sourceFilter" type="sourceElementSelectorType"/>
                <xs:element name="targetFilter" type="targetElementSelectorType"/>
                <xs:element name="link" type="linkType"/>
            </xs:sequence>
            <xs:attribute name="name" type="xs:string" use="required"/>
            <xs:attribute name="direction" type="directionType"/>
            <!--A rule must have source, target, link in the sequence-->
    </xs:complexType>

    <xs:simpleType name="directionType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="SourceToTarget"/>
            <xs:enumeration value="TargetToSource"/>
            <xs:enumeration value="Auto"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:complexType name="sourceElementSelectorType">
        <xs:choice>
            <xs:element name="element" type="elementFilterType"/>
            <xs:element name="endPoint" type="endPointType"/>
        </xs:choice>
    </xs:complexType>

    <xs:complexType name="targetElementSelectorType">
        <xs:choice>
            <xs:element name="element" type="elementFilterType"/>
        </xs:choice>
    </xs:complexType>

    <!-- Right now we are not creating simple types that can define a pattern for
specifying identifiers (like class name, feature name, type etc). If we have leisure
time later :) we can do that -->

    <xs:complexType name="elementFilterType">
        <xs:choice>
            <xs:element maxOccurs="unbounded" minOccurs="0" name="element"
type="elementFilterType"/>
        </xs:choice>
        <xs:attribute name="class" type="xs:string"/>
        <xs:attribute name="type" type="xs:string"/>
        <xs:attribute name="condition" type="xs:string"/>
```

```
    </xs:complexType>

    <xs:complexType name="endPointType">
        <xs:attribute name="class" type="xs:string"/>
        <xs:attribute name="type" type="xs:string"/>
        <xs:attribute name="connection" type="xs:string"/>
        <xs:attribute name="package" type="xs:string"/>
    </xs:complexType>

    <xs:complexType name="linkType">
        <xs:attribute name="condition" type="xs:string" use="required"/>
    </xs:complexType>

</xs:schema>
```

CHAPTER 14

Custom Resource Migration

This chapter includes the following topics:

Custom Resource Migration Overview

You can migrate custom resources between Metadata Manager instances. Use Metadata Manager and the Custom Metadata Configurator to migrate the models, custom resource templates, and linking rule sets associated with a custom resource from one Metadata Manager repository to another. Migrate custom resources so that you do not have to re-create them in the target environment.

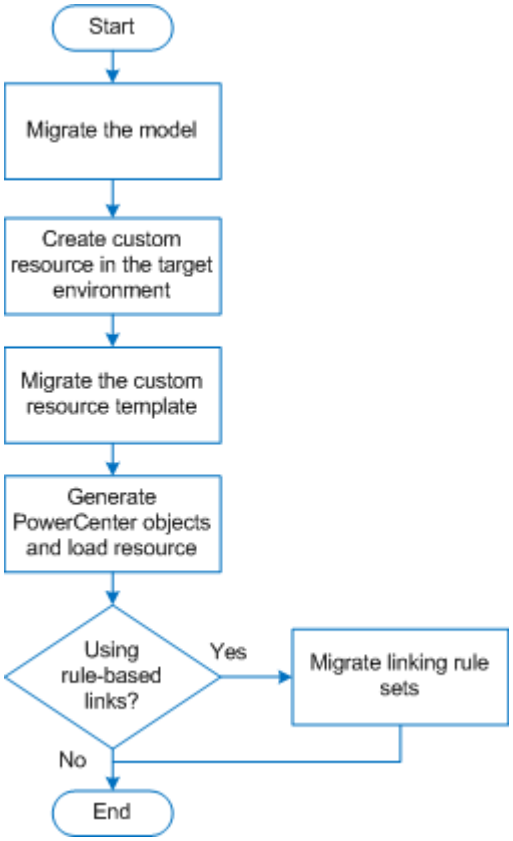
You might migrate custom resources when you move from a development to a production environment. Migrate the resources and custom resource templates, load the resources in the production environment, and then migrate the linking rule sets.

Note: The information in this chapter applies to migrating custom resources within the current Metadata Manager version. For information about migrating packaged or universal resources, see the *Metadata Manager Administrator Guide*.

Custom Resource Migration Steps

To migrate a custom resource between Metadata Manager instances of the same version, you migrate the models, custom resource templates, and linking rule sets to the target environment. To migrate objects, you export them from the source repository and import them into the target repository.

The following image shows the process to migrate a custom resource that you created with the Custom Metadata Configurator:



1. Migrate the model:
   a. Export the model from the source repository. On the **Model** tab, select **Actions** > **Export Models**. If the model has associated rule set definitions, include them in the export file.
   b. Import the model into the target repository. On the **Model** tab, select **Actions** > **Import Models**.
2. Create the custom resource in the target environment.
3. Migrate the custom resource template:
   a. Export the custom resource template from the source repository. In the Custom Metadata Configurator, log in to the source Metadata Manager repository, and click **Export**.
   b. Import the custom resource template into the target repository. In the Custom Metadata Configurator, log in to the target Metadata Manager repository, and click **Import**.
4. In the target environment, generate PowerCenter objects in the Custom Metadata Configurator, and then configure and load the resource.

5. If the resource uses rule based links, migrate the linking rule sets:
   a. In the source environment, run the mmcmd exportLinkRuleSets command to export all rule sets associated with the resource.
   b. In the target environment, run the mmcmd importLinkRuleSets command import the rule sets into the target repository.

Step 1. Migrate the Model

To migrate a model, you export the model from the source Metadata Manager repository to an XML file. You then import the XML file into the target Metadata Manager repository.

The model export XML file contains all the classes, attributes, and relationships for the model. When you import a model, Metadata Manager analyzes the contents of the XML file and compares it to the existing models. If the model does not exist, Metadata Manager creates the model. If the model exists, Metadata Manager imports the new or changed classes and relationships.

For example, you export a custom model from the development environment and then import it into the production environment. If you add a class to the custom model in the development environment, export it, and re-import it into the production environment, Metadata Manager imports the class that you added.

Exporting a Model

Export models to a model export XML file. You can include rule set definitions in the export file.

You can export packaged, universal, or custom models. You can export one packaged or universal model to an export file. You can also export one or multiple custom models to an export file. However, you cannot export a combination of packaged and custom models to one export file.

Note: You cannot export the Business Glossary model from Metadata Manager. To export business glossary assets and templates, use the Analyst tool.

1. On the **Model** tab, click **Actions** > **Export Models**.

   The **Export Model** window appears.
2. Select the models to export and add them to the **Selected models** list.

   Note: You can export multiple models to one export file only if all of the selected models are custom models.
3. To include the rule set definitions associated with the selected models in the export file, select **Include rule set definitions**.

   If you select this option, Metadata Manager creates a .zip file that contains the model export XML file plus an XML file for each rule set definition. If you do not select this option, Metadata Manager creates an XML file for the selected models.
4. Click **Export**.

   The options to save the XML file vary based on the browser.

Importing a Model

Use the Import Model wizard to import models into a Metadata Manager repository. When you import a model, you select the XML or .zip file that contains models and select the models you want to import.

The Import Model wizard analyzes and validates the file that you select. If the model does not exist in the Metadata Manager repository, Metadata Manager imports the entire model. If the model exists, Metadata Manager imports the new and changed classes and relationships. If the model in the file matches the model in the Metadata Manager repository, Metadata Manager does not import the model.

Note: You cannot import a Business Glossary model in Metadata Manager. To import business glossary assets and templates, use the Analyst tool.

1. On the **Model** tab, click **Actions** > **Import Models**.

   The **Import Model** window appears.
2. Select the XML or .zip file that contains the models that you want to import.
3. Click **Next**.
4. Select the models that you want to import, and click **Next**.

   The wizard analyzes the file and validates it against the models in the Metadata Manager repository. The wizard lists the classes and relationships that Metadata Manager will create and update, and the classes and relationships that are not affected by the import process.
5. Click **Import**.

Step 2. Create the Resource in the Target Environment

After you migrate the model, you must create the custom resource in the target environment. Create the resource so that you can migrate the custom resource template to the target environment.

Create the resource on the **Load** tab in the target Metadata Manager instance.

Step 3. Migrate Custom the Resource Template

Create a custom resource template in the Custom Metadata Configurator before you load custom metadata into the metadata catalog. When you migrate a custom resource template, you export the template from the source Metadata Manager repository and import it into the target Metadata Manager repository. Use the Custom Metadata Configurator to export and import custom resource templates.

When you export a custom resource template, the Custom Metadata Configurator exports it to a binary custom template file with the .ctf extension. Use the exported custom template file to import the custom resource template into the target repository. You can export or import one template at a time.

To troubleshoot errors and get more information about the export and import process, view the customwizard.log file. The customwizard.log file is located in the following directory:

```
<PowerCenter Client installation directory>\client\custom-configurator
```

Exporting a Custom Resource Template

Export a custom resource template through the Custom Metadata Configurator.

1. In the Custom Metadata Configurator, log in to the Metadata Manager repository that contains the custom resource template that you want to export.
2. Click **Export**.

   The **Export Custom Template** dialog box appears.
3. Select the custom resource for the template that you want to export.
4. Enter the path and name of the custom template file to which you want to export, or click the **Browse (...)** button to select the file.
5. Click **Export**.

The Custom Metadata Configurator exports the custom resource template to a .ctf file. Import the custom resource template into the target repository from the file.

Importing a Custom Resource Template

When you import a custom resource template, Metadata Manager creates a template in the Metadata Manager repository for a custom resource.

Before you import a custom resource template into the repository, verify that the custom model and custom resource exist in the Metadata Manager repository. You cannot import a custom resource template if the model and resource do not exist.

If the template contains relationships to other resources, you must also configure the resources for the relationships before you import the template. If the resources do not exist in Metadata Manager, you must load the resources before you import the template.

1. In the Custom Metadata Configurator, log in to the Metadata Manager repository where you created the custom resource.
2. Click **Import**.

   The **Import Custom Resource Template** dialog box appears.
3. Select the custom resource for which you want to import the template.
4. Click the **Browse (...)** button to select the custom template file that you want to import.
5. Click **Import**.

   The Custom Metadata Configurator imports the template from the custom template file.
6. If the custom resource template contains relationships to other resources, configure a target resource for the relationship for each source resource in the custom template file.

After you import the template, you must generate the PowerCenter objects that Metadata Manager requires to load the custom metadata.

Rules and Guidelines for Migrating Custom Resource Templates

Use the following rules and guidelines when you migrate custom resource templates:

- You can migrate custom resource templates between instances of the same version of Metadata Manager only.
- You must create a custom resource before you can import a custom resource template.

- When you import a custom resource template and configure the source and target resources for relationships, the names of the source and target resources do not need to match.
- You can import custom resource templates for custom resources in Metadata Manager only if the custom resource does not have a template configured.
- If the template you want to import exists in the Metadata Manager repository, the import process fails.
- The models for the source and target resources and the model names must be the same.

Step 4. Generate PowerCenter Objects and Configure the Resource

If you load metadata for a custom resource using a metadata source file, you must generate the PowerCenter objects, configure the metadata source file, and then load the custom resource in the target Metadata Manager environment.

To configure and load the custom resource in the target Metadata Manager environment, complete the following tasks

1. In the Custom Metadata Configurator, generate the PowerCenter objects.

   The PowerCenter objects include the mappings, sessions, and workflows that extract metadata from the metadata files and load it into the Metadata Manager repository.
2. On the **Load** tab in Metadata Manager, configure the custom resource.

   Configure the metadata source files for the custom resource.
3. On the **Load** tab in Metadata Manager, load the custom resource.

Step 5. Migrate Linking Rule Sets

If the custom resource uses linking rule sets, migrate the linking rule sets from the source Metadata Manager repository to the target Metadata Manager repository. To migrate linking rule sets, export the rule sets from the source repository to XML files and import the XML files into the target repository.

If the custom resource does not use linking rule sets, skip this step.

Use the following mmcmd commands to export and import linking rule sets:

- To export all rule sets associated with a custom resource from the source repository, run the mmcmd exportLinkRuleSets command.
- To import all rule sets from XML files in the specified path into the target repository, run the mmcmd importLinkRuleSets command.

When you import a rule set, Metadata Manager does not create the links between the resources. To use the rules to link metadata elements between resources, you must the load the resources or create the links in the **Resource Link Administration** window.

APPENDIX A

Resource Configuration File

This appendix includes the following topics:

Resource Configuration File Overview

The resource configuration file determines how Metadata Manager loads metadata into a custom resource.

Note: The information in this appendix applies to resource configurations for custom resources. For information about resource configurations for packaged or universal resources, see the *Metadata Manager Administrator Guide*.

The resource configuration file contains the following information:

- Mapping rules and mapping rule sets that apply to each metadata source file.
- Properties about each metadata source file, for example, the path to the metadata source file and whether the file has headers.
- Parameters that determine whether to use an extraction command file to generate and populate metadata source files each time the metadata loads.
- A parameter that determines whether to retrieve and use the latest metadata source files.

When you load custom metadata through a load template XConnect, you do not need to create or change the resource configuration file. However, if you want to create a resource configuration file manually, you can export the current resource configuration on the **Load** tab. Update the file with a text editor. After you update the file, import the modified resource configuration file on the **Load** tab.

After you import the resource configuration file and select the custom resource in Metadata Manager, the following tabs display information about the custom resource:

Properties

Displays the resource type, name, and description.

Input Files

Displays all metadata source files from which metadata is extracted when you load the metadata. This tab also displays the metadata source file information such as the file encoding, the column separator, and the number of rows to skip.

Mapping Rules

Displays the mapping rules for each metadata source file.

Enumerated Links

Displays the enumerated links files associated with the resource. This tab also displays enumerated links file information such as whether the file has a header and whether to use the latest source files.

Linking Rules

Displays links between source and target resources if you created rule sets.

Schedule

Lists the schedule, if any, that is attached to the resource.

Resource Configuration File Components

The resource configuration file consists of multiple XML elements. Each element describes a component of the resource configuration file. In addition, the resource configuration file can also contain run-time parameters.

The resource configuration file contains the following components:

Elements

Predefined XML elements that provide information about the metadata source files and how to apply mapping rules and mapping rule sets to each file.

Parameters

Parameters that specify run-time information about the metadata source files. For example, you can configure parameters to specify whether to get the latest metadata source files from a local machine.

Resource Configuration File Elements

The resource configuration file consists of different XML elements that provide information about the metadata source and enumerated links files.

The load template XML contains the following elements:

resource

The following table describes the attributes for the resource element:

Attributes	Description
name	Required. Name of the resource.
resourceType	Required. Type of resource. Same as the name of the model.

parameter

To add multiple parameters, you can add multiple parameter elements within the parameters element. The following table describes the attribute for the parameter element:

Attribute	Description
name	Name of the parameter.

ruleBinding

The ruleBinding element does not contain any attributes.

CSVFile

The following table describes the attributes for the CSVFile element:

Attributes	Description
columnSeparator	Character that separates entries in the file. Default is a comma (,).
encoding	Code page for the metadata source file. Default is UTF-8.
escapeCharacter	Escape character for values in the metadata source file. Default is the backslash character (\).
hasColumnHeaders	Indicates whether the file has a header row. Valid values are 'yes' and 'no.' Default is 'yes.'
path	Absolute path and name of the metadata source file.
rowSeparator	Character that separates rows in the file.
rowsToSkip	Number of rows to skip in the metadata source file. Skip preliminary rows that contain header information or no information. You must enter an integer equal to or greater than 0. Default is 0.
textQualifier	Character used to enclose text that should be treated as one entry. Use a text qualifier to disregard the delimiter character within text. Default is the double quotes character (“).
useLatestSourceFiles	Indicates whether to use the latest metadata source files before running the custom XConnect. Metadata Manager retrieves the latest metadata source files from the directory specified in the resource configuration file. Set this attribute to 'true' if you use an extraction command file. If you set the attribute to 'false,' Metadata Manager uses the existing metadata source files stored in Metadata Manager. Valid values are 'true' and 'false.' Default is 'false.'

EnumeratedLinks

The EnumeratedLinks element does not contain any attributes.

CSVEnumeratedLinks

The following table describes the attributes for the CSVEnumeratedLinks element:

Attributes	Description
path	Absolute path and name of the enumerated links file.
useLatestSourceFiles	Indicates whether to use the latest metadata source files before running the custom XConnect. Metadata Manager retrieves the latest metadata source files from the directory specified in the resource configuration file. Set this attribute to 'true' if you use an extraction command file. If you set the attribute to 'false,' Metadata Manager uses the existing metadata source files stored in Metadata Manager. Valid values are 'true' and 'false.' Default is 'false.'
hasColumnHeaders	Indicates whether the file has a header row. Valid values are 'yes' and 'no.' Default is 'yes.'

Resource Configuration File Parameters

You can configure parameters for the resource configuration file.

You can configure the following parameters:

Parameter	Description
encoding	Code page for the file.
useLatestSourceFiles	Determines whether to use the existing metadata source files stored in Metadata Manager or to retrieve the latest files from the specified file path. Valid values are 'true' and 'false.' Default value is 'false.'
preprocessCommandFile	The path and name of the extraction command file. If you specify an extraction command file, Metadata Manager generates the metadata source files each time you load the metadata.

Resource Configuration File Sample

The sample resource configuration file shows how to specify parameters, specify metadata source file properties, and apply mapping rules to metadata source files.

You can use this sample to help you create your own resource configuration file:

```
<?xml version="1.0" encoding="UTF-8"?>
<resource name="AS400" resourcetype="AS400">
    <parameters>
        <parameter name="useLatestSourceFiles">true</parameter>
        <parameter name="preprocessCommandFile">C:\Users\mcastro\Desktop\9.5\Writing
\MetadataManager\SourceFiles\cop.bat</parameter>
    </parameters>
    <RuleBinding>
        <CSVFile textQualifier="&quot;" escapeCharacter="\" columnSeparator=","
encoding="UTF-8" path="C:\Users\mcastro\Desktop\9.5\Writing\MetadataManager\SourceFiles
\Table.csv" useLatestSourceFiles="false" hasColumnHeaders="true">
            <Rule name="TableRule"/>
        </CSVFile>
```

```
        <CSVFile textQualifier="&quot;" escapeCharacter="\" columnSeparator="," 
encoding="UTF-8" path="C:\Users\mcastro\Desktop\9.5\Writing\MetadataManager\SourceFiles
\TableColumn.csv" useLatestSourceFiles="false" hasColumnHeaders="true">
            <Rule name="TableColumnRule"/>
        </CSVFile>
        <CSVFile textQualifier="&quot;" escapeCharacter="\" columnSeparator="," 
encoding="UTF-8" path="C:\Users\mcastro\Desktop\9.5\Writing\MetadataManager\SourceFiles
\View.csv" useLatestSourceFiles="false" hasColumnHeaders="true">
            <Rule name="ViewRule"/>
        </CSVFile>    <CSVFile textQualifier="&quot;" escapeCharacter="\" 
columnSeparator="," encoding="UTF-8" path="C:\Users\mcastro\Desktop\9.5\Writing
\MetadataManager\SourceFiles\ViewColumn.csv" useLatestSourceFiles="false" 
hasColumnHeaders="true">
            <Rule name="ViewColumnRule"/>
        </CSVFile>
    </RuleBinding>
</resource>
```

Exporting a Resource Configuration

On the **Load** tab, you can export a resource configuration to a resource configuration file. When you export a resource configuration, you can include rule sets and parameter definitions.

1. On the **Load** tab, select a resource.
2. Click **Actions** > **Export Resource Configuration**.
3. Metadata Manager prompts you to include rule sets and parameter definitions in the export file.

   Select one of the following options:

Option	Description
Yes	Export all source files associated with the resource, including rule sets and parameter definitions.
No	Export the resource configuration only.

4. Metadata Manager prompts you to include the resource password in the export file.

   Custom resources do not include passwords, so select **No**.

   The options to save the export file vary based on the browser.

If you include rule sets and parameter definitions in the export file, Metadata Manager creates a file with the .rcz extension in the location you specify. If you export the resource configuration only, Metadata Manager creates a file with the .rcf extension in the location you specify.

Importing a Resource Configuration

On the **Load** tab, you can import or update a resource configuration. If the resource does not exist, Metadata Manager creates it. If the resource already exists, Metadata Manager updates the resource configuration.

1. On the **Load** tab, click **Actions** > **Import Resource Configuration**.

   The **Import Resource Configuration** dialog box appears.

2. Select the import file.

   If you import a resource configuration file with the .rcf extension, Metadata Manager imports the resource configuration. If you import a resource configuration file with the .rcz extension, Metadata Manager imports the resource configuration plus all rule sets, parameter definitions, and enumerated links associated with the resource.

3. Custom resources do not include passwords or secure JDBC parameters, so leave the **Password** and **Secure JDBC Parameters** properties empty.

4. Click **OK**.

   If the resource does not exist, Metadata Manager creates it. If the resource already exists, Metadata Manager updates the resource configuration.

INDEX

D

E

G

I

L

M