Informatica® Development Platform
10.2

Informatica Connector Toolkit Developer Guide

Informatica Development Platform Informatica Connector Toolkit Developer Guide
10.2
September 2017

Publication Date: 2019-01-22

Table of Contents

Preface

The *Informatica Connector Toolkit Developer Guide* is written for independent software vendors, consulting organizations, and developers who want to develop adapters for the Informatica platform.

This guide assumes you have a working knowledge of Informatica services and are familiar with data sources, ODBC, JDBC, and APIs.

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.

CHAPTER 1

Introduction to the Informatica Connector Toolkit

This chapter includes the following topics:

Informatica Connector Toolkit Overview

Use the Informatica Connector Toolkit to build an adapter that provides connectivity between a data source and PowerCenter®, Big Data Management®, or Informatica Cloud®.

Although Informatica supports generic ODBC connectivity and allows access to any data source that has a standards-compliant ODBC driver, building an adapter by using the Informatica Connector Toolkit offers several advantages. In cases where no ODBC driver is available, building an adapter by using the Informatica Connector Toolkit might be the only solution.

When you use the Informatica Connector Toolkit to create an adapter, you can build functionality related to the data source. You can preserve data type integrity and metadata lineage of the data source when you retain the type system of the data source and perform optimal data type conversions.

The Informatica Connector Toolkit consists of libraries, plug-ins, and sample code to assist you in developing adapters for PowerCenter, Big Data Management, or Informatica Cloud. You can use the Informatica Connector perspective in the Eclipse IDE to quickly develop an adapter in an Eclipse environment.

The Informatica Connector Toolkit simplifies the following processes:

- Development. You can use the wizards in the Informatica Connector perspective to rapidly develop an adapter. The wizards simplifies the use of internal components and dependencies when you develop an adapter.
- Testing. After you define the adapter components, you can test the connection, metadata, and run-time components of the adapter.
- Deployment. You can deploy the adapter on local Informatica services and clients and register the adapter on local Informatica services. Alternatively, you can publish the adapter and manually deploy the adapter later.

The Informatica Connector Toolkit API is written in a combination of C/C++ and Java. The connection definition and metadata definition are available in Java. The run-time interfaces are available in C/C++ and Java.

Note: Run-time interfaces are available in C/C++ when you develop adapters for Big Data Management. Run-time interfaces are available only in Java when you develop connectors for Informatica Cloud and PowerCenter.

Supported Features for PowerCenter, Big Data Management, and Informatica Cloud

The following table lists the features that Informatica Connector Toolkit supports for PowerCenter, Big Data Management, and Informatica Cloud:

Adapter Component	Features	Big Data Management	Informatica Cloud	PowerCenter
Connection	Connection configuration	Yes	Yes	Yes
Metadata	Multiple native metadata objects	Yes	No	No
	Metadata write capability	No	Yes	No
	Metadata import configuration	Yes	No	No
	Partitioning capability	Yes	Yes **Note:** You can only implement static partitioning.	Yes **Note:** You can only implement static partitioning.
	Join operation	Yes	Yes	No
	Filter operation	Yes	Yes	Yes
	Sort operation	Yes	No	No
	Select operation	Yes	No	No
	Lookup	Yes	No	No
Runtime	C Runtime	Yes	No	No
	Java Runtime	Yes	Yes	Yes

Informatica Connector Perspective

The views in the Informatica Connector perspective enable you to develop adapters for the Informatica platform and connectors for Informatica Cloud.

After you click the **Create New Connector** icon in the Eclipse Workbench toolbar, you can choose to switch to the Informatica Connector perspective. You can also open the Informatica Connector perspective from the **Window** menu in the Eclipse IDE.

The Informatica Connector perspective consists of the Connector Navigator view and Connector Progress view in the Eclipse Workbench window, and icons in the Eclipse toolbar. You can use the Connector Navigator and Connector Progress views to define, edit, or test the adapter components and view the progress of the adapter project. Use the **Create New Connector** icon in the Toolbar to create an adapter project. You can use the **View or Create Messages** icon in the Eclipse toolbar to view or create messages for the adapter components.

Connector Navigator

You can use the Connector Navigator view to add, define, edit, or test adapter components, such as connection, metadata, and run time.

After you create an Informatica Connector project, you can right-click folders in the Connector Navigator view to define, edit, or test adapter components. You can also right-click the adapter source files and edit them in the Eclipse Workbench editor or any other editor.

Connector Progress

You can use the Connector Progress view to view the completeness of the connector project and to define or edit the connector project components in each phase.

The Connector Progress view consists of the following phases:

Connectivity

You can define, edit, or test the connection component of an adapter in the Connectivity phase.

Metadata

You can define and edit data types and native metadata object components of an adapter in the Metadata phase. You can also define common metadata for Informatica Cloud connectors in the Metadata phase. You can test the adapter metadata components in the Metadata phase.

Runtime

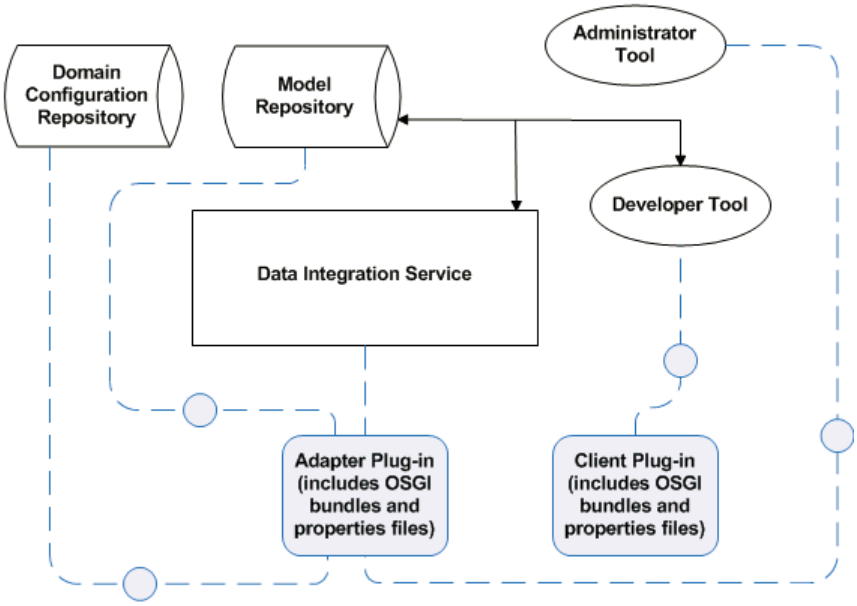
You can define, edit, and test the adapter run-time components in the Runtime phase.

Export/Publish

You can publish the adapter to a location or deploy the adapter on local Informatica services and client directories in the Export/Publish phase.

Architecture

The following image shows how an adapter works with Big Data Management services and tools:



The following Big Data Management services and tools work with the adapter plug-in and client plug-in components of the adapter:

Data Integration Service

Uses the adapter plug-in at run time to read data from or write data to the data source.

Developer tool

Uses the client plug-in to create and update connections, import metadata, and perform type system conversions.

Administrator tool

Uses the adapter plug-in to create and update the connection management properties.

Domain repository

Stores the connection objects that an end user creates using the adapter plug-in.

Model Repository Service

Stores the data source metadata and data source operation that an end user creates using the adapter plug-in.

CHAPTER 2

Installing and Upgrading the Informatica Connector Toolkit

This chapter includes the following topics:

Installing the Informatica Connector Toolkit Overview

Install the Informatica Connector Toolkit and set up the Informatica Connector Toolkit Eclipse plug-in. To set up the Informatica Connector Toolkit Eclipse plug-in, you install the required software on the machine where you plan to develop the adapter.

The Informatica Connector Toolkit is part of the Informatica Development Platform (IDP). You can use the Informatica Connector Toolkit installer to install the Informatica Connector Toolkit on the machine in which you plan to develop an adapter.

You can get the Informatica Connector Toolkit installer from the following sources:

- Informatica Development Platform installation DVD.
  Informatica ships the Informatica SDKs on a separate DVD. Run the Informatica Connector Toolkit installer to install the Informatica Connector Toolkit on the machine where you plan to do your development.
- Informatica electronic software download site.
  When you purchase an Informatica product and choose to download the software, you receive a site link, user ID, and password to access the Informatica electronic software download site. Follow the instructions on the download site to download the Informatica Connector Toolkit installation file.
- Informatica Technology Network.
  If you are a registered user of the Informatica Technology Network, you can download the Informatica Connector Toolkit installation file from the Informatica Development Platform page. When you download the file, the Informatica Development Network provides you with a password. Use this password when you extract the files from the download file.

When you run the Informatica Connector Toolkit installer, the installer installs the Informatica Connector Toolkit Eclipse plug-in in the Eclipse IDE.

Before You Begin

Before you develop an adapter with the Informatica Connector Toolkit, you need to install the required software and analyze the data source.

Run the Informatica Connector Toolkit installer to install the Informatica Connector Toolkit on the machine where you plan to develop the adapter. The Informatica Connector Toolkit contains the binaries, tools, samples, and documents that you require to build an adapter.

Install Required Software

Install the following software on the machine where you plan to develop the adapter:

- Eclipse version with support for Plug-in Development Environment (PDE). Informatica Connector Toolkit supports the following Eclipse versions:
  - Eclipse Kepler 4.2.2 and 4.2.3.
  - Eclipse Luna 4.4.2 and 4.4.3.
  - Eclipse Mars 4.5.0 M1 and 4.5.1.
  - For AIX, Eclipse 4.2.0 M1.
- JDK version 1.7.x.
- Informatica services and clients version 10.1.0 to validate and use the adapter.
- If you implement the adapter run-time behavior in C/C++, install the following software:
  - For Windows, MinGW 4.8.1 or later, CMake 3.0.2 or later, and Microsoft Visual C++ 2013.
  - For Unix, gcc/g++ 5.2 and CMake 2.6.
  - For AIX, gcc/g++ 4.8.2 or later.
- MySQL Connector/J JDBC driver version 5.1.26 or later if you use the MySQL sample adapter.
- YouTube Data API client library for Java version 3 or later if you use the YouTube sample adapter.

Installation of Eclipse IDE

You can install Eclipse IDE on the machine in which you plan to develop the adapter. Use the Eclipse IDE package to install Eclipse IDE.

You must install Eclipse version supported by Informatica Connector Toolkit for the Informatica Connector Toolkit plug-in to work with the Eclipse IDE.

You can download the Eclipse IDE package from the following location:

http://www.eclipse.org/downloads/

Installing Informatica Connector Toolkit on Windows

Install the Informatica Connector Toolkit to install the Informatica Connector Toolkit Eclipse plug-in and other components that you require to build an adapter.

1. Close all other applications.
2. Run the `install.bat` file from the root directory.
3. In the **Welcome** page, click **Next**.

   The **Installation Directory** page appears.
4. Select the installation directory in which you want to install the Informatica Connector Toolkit.
5. Select the installation directory in which you installed Eclipse IDE.
6. Select **Install Apache log4j** to install the log4j plug-in version that Informatica Connector Toolkit requires. The installer installs the required Apache log4j plug-in by default. If you have log4j plug-in installed and do not want to install a different version of the library, clear the **Install Apache log4j** option.
7. Select **Install Apache Commons library** to install the commons library that Informatica Connector Toolkit requires.
8. Click **Next**.

   The **Pre-Installation Summary** page appears.
9. Click **Install**.

   The **Installing** page appears and displays the installation progress.
10. Click **Done** to complete the installation procedure and then exit the installer.

Installed Informatica Connector Toolkit Components

After you install the Informatica SDKs, you can find the Informatica Connector Toolkit in the installation directory where you installed the Informatica Connector Toolkit.

The Informatica Connector Toolkit installation includes the following components to assist you in developing an adapter for the Informatica platform:

- Informatica Connector Toolkit API files. Library files of the Informatica Connector Toolkit API.
- Informatica Connector Toolkit Eclipse plug-in. You can use the wizards and menus that the Informatica Connector Toolkit plug-in adds to the Eclipse IDE to develop, test, and deploy an adapter.
- Sample MySQL, Instagram, and YouTube adapters. The sample MySQL, Instagram, and YouTube adapters include source code that you can use as a model to build an adapter.

  Note: The sample MySQL, Instagram, and YouTube adapters are for illustration purposes only.
- Informatica Connector Toolkit API Reference. Online documentation for the Informatica Connector Toolkit API specification.

CHAPTER 3

Building an Adapter

This chapter includes the following topics:

Building an Adapter Overview

The Informatica Connector Toolkit consists of the libraries, plug-ins, and sample code to assist you in developing an adapter for the Informatica platform.

To build an adapter, use the Informatica Connector Toolkit and perform the following tasks:

1. Create an adapter project and define the basic properties of the adapter such as name, ID, and vendor name.
2. Define the connection attributes to connect to the data source. Implement the methods to open connection or close connection to the data source, validate connection, and specify attribute dependencies. Before you define the type system for the adapter, you can test and debug the connection components of the adapter.
3. Define the type system for the adapter. Specify the data types supported by the data source and the corresponding data types supported by Informatica.
4. Define the adapter metadata, create native metadata object (NMO), operations for the NMO, and partition methods for the NMO operations. Implement the methods to fetch metadata from the data source. You can also test and debug the metadata components of the adapter.

5. Define the adapter run-time behavior that defines how the adapter reads from and writes to the data source. Before you deploy the adapter, you can test and debug the read capability and write capability of the adapter.
6. Deploy the adapter to the Informatica server and client directories and register the adapter with the Informatica domain.

Step 1. Define the Adapter Properties

The adapter properties describe and identify the adapter in the Informatica domain. Use the Informatica Connector Toolkit to create an adapter project and define the adapter properties.

1. From the Eclipse IDE, click the **Connector** button ().
   The **Create Informatica Connector Project** dialog box appears.
2. Enter the following adapter project details:

Property	Description
Connector ID	An identifier that uniquely identifies the adapter. For example: mysql.
Connector Name	Name of the adapter. The adapter name is an alphanumeric string. The first character of the name must be a letter. For example: PowerExchange for MySQL.
Vendor Name	Name of the vendor building the adapter.
Vendor ID	A unique identifier for the vendor.
Version	The version number of the adapter must have the format x.x.x and must include numeric characters. For example: 1.0.0.
Description	Description of the adapter.

3. Click **Finish**.
   The adapter project appears in the **Connector Navigator** view of the Eclipse IDE.

Step 2. Define the Connection Attributes

The connection attributes of an adapter determine how the adapter connects to the data source. Use the Informatica Connector Toolkit to define the connection attributes and specify the libraries required to connect to the data source.

1. In the **Connector Navigator** view, right-click the project and select **Add Connection**.
   The **Add Connection** dialog box appears.
2. Enter the connection type name.
3. Select **Add New** and enter the category name.

The category name that you configure corresponds to the connection object category.

4. Click **Next**.
   The **Connection Attributes** page appears.
5. Click **Add** to enter each connection attribute.
   The attributes that you configure correspond to the connection object properties. The **Add Attribute** dialog box appears.
6. Enter the following properties for the connection attribute:

Attribute Property	Description
Name	Name of the connection attribute.
Display Name	Display name for the connection attribute.
Description	Description of the connection attribute.
Data Type	Data type of the connection attribute.
Default Value	Default value for the attribute.
Min Length	Minimum length for the value of the attribute.
Max Length	Maximum length for the value of the attribute.
Encrypted	Indicates whether you can encrypt the attribute.
Mandatory	Indicates whether a connection attribute requires a value. If you set the Mandatory property to true but you do not display the attribute on the connection management user interface, you must set a default value for the attribute.
Hidden	Indicates whether you can hide the attribute.
Has Dependent Fields	Indicates whether the attribute has dependent fields.
Allowed Values	List of values allowed for the attribute.

7. Click **OK**.
8. Click **Next**.
   The **Configure Connection Pages** page appears.
   a. Group the connection attributes under one or more connection sections.
   b. To create a new connection section, click **Add Section** and enter the section name, section title, tool tip, and section description.
   c. To create a new connection page, click **Add Page** and enter the page name and page description.
   d. To change the order in which connection sections appear in a connection page, use **Move Up** or **Move Down**.
9. Click **Next**.
   The **Configure Libraries** page appears.
10. Click **Add** to select each library that the adapter requires to connect to the data source.
11. Click **Generate Code**.

After you define the connection attributes, the Informatica Connector Toolkit generates the following Java files:

- <AdapterName>ConnectInfoAdapter.java
- <AdapterName>Connection.java

Update the ConnectInfoAdapter.java file to implement connection validation and attribute dependancies. Update the Connection.java file to implement the methods that open and close connection to the data source.

Note: If you regenerate code for the connection project, the Informatica Connector Toolkit does not regenerate code for the user-exposed source code visible in the Informatica perspective. You have to manually edit the source code and make changes if you add, remove, or modify connection attributes.

Test and Debug the Connection to the Data Source

After you define the connection attributes, you can test and debug the connection to the data source.

Create a debug configuration in Eclipse IDE from the **Run** menu. Select the adapter connection project to debug and provide the connection type and connection properties. You can also set breakpoints in the code that you want to debug.

After you create a debug configuration, you can launch the **Test Connection** dialog box to test and debug the connection definition.

1. In the **Connector Progress** view, select **Test Connection**.

   The **Test Connection** dialog box appears with the default Java Virtual Machine (JVM) environment settings. The Informatica Connector Toolkit uses the JVM settings to run the debug configuration.
2. Edit the JVM environment settings, if required. Ensure that you use the same port number that appears in JVM settings for the connection properties in the debug configuration.
3. Click **Next**.

   The **Connection Details** page appears with the connection attributes that you defined for the adapter.
4. Enter values for the connection attributes.
5. Click **Test Connection**.

   A success message appears if the connection to the data source is successful.

   Note: You cannot use the Informatica Connector Toolkit to test attribute dependencies.

Step 3. Define the Type System

Categorize each data type in the data source into one of the Informatica platform data types or Java data types supported by the Informatica Connector Toolkit API. Use the Informatica Connector Toolkit to create a type system or generate and use a predefined type system.

Defining the Type System Manually

You can manually define a type system that maps the data types that the data source supports with the data types that Informatica Platform supports.

1. In the **Connector Navigator** view, right-click the project and select **Define Typesystem**.
   The **Define Type System** dialog box appears.
2. To manually define a type system that matches the data types in the data source, select **Manually Create Type System**.
3. To map the native data types to the Informatica platform data types, select **Native to Platform Type**.
4. To map the native data types to the Java data types, select **Native to Java Type**.
5. Click **Add** to configure each type system attribute.
   The **Add Type** dialog box appears.
6. Enter the following properties for the type system attribute:

Property	Value
Type Name	Name for the native type system attribute.
Comments	Comments for the native type system attribute.
Best Platform Type for Read/ Best Java Type for Read	The platform or Java data type that maps best to the native data type when the adapter reads from the data source.
Properties	Based on the selected best platform or Java type for read, you must set one or more of the following native type properties: - Precision properties such as max precision and default precision. - Scale properties such as max scale, default scale, and min scale. - Length properties such as max length and default length. The maximum length of any attribute cannot be greater than Integer.MAX_VALUE, which is 2,147,483,647. - Unit of length such as characters, bytes, and bits. - Date properties such as hour, minute, second, year, month, day, and time zone.

7. If the platform data type or Java data type is the best match when writing to the native data type, select **Mapped** and then select **Best Native Type to Write**. For example, to map the Java INTEGER data type to the MySQL data type INT as one of the Best Native Type to Write, select Mapped in the INTEGER row and then select Best Native Type to Write.
8. If you want to map a platform data type or Java data type to the native data type but not as a best match, select only **Mapped**. For example, if the Java SHORT data type maps to the MySQL data type INT but might result in loss of data when converted, select Mapped in the INTEGER row.
9. Click **OK**. The **Define Type System** dialog box appears.
10. After you add and map the required native data types for the adapter, click **Generate Code**.

Generating a Predefined Type System

You can generate a type system with the basic data types that Informatica Platform supports. Use the basic data types in the predefined type system for use with REST-based procedure.

1. In the **Connector Navigator** view, right-click the project and select **Define Typesystem**.
   The **Define Type System** dialog box appears.

2. To generate and use basic data types that the Informatica Platform supports, select **Use Predefined Type System**.
3. To map the native data types to the Informatica platform data types, select **Native to Platform Type**.
4. To map the native data types to the Java data types, select **Native to Java Type**.
5. Click **Generate Code**.

Step 4. Define the Common Metadata

You can define the common metadata for Informatica Cloud connectors.

1. In the **Connector Navigator** view, right-click the project and select **Edit Common Metadata**.

   The **Edit Common Metadata** dialog box appears.
2. If the connector supports schema, select **Supports Schema** and add the **Schema Display Name**.
3. If the connector supports foreign key, select **Support Foreign Key** and add the **Foreign Key Display Name**.
4. Click **Generate Code**.

Step 5. Define the Adapter Metadata

The adapter metadata represents the metadata in the data source for which you build the adapter. You can define multiple metadata definitions to represent the differently structured metadata objects of the data source.

Use the Informatica Connector Toolkit to define the adapter metadata. You can represent the metadata for data sources in which data is stored as records and for procedures in data sources. You can manually create native metadata for procedures or use swagger specifications to define the native metadata.

You can use the procedure pattern to define native metadata objects for Informatica Cloud connectors.

Note: Procedure pattern is available only for Informatica Cloud and not supported for PowerCenter and Big Data Management in this release.

Define the following adapter components to specify the adapter metadata:

- Native metadata definition for the adapter. You can define multiple native metadata definitions for an adapter. For example, you can create different native metadata objects for tables, views, and synonyms in a relational data source.
- Record extensions and field extensions. You can define record extensions and field extensions to define additional metadata for records and fields.
- Read and write capability for the adapter. You can add attributes that you can use to read from or write to the data source.
- Import dialog box settings. You can define import options that appear in the Developer tool when an adapter consumer creates a data object.

Note: If you regenerate code for the native metadata definition project, the Informatica Connector Toolkit does not regenerate code for the user-exposed source code visible in the Informatica perspective. You have

to manually edit the source code and make changes if you add, remove, or change the native metadata attributes.

Defining the Adapter Metadata for Record Pattern

For data sources in which data is stored as records, you can define the native metadata definition for the adapter by using the record pattern type.

1. In the Connector Navigator view, right-click the project and select **Add Native Metadata Definition**. The **Add Native Metadata Definition** dialog box appears.
2. Enter the native metadata details.
   The following table describes the properties to enter:

Property	Description
Name	Name for the native metadata.
Display Name	Display name for the native metadata.
Description	Description of the native metadata.

3. Specify the pattern type as record to create a native metadata object based on the record structure of the data source.
4. To add additional metadata information for records, select **Add Record Extension** and add the following properties for the attribute:

Attribute Property	Description
Name	Name of the attribute.
Display Name	Display name for the connection attribute.
Description	Description of the connection attribute.
Data Type	Data type of the connection attribute.
Default Value	Default value for the attribute.
Min Length	Minimum length for the value of the attribute.
Max Length	Maximum length for the value of the attribute. The maximum length of any attribute cannot be greater than Integer.MAX_VALUE, which is 2,147,483,647.
Encrypted	Indicates whether you can encrypt the attribute.
Mandatory	Indicates whether a connection attribute requires a value. If you set the Mandatory property to True but you do not display the attribute on the connection management user interface, you must set a default value for the attribute.

Attribute Property	Description
Hidden	Indicates whether you can hide the attribute.
Allowed Values	List of values allowed for the attribute.

5. Click **Next**.
   The **Native Metadata Field** page appears.
6. To add additional metadata information for fields, select **Add Field Extension** and add the attribute properties.
7. Click **Next**.
   The **Native Metadata Read Capability** page appears.
8. To define read capability for the native metadata object, select **Enable Read Capability**.
9. Select whether the adapter supports lookup of data when the adapter reads from the data source.
10. Select whether the adapter supports join and filter operations when the adapter reads from the data source.
    - To specify operators and expression syntax recognized by the Informatica platform for the join or filter operation, select **Platform Expression**.

      Note: If you plan to add key range partitioning capability for the adapter, you must select support for filter operation and platform expression.
    - To specify an expression for the join or filter operation that is specific to the data source for which you build the adapter, select **Native Expression**.
11. Select whether the adapter supports sort filter to retrieve data from the data source in a specific order.
12. Select whether the adapter supports select filters when the adapter reads from the data source.
13. To add read capability attributes, click **Add** and add the attribute properties.
    You can also define the following parameterization and partitioning support for an attribute:
    - When you specify the read attribute properties, you can define whether you can parameterize the attribute. When you parameterize an attribute, you can assign values for the attribute at run time. For example, you can parameterize the user name and add user names to a configuration file and then run the same mapping to read data of different users.

      You can define the following parameterization support for an attribute:
      Full Parameterization

      The attribute supports parameterization. You can parameterize the value of an attribute completely.

      Partial Parameterization

      The attribute supports partial parameterization. You can parameterize a part of the attribute value.

      No

      The attribute does not support parameterization.
    - You can select **Override Partitions** to specify if the attribute can be overridden for each partition. Implement the <ConnectorID><NMOName>AutoPartitioningMetadataAdapter file to define the partition support.

    You can also define whether the attribute supports paramertization and if the attribute can be overridden for each partition.
14. Click **Next**.

The **Native Metadata Write Capability** page appears.

15. To define write capability for the native metadata object, select **Enable Write Capability** and add the attribute properties.
When you specify the write attribute properties, you can define whether you can parameterize the attribute.

    You can define the following parameterization support for an attribute:

    Full Parameterization

    The attribute supports parameterization. You can parameterize the value of an attribute completely.

    Partial Parameterization

    The attribute supports partial parameterization. You can parameterize a part of the attribute value.

    No

    The attribute does not support parameterization.

16. Select whether the adapter supports upsert operation when the adapter writes to the target.
When you select **Enable Upsert Support**, Informatica Connector Toolkit adds the `UpdateMode` attribute to the write capability attribute list.

    You can specify one of the following values for the `UpdateMode` attribute during runtime:

    - **Update As Update**. If you specify the **Update As Update** value, you must implement the upsert logic so that the adapter updates an existing row while writing to the target.
    - **Update Else Insert**. If you specify the **Update Else Insert** value, you must implement the upsert logic so that the adapter updates an existing row if the row exists in the target, else inserts a row while writing to the target.

17. Click **Next**.
The **Native Metadata Partitioning Capability** page appears.
18. Select whether the adapter supports partitioning capability for the read operation.
19. To configure the adapter to fetch partition information from the data source, select the **Dynamic** partitioning method and implement the partition logic. Extend the AutoPartitioningMetadataAdapter class to implement the partition logic.
20. To configure the adapter to get partition information from the user, select the **Static** partitioning method. The user enters the partition information, such as number of partitions or key range.
    - If you want to implement a partition logic based on the partition information that the user specifies, select **Fixed**. Implement the partition logic in the DataAdapter class. The user can specify the required partition information before the adapter reads from the data source.
    - If the tables in the data source support key range partitioning, select **Key Range**. The user can specify the partition keys and key range type before the adapter reads from the data source. If you add support for key range partitioning, ensure that the adapter supports filter operation and platform expression. You need not implement the partition logic for key range partitioning because the Informatica Connector Toolkit implements key range partitioning as a filter query.

      Note: Key range partitioning is supported only for Big Data Management.
21. Select whether the adapter supports partitioning capability for the write operation. By default, the dynamic partitioning method is selected for partition-enabled write operations. Extend the AutoPartitioningMetadataAdapter class to implement the partition logic.
22. Click **Next**.
The **Import Dialog Box Settings** page appears.
23. In the **Metadata Import Dialog Box Settings** section, select the metadata import options that appear in the client tool when an adapter consumer creates a data object and click **Save**.

24. If you are developing a connector for the Informatica Cloud, specify whether the adapter supports DDL generation and select the supported modifications to the data source schema in the **Metadata Write Settings** section.
You can configure the connector to support the following modifications in the data source schema:
Create Object

    To add the capability to create objects in the target, select **Create Object**. To add the capability to drop the existing object and then create a new object, select **Drop and Create**.

    Alter Object

    To add the capability to alter objects in the target, select **Alter Object**. To add the capability to create a object if the object to alter does not exist, select **Alter Else Create**.

    Drop Object

    To add the capability to drop objects in the target, select **Drop Object**.

25. Click **Generate Code**.
After you define the adapter metadata, the Informatica Connector Toolkit generates the `<NMOName>MetadataAdapter.java` file in the **Metadata** folder. Implement the following methods in the `<NMOName>MetadataAdapter.java` file to import metadata.

    populateObjectCatalog()

    Populates metadata details in the import wizard for the adapter consumer.

    Note: Data preview in the PowerCenter client or Informatica Cloud does not work if the values of the Record.setName and Record.setNativeName methods are different.

    populateObjectDetails()

    Gets metadata from the data source based on the import dialog options settings.

If you configured metadata write settings for the connector, implement the `writeObjects` method in the `<NMOName>MetadataAdapter.java` file.

Note: If you regenerate code for the native metadata definition project, the Informatica Connector Toolkit does not regenerate code for the user-exposed source code visible in the Informatica perspective. You have to manually edit the source code and make changes if you add, remove, or change the native metadata attributes.

Define the Adapter Metadata for Procedure Pattern

For procedures and functions in data sources, you can define the native metadata definition for the Informatica Cloud connectors by using the procedure pattern type.

You can use the manual method to create the native metadata object for the procedure or use a REST-based specification. You can choose to use an existing swagger specification or generate and use a swagger specification.

Defining the Adapter Metadata for Procedure Pattern Manually

When you use the procedure pattern type to define the native metadata for the adapter, you can use the manual method to define the native metadata object.

Note: Procedure pattern is available only for Informatica Cloud and not supported for PowerCenter and Big Data Management in this release.

1. In the Connector Navigator view, right-click the project and select **Add Native Metadata Definition**.
The **Add Native Metadata Definition** dialog box appears.

2. Enter the native metadata details.
   The following table describes the properties to enter:

Property	Description
Name	Name for the native metadata.
Display Name	Display name for the native metadata.
Description	Description of the native metadata.

3. Specify the pattern type as procedure to create a native metadata object for procedures or functions in a data source.
4. To manually define the native metadata object for the data source with procedure pattern type, select **Manual** as the creation method.
5. Click **Next**.
   The **Native Metadata Procedure Definition** page appears.
6. To add additional metadata information for procedure, select **Add Procedure Extension** and add the following properties for the attribute:

Attribute Property	Description
Name	Name of the attribute.
Display Name	Display name for the connection attribute.
Description	Description of the connection attribute.
Data Type	Data type of the connection attribute.
Default Value	Default value for the attribute.
Min Length	Minimum length for the value of the attribute.
Max Length	Maximum length for the value of the attribute. The maximum length of any attribute cannot be greater than Integer.MAX_VALUE, which is 2,147,483,647.
Encrypted	Indicates whether you can encrypt the attribute.
Mandatory	Indicates whether a connection attribute requires a value. If you set the Mandatory property to True but you do not display the attribute on the connection management user interface, you must set a default value for the attribute.
Hidden	Indicates whether you can hide the attribute.
Allowed Values	List of values allowed for the attribute.

7. Click **Next**.
   The **Native Metadata Parameter Definition** page appears.

8. To add additional metadata information for parameters, select **Add Parameter Extension** and add the attribute properties.
9. Click **Next**.
   The **Native Metadata Call Capability Attributes** page appears.
10. To add call capability attributes, click **Add** and add the attribute properties.
    You can select **Override Partitions** to specify if the attribute can be overridden for each partition. Implement the <ConnectorID><NMOName>AutoPartitioningMetadataAdapter file to define the partition support.
11. Click **Next**.
    The **Import Dialog Box Settings** page appears.
12. In the **Metadata Import Dialog Box Settings** section, select the metadata import options that appear in the client tool when an adapter consumer creates a data object and click **Save**.
13. Click **Generate Code**.
    After you define the adapter metadata, the Informatica Connector Toolkit generates the `<NMOName>MetadataAdapter.java` file in the **Metadata** folder. Implement the following methods in the `<NMOName>MetadataAdapter.java` file to import metadata.

    populateObjectCatalog()

    Populates metadata details in the import wizard for the adapter consumer.

    Note: Data preview in the PowerCenter client or Informatica Cloud does not work if the values of the Record.setName and Record.setNativeName methods are different.

    populateObjectDetails()

    Gets metadata from the data source based on the import dialog options settings.

If you configured metadata write settings for the connector, implement the `writeObjects` method in the `<NMOName>MetadataAdapter.java` file.

Note: If you regenerate code for the native metadata definition project, the Informatica Connector Toolkit does not regenerate code for the user-exposed source code visible in the Informatica perspective. You have to manually edit the source code and make changes if you add, remove, or change the native metadata attributes.

Defining the Adapter Metadata for Procedure Pattern by Using REST-based Specification

You can define the native metadata definition for the adapter by using an existing REST-based specification or generate and use a REST-based specification.

Note: Procedure pattern is available only for Informatica Cloud and not supported for PowerCenter and Big Data Management in this release.

1. In the Connector Navigator view, right-click the project and select **Add Native Metadata Definition**.
   The **Add Native Metadata Definition** dialog box appears.
2. Enter the native metadata details.

The following table describes the properties to enter:

Property	Description
Name	Name for the native metadata.
Display Name	Display name for the native metadata.
Description	Description of the native metadata.

3. Specify the pattern type as procedure to create a native metadata object for procedures or functions in a data source.
4. To create native metadata object based on a REST-based specification, select REST-based creation method.
5. To use an existing swagger specification, click **Browse** and open the JSON file. After you open the JSON file, skip to step 7.
6. To generate a swagger specification by sampling, select **Generate swagger specification by sampling** and then click **Next**.
   Perform the following steps to sample the REST end point:
   a. Enter the base URL.
   b. Click **Add** to get the procedure details.
   c. Specify the **Path**, **Display Name**, **Method**, **Content Type**, and **Accept Type** properties.
   d. Add the parameter details.
   e. Click **Test** to validate and sample the REST end point.
   f. Click **OK** to add the procedure details.
7. Click **Generate Code**.
   After you define the adapter metadata, the Informatica Connector Toolkit generates the `<NMOName>MetadataAdapter.java` file in the **Metadata** folder. To implement features specific to the data source, you can also modify code in the following methods in the `<NMOName>MetadataAdapter.java` file to import metadata.

   populateObjectCatalog()

   Populates metadata details in the import wizard for the adapter consumer.

   Note: Data preview in the PowerCenter client or Informatica Cloud does not work if the values of the Record.setName and Record.setNativeName methods are different.

   populateObjectDetails()

   Gets metadata from the data source based on the import dialog options settings.

If you configured metadata write settings for the connector, implement the `writeObjects` method in the `<NMOName>MetadataAdapter.java` file.

Note: If you regenerate code for the native metadata definition project, the Informatica Connector Toolkit does not regenerate code for the user-exposed source code visible in the Informatica perspective. You have to manually edit the source code and make changes if you add, remove, or change the native metadata attributes.

Test Metadata from the Data Source

After you define the native metadata objects, you can test metadata that you import from the data source.

To debug when you test metadata from the data source, use the same debug configuration that you used to test the connection to the data source. You can also set breakpoints in the code that you want to debug.

After you define the debug configuration, you can launch the **Test Metadata** dialog box to test the connection definition and test the metadata import from the data source.

1. In the **Connector Progress** view, select Test<NMOName> under the **Test Metadata** section.

   The **Test Metadata** dialog box appears with the connection attributes that you defined for the adapter.
2. Enter values for the connection attributes to connect to the data source.
3. Click **Connect**.

   The adapter retrieves the metadata from the data source and the metadata appears in the **Test Metadata** page.
4. Browse and verify the retrieved metadata.
5. Click **Close**.

Step 6. Implement the Adapter Run-Time Behavior

Use the Informatica Connector Toolkit to implement the adapter run-time behavior in C/C++ or Java. The adapter run-time behavior defines how the adapter performs operations, such as establishing a connection, closing a connection, preparing SQL statements, and running SQL statements.

You can set up the run-time implementation for each NMO of the adapter. For example, if you define an NMO with read capability and another NMO with write capability, you can choose to implement the run time for the first NMO with read capability in C++ and implement the run time for the other NMO with write capability in Java.

You can also define the adapter run-time behavior to support pre and post commands to perform tasks before and after a mapping run. For example, you can define the adapter run-time behavior to support a pre command that initializes environment variables before the mapping run.

Implement the Adapter Run-Time Behavior in Java

You can use the **Run-Time Implementation** wizard in the Informatica Connector Toolkit to implement the run-time behavior in Java.

1. In the **Connector Navigator** view, right-click the project and select **Runtime** > **<NMOName>** > **Set Up**.

   The **Run-Time Implementation** wizard appears.
2. Choose to implement the run-time behavior in Java.
3. Add the required run-time library files and generate the `<ConnectorID><NMOName>DataAdapter.java` and `<ConnectorID>DataConnection.java` files.

4. Implement the following methods in the `<ConnectorID><NMOName>DataAdapter.java` file:
   - initDataSourceOperation. Implement this method to perform tasks before a mapping run. For example, you can implement code to initialize environment variables. The scope of the RuntimeConfig and Metadata handles available in this method is within the initDataSourceOperation method.
   - deinitDataSourceOperation. Implement this method to perform tasks after a mapping run. The scope of the RuntimeConfig and Metadata handles available in this method is within the deinitDataSourceOperation method.
   - initDataSourceOperation
   - deinitDataSourceOperation
   - initDataSession
   - deinitDataSession
   - read
   - write
   - reset (Optional. Implement this method if lookup operation is supported.)
   - beginDataSession
   - endDataSession
5. Implement the following methods in the `<ConnectorID>DataConnection.java` file:
   - connect
   - disconnect
6. To support data preview in the Informatica Cloud, implement the following method in the `<ConnectorID>ASOOperationObjMgr.java` file:

   prepareRuntimeOperation

Implement the Adapter Run-Time Behavior in C/C++

You can use the **Run-Time Implementation** wizard in the Informatica Connector Toolkit to implement the run-time behavior in C/C++.

1. In the **Connector Navigator** view, right-click the project and select **Runtime** > **<NMOName>** > **Set Up**.

   The **Run-Time Implementation** wizard appears.
2. Choose to implement the run-time behavior in C/C++.
3. Add the supported platform for which you need to implement the run-time behavior in C/C++.
4. Configure the following run-time settings for the platform that you select:
   - Include directory path and compile options
   - Library paths, library names, and link options
   - Environment variables
   - Other run-time library files

   For Windows platform, you can choose to generate the C/C++ project for the Eclipse IDE or Visual Studio.
5. Open the C++ project and implement the following methods in the <ConnectorID><NMOName>RuntimeAdapter.cpp file:
   - INFAADPInitPlugin

- INFAADPDeinitPlugin
- INFAADPInitDataSourceOperation. The scope of the RuntimeConfig and Metadata handles available in this method is within the INFAADPInitDataSourceOperation method.
- INFAADPDeinitDataSourceOperation. The scope of the RuntimeConfig and Metadata handles available in this method is within the INFAADPDeinitDataSourceOperation method.
- INFAADPInitDataSession
- INFAADPDeinitDataSession
- INFAADPBegindataSession
- INFAADPEndDataSession
- INFAADPConnect
- INFAADPDisConnect
- INFAADPRead
- INFAADPWrite

Create Messages

You can use the Informatica Connector Toolkit to create, edit, or delete messages and handle exceptions that occur during the design time or run time of the adapter.

When you create messages, you specify the message text and message code, and you can include information on the message severity, cause, and user action. After you create messages, you can implement the code to handle exceptions. When you implement the code to handle the exception, you pass the message as an argument to the exception handing method.

Create design-time messages to handle design-time exceptions, such as service exceptions. Create run-time messages to handle run-time exceptions.

To create messages, perform the following steps:

1. In the **Connector Navigator** view, right-click the project and select **View and Create Messages**. The **Messages** dialog box appears.
2. Click **Add**. The **Add New Message** dialog box appears.
3. Enter an ID for the message.
4. Enter a code for the message. At run time, the message code and the message text appears in the session log.
5. Specify the severity of the message.
6. Enter the message text. You can include parameters in the message text and specify the parameters in C format or Java message format.
   The following examples show parameters used in C format and Java message format:
   - The parameters in the following message text uses C format: Connection User [%s], Port [%d], Connection time [%f] milliseconds.
   - The parameters in the following message text uses Java format: Connection User [{0}], Port [{1,number,integer}], Connection time [{1,number}] milliseconds.
7. Enter a description for the message.
8. Enter the cause of the error message.

9. Enter the suggested user action when the user encounters the error.
10. Specify whether the message is a design-time message or a run-time message.
11. Click **OK**. The **Messages** dialog box appears.
12. Click **Finish**. The message XML files appear under the **Message** folder in the **Connector Navigator** view.

Implement Design-Time Messages

You can use the methods in the com.informatica.sdk.exceptions.ExceptionManager class to implement design-time messages.

To enable localization of messages, implement the createNlsAdapterSDKException() method in the ExceptionManager class.

The following sample code shows the message parameters passed to the createNlsAdapterSDKException() method:

```
ExceptionManager.createNlsAdapterSDKException(ExceptionManager.createNlsAdapterSDKExcepti
on(MessageBundle.getInstance(),
Messages.Test_CONN_SUCC_200, "admin", 5040, 138.76);
```

To implement design-time messages that do not require localization, implement the createNonNlsAdapterSDKException() method in the ExceptionManager class.

The following sample code shows the message string passed to the createNonNlsAdapterSDKException() method:

```
ExceptionManager.createNonNlsAdapterSDKException("Unknown error:" + e.getMessage());
```

Implement Run-Time Messages

To implement run-time messages that require localization, implement the logMessage() helper method in the <ConnectorID><NMOName>DataAdapter.<C or Java> file. The logMessage() method logs messages to the session log.

The following Java sample code shows the message parameters passed to the logMessage() method:

```
logMessage(Messages.Test_CONN_SUCC_200, "admin", 5040, 138.76);
```

The following C sample code shows the message parameters passed to the INFAADPLogMessage() method:

```
INFAADPLogMessage(infaDataSessionHandle, INFA_MSG_ERROR,INFA_TRACE_NONE, CONN_ID,"admin",
12,12.3);
```

To implement the run-time messages that do not require localization, implement the logger.logMessage() method.

Test the Read and Write Capabilities of the Adapter

Before you deploy the adapter in the Informatica domain, you can test the read and write capabilities of the adapter.

After you define the adapter run-time components, you can use the **Test Read** and **Test Write** wizards to test the read and write capabilities of the adapter. After you debug and fix issues in the read and write capabilities of the adapter, you can deploy the adapter in the Informatica domain.

You can test the read and write capabilities of the adapter only for the Windows platform.

Test the Read Capability of the Adapter

When you test the read capability of the adapter, you test the connection definition, metadata of the data source, and operations that the adapter supports. After you specify the test settings and run the test, you can view the result of the read operation, read operation statistics, and the log file.

To debug the code, use the same debug configuration that you used to test the connection and metadata components of the adapter. You can also set breakpoints in the code that you want to debug.

After you define the debug configuration, you can launch the **Test Read** dialog box to test the read capabilities of the adapter.

1. In the **Connector Progress** view, select the native metadata object that appears under **Test Read**.

   The **Test Read** dialog box appears with the default JVM environment settings and tracing level. The Informatica Connector Toolkit uses the JVM settings to run the debug configuration.
2. If required, edit the JVM environment settings. Ensure that you use the same port number that appears in JVM settings for the connection properties in the debug configuration.
3. Select the required tracing level. The default is normal. Based on the amount of detail that you require in the log file, you can override the default tracing level.

   You can set the following types of tracing level:

   None

   Does not override the default tracing level.

   Terse

   Logs initialization information and error messages and notification of rejected data.

   Normal

   Logs initialization and status information, errors encountered, and skipped rows due to transformation row errors. Summarizes mapping results, but not at the level of individual rows. This is the default tracing level.

   Verbose Initialization

   In addition to normal tracing, logs additional initialization details, names of index and data files used, and detailed statistics.

   Verbose Data

   In addition to verbose initialization tracing, logs each row. You can also get detailed statistics on where string data was truncated to fit the precision of a column.
4. Click **Next**.

   The connection attributes that you defined for the adapter appears.
5. Enter values for the connection attributes to test the connection to the data source.
6. Click **Connect**.

   The **Test Metadata** page appears with the metadata imported from the data source.
7. Select the native metadata objects and the corresponding native metadata fields to test the read operation.
8. Click **Next**.

   Based on whether the selected native metadata object supports join and filter operations, the **Join Condition** page or **Filter Condition** page appears.
9. If the native metadata object supports join operation, specify an expression in the **Join Condition** page.

- To specify an Informatica platform expression for the join operation, perform the following steps:
    1. In the **Definition** section, select the native metadata object for which you want to specify the join condition.
    2. To change the join order, click **Move Up** or **Move Down**.
    3. Specify the join type in **Join condition**.
    4. To define a join condition, select values in the **Left Field**, **Operator**, and **Right Field** columns.
    5. To add additional join conditions, click **Add**.
    6. To remove a join condition, select the join condition and click **Remove**.
- To specify a native expression for the join operation, enter the expression in the **Definition** section.

10. If the native metadata object supports filter operation, select the **Configure Filter** option in the **Filter Condition** page and specify the filter condition.
    - To specify an Informatica platform expression for the filter operation, perform the following steps:
        1. In the **Definition** section, click **Add** to add an Informatica platform expression.
        2. In the **Field** column, select the field to use in the expression.
        3. In the **Operation** column, select a conditional operator to use in the expression.
        4. In the **Value** column, enter a value for the conditional expression.
    - To specify a native expression for the filter operation, enter the expression in the **Definition** section.
11. After you specify expressions for the native metadata object, click **Next**.

    The **Read Capability** page appears.
12. Specify values for the read capability attributes and then click **Run**.

    The **Result** page appears. You can view the result of the read operation, read operation statistics, and the log file in the **Result** page.
13. Click **Close**.

Test the Write Capability of the Adapter

When you test the write capability of the adapter, you test the components of the adapter and write sample data to the data source. After you specify the test settings and run the test, you can view the result of the write operation, write operation statistics, and the log file.

To debug the code, use the same debug configuration that you used to test the connection and metadata components of the adapter. You can also set breakpoints in the code that you want to debug.

After you define the debug configuration, you can launch the **Test Write** dialog box to test the write capabilities of the adapter.

1. In the **Connector Progress** view, select the native metadata object that appears under **Test Write**.

   The **Test Write** dialog box appears with the default JVM environment settings and tracing level. The Informatica Connector Toolkit uses the JVM settings to run the debug configuration.
2. If required, edit the JVM environment settings. Ensure that you use the same port number that appears in JVM settings for the connection properties in the debug configuration.
3. Select the required tracing level. The default is normal. Based on the amount of detail that you require in the log file, you can override the default tracing level.

   You can set the following types of tracing level:

None

Does not override the default tracing level.

Terse

Logs initialization information and error messages and notification of rejected data.

Normal

Logs initialization and status information, errors encountered, and skipped rows due to transformation row errors. Summarizes mapping results, but not at the level of individual rows. This is the default tracing level.

Verbose Initialization

In addition to normal tracing, logs additional initialization details, names of index and data files used, and detailed statistics.

Verbose Data

In addition to verbose initialization tracing, logs each row. You can also get detailed statistics on where string data was truncated to fit the precision of a column.

4. Click **Next**.

   The connection attributes that you defined for the adapter appears.
5. Enter values for the connection attributes to test the connection to the data source.
6. Click **Connect**.

   The **Test Metadata** page appears with the metadata imported from the data source.
7. Select a native metadata object to test the write operation.

   The metadata of the native metadata object along with the data type, scale, and precision appears in the **Test Write** page.
8. Select the columns to which you want to write data.
9. Click **Next**.

   The **Test Data** page appears.
10. In the Test Data page, you can load test data from a file or you can generate test data.
    - To load the test data from a file, perform the following steps:
      1. Select the **Load from a File** option. You must load a comma-delimited TXT file or CSV file.

         Note: The date and time data types in the file must have the following timestamp format: MM/DD/YYYY hh24:mm:ss
      2. Click **Browse** and select the file that contains the test data.
    - To generate test data, perform the following steps:
      1. Select the **Auto generate data** option.
      2. Enter the number of rows to generate. You can specify a maximum of 1000 rows.
      3. Click **Generate**. The test data appears in the **Data Preview** section.
      4. If required, you can edit the test data that appears in the **Data Preview** section.
11. After you load a test data file or generate test data, click **Next**.

    The **Write Capability** page appears.
12. Specify values for the write capability attributes and then click **Run**.

    The **Result** page appears. You can view the result of the write operation, write operation statistics, and the log file in the **Result** page.

13. Click **Close**.

Step 7. Export or Publish the Adapter

Use the Informatica Connector Toolkit to export or publish the adapter. You can export the adapter files to a location or publish the adapter on local Informatica services and client directory.

1. In the **Connector Navigator** view, right-click the project and select **Publish**.
   The **Export/Publish Informatica Connector** dialog box appears.
2. Click **Browse** and specify a location to export the adapter.
   The **Export Location** defaults to the adapter project workspace.
3. To export the connector for Informatica Cloud, select **Informatica Cloud** and specify the following details:
   - If you use record pattern type to create the connector, specify a plugin version.
   - If you use procedure pattern type to create the connector, select **Specify Plugin ID** and specify a plugin ID and a plugin version.
4. To export the connector for PowerCenter, select **PowerCenter** and specify the plugin ID.
5. If you want to use the Informatica Connector Toolkit to install the adapter on a local Informatica services and client machine, select **Publish connector on local Informatica services and client** and perform the following steps:
   a. Specify the location of the local Informatica server directory.
   b. Specify the location of the client installation directory.

   To reinstall the adapter or install a newer version of the adapter, select **Upgrade**.
6. Click **Next** to view the contents of the client and server bundles.
7. Click **Finish**.

If you export the adapter, the Informatica Connector Toolkit bundles the client and service artifacts and generates batch files that you can use to install the adapter. If you encounter any error or fail to export the adapter, refer to the <adapter_name>_codebuilder.log file for more information. The <adapter_name>_codebuilder.log file is available in the <Eclipse workspace>/<adapter project> folder.

If you publish the adapter, the Informatica Connector Toolkit restarts the Informatica domain and registers the adapter with the Informatica services. The Data Integration Service can connect to the data source and you can create connections to the data source in the Developer tool and Administrator tool. If you fail to publish the adapter, install the adapter manually.

Note: If the Informatica domain is not available after you publish the adapter, ensure that no Informatica services are running in the domain and then publish the adapter.

Manually Install the PowerCenter Adapter

You can manually install the PowerCenter adapter on Windows or Linux after you build the adapter using the Informatica Connector Toolkit.

Install the PowerCenter Adapter on Windows (Optional)

To install the adapter on Windows, deploy the adapter to the Informatica services and client installation directories. You must also register the adapter to the Informatica services installation directory. If you use the **Publish connector on local Informatica services and client** option to deploy the adapter, you do not need to complete these steps.

After you use the Informatica Connector Toolkit to export the adapter, perform the following steps to manually install the adapter:

1. Shut down the Informatica domain and Informatica Developer tool applications on which you plan to deploy the adapter files.
2. Perform the following steps to deploy and register the adapter to the server installation directory:
   a. Copy the `installServer.bat` file and the server folder from the publish location to the machine in which you have installed Informatica services.
   b. Launch a command window and navigate to the location of the `installServer.bat` file.
   c. Run following command in the command window: `installServer.bat [InformaticaInstallationDir]`

      For example, `installServer.bat C:\Informatica\10.1.1`
   d. If you need to reinstall the adapter or install a newer version of the adapter, run the following command in the command window: `installServer.bat [InformaticaInstallationDir] true`
3. Perform the following steps to deploy the adapter files to the client installation directory:
   a. Copy the `installClient.bat` file and the client folder from the publish location to the machine in which you have installed Informatica client.
   b. Launch a command window and navigate to the location of the `installClient.bat` file.
   c. Run following command in the command window: `installClient.bat [InformaticaClientInstallationDir]`

      For example, `installClient.bat C:\Informatica\10.1.1\clients`
   d. Run the following adapter registry file:

      For a Windows 32-bit system: `m/c:<adapter>.reg`
      For a Windows 64-bit system: `m/c: <adapter>_64.reg`

      The generated adapter registry file is deployed in the following location:

      `<deploy>\Informatica_PowerCenter\client\PowerCenterClient\client\bin`
4. Start the Informatica domain.

Install the PowerCenter Adapter on Linux

To install the adapter on Linux, you must copy the adapter plugins and third-party jars to the installation directory. Then, register the adapter to the PowerCenter Repository Service.

To manually install the adapter on Linux, perform the following steps:

1. Copy the adapter plugins from the following directory:

   `/deploy/Informatica_PowerCenter/server/server/connectors/cci/plugins`

2. Paste the adapter plugins to the following directory:

   `<Installation_Directory>/server/connectors/cci/plugins`

3. Register the adapter plugin available in the following directory to the PowerCenter Repository Service:

   `/deploy/Informatica_PowerCenter/server/server/bin/Plugin`

Naming Convention

A naming convention makes it easy to identify the files that belong to an adapter and the usage of those files.

Use the following guidelines when you name the adapter files:

- Determine a name for the adapter. Use the adapter name as a prefix for the adapter file names. The adapter name is an alphanumeric string that can include the uppercase and lowercase letters A to Z and the numbers 0 to 9. The first character of the name must be a letter.
- Determine a unique name to identify the company building the adapter. The company name is included in the package name for the adapter classes.
- Use the adapter name when you create a directory for the adapter in the Informatica directory:

  `INFA_HOME\plugins\dynamic\`*`AdapterName`*

The following table lists the recommended naming convention for adapter files:

Component	Naming Format
Package for the adapter definition and type system classes	com.*vendorname*.*adaptername*.connection
Adapter definition class	*AdapterName*Defn
Type system class	*AdapterName*TypeSystem
Resource file for the connection management user interface	*AdapterName*Bundle.properties or *AdapterName*Bundle_*lang*.properties

CHAPTER 4

Connection Attributes

This chapter includes the following topics:

Connection Attributes Overview

The adapter connection attributes determine the behavior and functionality of the adapter when it connects to a data source. You can configure connection attributes such as connection-level encryption, connection pool settings, database user credentials, and other database-specific attributes.

Define the connection attributes that are specific for the data source. You can set the default value of a connection attribute or mark a connection attribute as a required attribute.

Connection Attribute Properties

You can configure properties for each connection attribute that you define for an adapter. You can specify whether an attribute is required, provide the default, maximum, and minimum values for the attribute, or specify a list of possible values for the attribute.

All connection management user interfaces, including command line programs, validate the values set for the properties of a connection attribute.

The following table describes the properties that you can set for a connection attribute:

Attribute Property	Description
Name	Name of the connection attribute.
Display Name	Display name for the connection attribute.
Description	Description of the connection attribute.
Data Type	Data type of the connection attribute.
Default Value	Default value for the attribute.

Attribute Property	Description
Min Length	Minimum length for the value of the attribute.
Max Length	Maximum length for the value of the attribute.
Encrypted	Indicates whether the attribute is encrypted.
Mandatory	Indicates whether a connection attribute requires a value. If you set the Mandatory property to true but you do not display the attribute on the connection management user interface, you must set a default value for the attribute.
Hidden	Indicates whether the attribute can be hidden.
Has Dependent Fields	Indicates whether the attribute has dependent fields.
Allowed Values	List of values allowed for the attribute.

CHAPTER 5

Type System

This chapter includes the following topics:

Type System Overview

A type system is a framework that specifies characteristics of data types. The adapter type system must define the data types that the data source supports, the semantic category that matches the data type, and how they map to the data types that the Informatica platform supports.

The type system you define for an adapter consists of the following sets of data types:

- Native types. Data types that the data source supports.
- Semantic types. Semantic category that matches the native data type. For example, the semantic type Length matches the native data types, such as Char, Varchar2, Binary, Varbinary, Blob, and Clob.
- Informatica platform types. Data types that the Informatica platform supports. The Informatica Connector Toolkit API uses ODBC as a model for describing Informatica platform data types.

Native Types and Semantic Categories

The native types included in the type system are all possible data types in the data source for which the adapter is built.

You must define the semantic category of each data type in the data source. The Informatica Connector Toolkit API defines each semantic category. Check the data source documentation to verify the data types that are available in the data source.

Use the Informatica Connector Toolkit to associate native data types with semantic categories. The data type name must match the character string that corresponds to the type name returned during the metadata import process, such as Integer, Varchar2, or Blob.

When you define the semantic category for a data type, you can modify the precision and scale returned by the import process so that the data type matches the requirements of the type system.

Use the following semantic categories to classify the native types:

Length semantics

Use this category for native types where length is the principal characteristic. This category can include data types such as Char, Varchar2, Binary, Varbinary, Blob, and Clob.

Integer semantics

Use this category for native types that can contain signed integers. The length of the data type is the number of decimal digits specified in the data type precision. This category can include data types such as Integer, Smallint, Bigint, and Tinyint.

Machine integer semantics

Use this category for native types that can contain signed or unsigned integers. The length of this data type is measured in bytes. The precision of a machine integer type is the maximum number of decimal digits that fits within the length of the data type, regardless of whether all possible values can be stored. For example, a 32-bit (4 byte) machine integer can store up to 10 digits but if the value of each digit is 9, then the value of the integer can result in an overflow.

Decimal semantics

Use this category for native types that can contain an exact real number where precision is the total number of digits and scale is the number of digits to the right of the decimal point. The precision for this semantic category must be greater than or equal to the scale. This category can include data types such as Decimal and Numeric.

Scientific decimal semantics

Use this category for native types that can contain an exact real number where precision is the number of digits stored rather than the total number of digits represented by the number. The total number of digits represented by the number can exceed the precision. The scale of the data type can exceed precision and can be positive or negative. A positive scale represents digits to the right of the decimal point. A negative scale represents the rounding position to the left of the decimal point. This category can include data types such as the Number data type in Oracle with precision and scale specified.

Float semantics

Use this category for binary or decimal floating point data types. An example of a binary floating point type is the binary_float type in Oracle. An example of a decimal floating point number is the Number data type in Oracle with precision and scale not specified.

Gregorian date semantics

Use this category for date types that the adapter can expose as Gregorian dates, times, and timestamps.

Native Type Properties

Depending on the semantic category, you must set one or more of the following native type properties:

- Maximum precision
- Minimum precision
- Default precision
- Support for changes to precision
- Maximum scale
- Minimum scale
- Default scale
- Support for changes to scale
- Unit of length, such as characters, bytes, and bits

- For Gregorian date semantics, support for hour, minute, second, year, month, day, or time zone
- For float semantics, radix of the precision and exponent

Set the maximum and minimum values for precision and scale to validate the following:

- Imported database metadata
- Data type of a column that an end user adds to a table in a mapping

The default precision, default scale, and the specification of whether the precision and scale can be modified apply to metadata that is manually defined by the user. If the precision or scale cannot be modified, then the default value and the maximum value must be set to the same value.

Informatica Platform Types

Informatica platform types include all the data types that are recognized by Informatica. The Informatica Connector Toolkit API uses ODBC as a model to describe the Informatica platform data types.

In addition to assigning semantic categories to native types, you must match each semantic category with one or more Informatica platform data types. The Informatica platform uses ODBC types to determine what type of data buffers to bind to the run-time adapter for data access. In some cases, the match between a semantic type and an Informatica platform type is not exact. For example, the Informatica platform always binds a timestamp buffer if the semantic category is mapped to an Informatica platform date, time, or timestamp data type.

In many cases, a semantic category matches one of the available Informatica platform types. In cases where there is no exact match between types, select the best possible match.

In addition to defining the best mappings to convert native types to Informatica platform types, you can specify alternate mappings. An alternate mapping allows the end user to select alternate transformation data types to associate with the input and output ports of a data source used in a mapping.

Use the Informatica Connector Toolkit to add matching Informatica platform data types for the semantic category. You can indicate whether the mapping is the exact match or whether the mapping can result in a lossy data conversion due to a match that is not exact.

Note: Informatica platform types are not the same as the transformation data types that appear in a transformation that you add to a mapping. Informatica platform types are internal data types used only within the adapter type system. Informatica can convert Informatica platform types to transformation data types.

CHAPTER 6

Metadata Objects

This chapter includes the following topics:

Metadata Objects Overview

Define the adapter metadata components to represent the metadata of the data source. After you define the metadata components, you can fetch and display the metadata in the Developer tool with information on the native data type, precision, and scale.

Use the Informatica Connector Toolkit to define one or more native metadata definitions for the adapter to read from and write to the data source. The Informatica Connector Toolkit internally uses a metadata model that represents the metadata.

The pattern classes in the metadata model describe the metadata structure. When the Informatica Connector Toolkit publishes the adapter, it registers the metadata model in the Model repository. After the Informatica Connector Toolkit registers the metadata model in the Model repository, you can use the Developer tool to import metadata from the data source.

Metadata Components

The metadata components in the Informatica Connector Toolkit represent the metadata of the data source and the associated operations to read from and write to the data source.

Use the Informatica Connector Toolkit to define the following metadata components:

Native metadata

Define one or more native metadata objects to represent the metadata of the native source. Associate the native metadata objects with operations to read metadata from a data source, look up metadata from a data source, or write metadata to a data source.

Record extensions

You can define record extensions to define additional metadata for records. For example, you can add the attribute query hint as a record extension.

Field extensions

You can define field extensions to define additional metadata for fields. For example, you can add the attribute isNullable as a field extension to indicate that the field can contain a null value.

Procedure Extension

You can define procedure extensions to define additional metadata for procedures.

Parameter extension

You can define parameter extensions to define additional metadata for parameters.

Call Capability

You can define additional attributes to create a read or a write call to a procedure.

Read capability

Define attributes that you can read from the data source. You can also define whether the adapter supports look up of data, join operations, and filter operations when the adapter reads from the data source.

Write capability

Define attributes that you can write to the data source. You can also define whether the adapter supports upsert operation when the adapter writes to the target.

Import Options

Use the Informatica Connector Toolkit to define the import options that appear in the Developer tool when an adapter consumer creates a data object.

The following tables describes the import options:

Option	Description
Allow Multi Select	Allows selection of multiple importable objects. Default is false.
Display Filter By Name	Displays the filter by name option. Default is true.
Display Filter By Description	Displays the filter by description option. Default is true.
Display Filter By Path	Displays the filter by path option. Default is true.
Display Skip Description	Displays the skip descriptions check box. Default is true.
Show Entity	Shows entity details. Default is true.
Show Hierarchy	Shows metadata hierarchy. Default is true.
Show Related Records	Shows related records. Default is true.

CHAPTER 7

Partitioning Capability

This chapter includes the following topics:

Partitioning Capability Overview

You can use the Informatica Connector Toolkit to specify the partition type and implement the partition logic to use when the adapter reads or writes data.

Based on the partition logic you implement for an adapter, the Data Integration Service dynamically divides the underlying data into partitions and processes all of the partitions concurrently.

You can specify the following partitioning types for an adapter:

Dynamic

A partitioning logic that determines the number of partitions at run time based on the partition information from the data source. If the data source provides partition information, you can use dynamic partitioning to increase the performance of adapter read and write operations.

Static

A partitioning logic that is based on the partition information that the user specifies, such as number of partitions or key range. If you require the user to specify the partition information, you can implement fixed partitioning for the adapter. If the tables in the data source support key range partitioning, you can add key range partitioning capability for the adapter.

Dynamic Partitioning

You can use the Informatica Connector Toolkit to configure an adapter to dynamically determine partition information from the data source. Adapters with dynamic partitioning capability do not require partition information from the user when the adapter reads or writes data.

If you implement dynamic partitioning for an adapter, the adapter queries the data source for the number of source or target partitions and other partition-specific attributes. For example, if the data source is a

relational database, you can make use of the partition information from the database to implement the partition logic.

When you define a native metadata object, you can implement dynamic partitioning for both read and write capabilities of the native metadata object.

Static Partitioning

You can use the Informatica Connector Toolkit to configure static partitioning if you require the user to specify the partition information before the adapter reads data.

When you define a native metadata object, you can implement static partitioning for read capabilities of the native metadata object. Based on the data source, you can implement following static partitioning types:

Fixed

If you require partition logic based on the partition information specified by the user, implement fixed partitioning capability. For example, if the data source does not provide partition information, you can implement partitioning logic based on the user inputs. The user enters the partition information, such as the number of partitions, before the adapter reads data from the data source.

Key range

If the tables in the data source support key range partitioning, you can add support for key range partitioning capability. Before you add support for key range partitioning, you must ensure that the adapter supports filter operation and platform expression. The Informatica Connector Toolkit implements key range partitioning as a filter query. The adapter user enters the partition keys and key range when the adapter reads data from the data source.

You can implement static partitioning only for adapters with read capability.

CHAPTER 8

Run-time Behavior

This chapter includes the following topics:

Run-time Behavior Overview

Use the functions in the Informatica Connector Toolkit API to specify the run-time behavior of the adapter. You must write the code to define how the adapter connects, disconnects, reads from and writes to the data source.

The run-time functions are available in C and Java. Within the run-time functions, you can use any API that is appropriate for communicating with the data source. You must implement all the run-time functions.

The run-time functions use character string arguments and character data buffers in UCS-2 format. If the database API communicates with the data source through the UCS-2 character set, then pass the character strings and data buffers directly to the database API. If the database API does not use the UCS-2 character set, you must convert the data to UCS-2 format before you pass the data to the database API.

To reduce complexity, design the adapter so that it does not require the end user to specify any character set information. Use the Unicode mode if the database client API provides a Unicode mode. Or, query the database to determine the correct character set to use when reading or writing data to the database. If you require input from the end user about the correct character set, define a custom connection attribute to store this information.

You can define the adapter run-time behavior to support pre and post commands to perform tasks before and after a mapping run. For example, you can define the adapter run-time behavior to support a pre command that initializes environment variables before the mapping run.

Run-time Java Functions

Extend the following classes to define the adapter run-time behavior with Java functions:

DataConnection

Implement the methods in this class to connect and disconnect from the data source.

DataAdapter

Implement the methods in this class to initialize the data session, deinitialize the data session, begin the data session, end the data session, read data from the data source, and write data to the data source.

AutoPartitioningMetadataAdapter

If the adapter supports partitioning capability, implement the methods in this class to specify the partition type and logic.

OperationAdapter

If the adapter supports partitioning capability, implement the methods in this class to perform any operation before or after the partitioning.

ASOOperationObjMgr

Implement the methods in this class to perform any custom metadata validations.

Run-time C/C++ Functions

Implement the following functions in the runtimeadapter.cpp file to define the adapter run-time behavior with C/C++ interfaces:

- INFAADPInitPlugin
- INFAADPInitDataSession
- INFAADPInitDataSourceOperation. The scope of the RuntimeConfig and Metadata handles available in this method is within the INFAADPInitDataSourceOperation method.
- INFAADPDeinitDataSourceOperation. The scope of the RuntimeConfig and Metadata handles available in this method is within the INFAADPDeinitDataSourceOperation method.
- INFAADPConnect
- INFAADPBegindataSession
- INFAADPRead
- INFAADPReset
  If you defined lookup support for the adapter, implement INFAADPReset. The INFAADPReset method is called after INFAADPRead if the mapping contains a lookup operation.
- INFAADPWrite
- INFAADPDeinitDataSession
  Even if INFAADPInitdataSession returns failure, INFAADPDeinitDataSession is called to deinitialize the data session.
- INFAADPEndDataSession
- INFAADPDisConnect
- INFAADPDeinitPlugin

CHAPTER 9

Adapter Example: Instagram

This chapter includes the following topics:

Instagram Adapter Overview

The Informatica Connector Toolkit includes sample source code of the Instagram adapter. You can use the Instagram adapter sample source code as a model to develop an adapter for a data source. The Instagram sample was developed with the Informatica Connector Toolkit.

Note: The sample Instagram adapter is for illustration purposes only.

Building the Sample Adapter

You can use the sample adapter source code to develop an adapter for a data source. Import the sample adapter project in to the Eclipse IDE and use the Informatica Connector Toolkit to publish or deploy the sample adapter.

To use the sample source code and develop an adapter, perform the following tasks:

1. From the Eclipse IDE, click **File** > **Import**. The **Import** dialog box appears.
2. Select **Existing Projects into Workspace** and then click **Next**. The **Import Projects** page appears.
3. Select **Select root directory** and browse to the directory of the sample adapter that you want to import into Eclipse.
4. Click **Finish**.
   The sample adapter project appears in the package explorer.
5. Change to the **Informatica Connector** perspective.
6. Edit the connection, types system, metadata, and run-time components, if required.
7. Publish the sample adapter or deploy the sample adapter on local Informatica services and client.

For more information, see the *Sample Adapter Readme*.

CHAPTER 10

Adapter Example: MySQL

This chapter includes the following topics:

MySQL Adapter Overview

The Informatica Connector Toolkit includes sample source code of the MySQL adapter. You can use the MySQL adapter sample source code as a model to develop an adapter for a data source. The MySQL sample was developed with the Informatica Connector Toolkit.

Note: The sample MySQL adapter is for illustration purposes only.

MySQL Adapter Requirements

To run the sample adapter, you must install the MySQL JDBC driver on your development machine. Use the MySQL JDBC driver to access the metadata and perform read and write operations on the MySQL database. Download MySQL JDBC driver version 5.1.26 or later from the following URL:

```
http://dev.mysql.com/downloads/connector/j
```

Building the Sample Adapter

You can use the sample adapter source code to develop an adapter for a data source. Import the sample adapter project in to the Eclipse IDE and use the Informatica Connector Toolkit to publish or deploy the sample adapter.

To use the sample source code and develop an adapter, perform the following tasks:

1. From the Eclipse IDE, click **File** > **Import**. The **Import** dialog box appears.
2. Select **Existing Projects into Workspace** and then click **Next**. The **Import Projects** page appears.

3. Select **Select root directory** and browse to the directory of the sample adapter that you want to import into Eclipse.
4. Click **Finish**.
   The sample adapter project appears in the package explorer.
5. Change to the **Informatica Connector** perspective.
6. Edit the connection, types system, metadata, and run-time components, if required.
7. Publish the sample adapter or deploy the sample adapter on local Informatica services and client.

For more information, see the *Sample Adapter Readme*.

MySQL Adapter Components

The sample MySQL adapter includes the following components:

Contribution

Use the contribution plug-in project to get information on plug-ins contributing to the MySQL adapter project. The name of the MySQL contribution plug-in is `com.infa.adapter.mysql.adapter.contribution`.

Connection model

Use the connection model Java project to represent the connection model for the MySQL adapter. The name of the MySQL connection model Java project is `com.infa.products.adapters.mysql.models.connection.annotatedjava`.

Connection adapter

Use the connection adapter plug-in project to provide connection attribute information and consumer information for the MySQL adapter. The name of the MySQL connection attributes plug-in is `com.infa.products.adapter.mysql.connection.adapter`.

Seed provider

Use the seed provider Java project to map native data types to Informatica data types. The name of the MySQL seed provider Java project is `com.infa.adapter.mysql.seedprovider`.

Type system

Use the type system plug-in project to contribute the seed provider of the adapter to the Informatica platform. The name of the MySQL type system plug-in project is `com.infa.products.adapter.mysql.typesystem`.

Metadata model

Use the metadata model Java project to represent the metadata model for the MySQL adapter. The name of the MySQL metadata model Java project is `com.infa.products.adapters.mysql.models.metadata.annotatedjava`.

Metadata adapter

Use the metadata adapter Java project to provide the functionality to open and close connections to the MySQL adapter. The name of the MySQL metadata adapter Java project is `com.infa.products.adapter.mysql.metadata.adapter`.

Run-time model

Use the run-time model Java project to represent the run-time model for the MySQL adapter. The name of the MySQL run-time model java project is `com.infa.products.adapters.mysql.models.runtime.annotatedjava`.

Run-time adapter

Use the run-time adapter plug-in project to implement run-time adapter for the MySQL adapter in Java. The name of the MySQL run-time adapter plug-in project is `com.infa.products.adapter.mysql.runtime.adapter`.

Run-time adapter C

Use the C run-time adapter C++ project to implement run-time adapter for the MySQL adapter in C++. The MySQL C++ run-time adapter project is a Visual Studio 2012 C++ project.

Metamodel bundle

Use the MySQL metamodel bundle plug-in project to list the metaclasses for adapter packages such as connection, metadata, run-time ASO, and run-time capability. The name of the MySQL metamodel bundle plug-in project is `com.infa.products.adapter.mysql.metamodel`.

Model attributes

Use the model attributes plug-in project to define the presentation labels for the field, record, and run-time extensions. The name of the MySQL model attributes plug-in project is `com.infa.products.adapter.mysql.modelAttributes`.

Design-time messages

Use the messages design plug-in project to implement design-time messages for the MySQL adapter in Java. The name of the MySQL messages design plug-in project is com.infa.products.adapter.mysql.messages.design.

Run-time messages

Use the messages runtime plug-in project to implement run-time messages for the MySQL adapter in Java. The name of the MySQL messages runtime plug-in project is com.infa.products.adapter.mysql.messages.runtime.

Library information

Use the library info plug-in project to define the run-time adapter based on the programming language in which you implement the run-time adapter. Currently, you can use only Java interfaces to implement the run-time adapter. The name of the MySQL library Info plug-in project is `com.infa.products.adapter.mysql.libraryInfo`.

License

Use the license Java project to perform license checks for the MySQL adapter. The name of the MySQL license Java project is `com.infa.products.adapter.mysql.license`.

UI wizard

Use the UI wizard project to define the icons for the Import wizard. The UI wizard project is a Java project. The Developer tool uses the import options to display UI components when an adapter consumer creates a data object. The name of the UI wizard project is `com.infa.products.adapter.mysql.wizard`.

Feature

Use the feature plug-in project, which is an Eclipse feature, to define an adapter plug-in. The name of the MySQL feature plug-in project is `com.infa.products.adapter.mysql.feature`.

CHAPTER 11

Adapter Example: YouTube

This chapter includes the following topics:

YouTube Adapter Overview

The Informatica Connector Toolkit includes sample source code of the YouTube adapter. You can use the YouTube adapter sample source code as a model to develop an adapter for a data source. The YouTube sample was developed with the Informatica Connector Toolkit.

Note: The sample YouTube adapter is for illustration purposes only.

YouTube Adapter Requirements

To run the sample adapter, you must install the following software on your development machine:

Json-smart library

Use the json-smart library to parse data from YouTube.

Download JSON-smart version 1.1.1 from the following URL:

```
http://code.google.com/p/json-smart/
```

GData client library

Use the GData client library for to read and write data from YouTube. Download GData client library from the following URL:

```
https://developers.google.com/gdata/articles/java_client_lib
```

YouTube Adapter Components

The sample YouTube adapter includes the following components:

Contribution

Use the contribution plug-in project to get information on plug-ins contributing to the YouTube adapter project. The name of the YouTube contribution plug-in is `com.infa.adapter.youtube.adapter.contribution`.

Connection model

Use the connection model Java project to represent the connection model for the YouTube adapter. The name of the YouTube connection model Java project is `com.infa.products.adapters.youtube.models.connection.annotatedjava`.

Connection adapter

Use the connection adapter plug-in project to provide connection attribute information and consumer information for the YouTube adapter. The name of the YouTube connection attributes plug-in is `com.infa.products.adapter.youtube.connection.adapter`.

Seed provider

Use the seed provider Java project to map native data types to Informatica data types. The name of the YouTube seed provider Java project is `com.infa.adapter.youtube.seedprovider`.

Type system

Use the type system plug-in project to contribute the seed provider of the adapter to the Informatica platform. The name of the YouTube type system plug-in project is `com.infa.products.adapter.youtube.typesystem`.

Metadata model

Use the metadata model Java project to represent the metadata model for the YouTube adapter. The name of the YouTube metadata model Java project is `com.infa.products.adapters.youtube.models.metadata.annotatedjava`.

Metadata adapter

Use the metadata adapter Java project to provide the functionality to open and close connections to the YouTube adapter. The name of the YouTube metadata adapter Java project is `com.infa.products.adapter.youtube.metadata.adapter`.

Run-time model

Use the run-time model Java project to represent the run-time model for the YouTube adapter. The name of the YouTube run-time model java project is `com.infa.products.adapters.youtube.models.runtime.annotatedjava`.

Run-time adapter

Use the run-time adapter plug-in project to implement run-time adapter for the YouTube adapter in Java. The name of the YouTube run-time adapter plug-in project is `com.infa.products.adapter.youtube.runtime.adapter`.

Metamodel bundle

Use the YouTube metamodel bundle plug-in project to list the metaclasses for adapter packages such as connection, metadata, run-time ASO, and run-time capability. The name of the YouTube metamodel bundle plug-in project is `com.infa.products.adapter.youtube.metamodel`.

Model attributes

Use the model attributes plug-in project to define the presentation labels for the field, record, and run-time extensions. The name of the YouTube model attributes plug-in project is `com.infa.products.adapter.youtube.modelAttributes`.

Library information

Use the library info plug-in project to define the run-time adapter based on the programming language in which you implement the run-time adapter. Currently, you can use only Java interfaces to implement the run-time adapter. The name of the YouTube library Info plug-in project is `com.infa.products.adapter.youtube.libraryInfo`.

License

Use the license Java project to perform license checks for the YouTube adapter. The name of the YouTube license Java project is `com.infa.products.adapter.youtube.license`.

UI wizard

Use the UI wizard project to define the icons for the Import wizard. The UI wizard project is a Java project. The Developer tool uses the import options to display UI components when an adapter consumer creates a data object. The name of the UI wizard project is `com.infa.products.adapter.youtube.wizard`.

Feature

Use the feature plug-in project, which is an Eclipse feature, to define an adapter plug-in. The name of the YouTube feature plug-in project is `com.infa.products.adapter.youtube.feature`.

Building the Sample Adapter

You can use the sample adapter source code to develop an adapter for a data source. Import the sample adapter project in to the Eclipse IDE and use the Informatica Connector Toolkit to publish or deploy the sample adapter.

To use the sample source code and develop an adapter, perform the following tasks:

1. From the Eclipse IDE, click **File** > **Import**. The **Import** dialog box appears.
2. Select **Existing Projects into Workspace** and then click **Next**. The **Import Projects** page appears.
3. Select **Select root directory** and browse to the directory of the sample adapter that you want to import into Eclipse.
4. Click **Finish**.
   The sample adapter project appears in the package explorer.
5. Change to the **Informatica Connector** perspective.
6. Edit the connection, types system, metadata, and run-time components, if required.
7. Publish the sample adapter or deploy the sample adapter on local Informatica services and client.

For more information, see the *Sample Adapter Readme*.

APPENDIX A

Metadata Models

This appendix includes the following topics:

Metadata Model Overview

The Informatica Connector Toolkit internally uses metadata models to define native metadata objects for the adapter to read from and write to the data source. The metadata model contains components that represent the metadata of the data source.

The Informatica Connector Toolkit uses pattern classes in the metadata model to describe the metadata structure of the data source. The metadata model uses a pattern template to define the metadata pattern of an adapter.

After the Informatica Connector Toolkit registers the metadata model of the data source in the Model Repository Service, you can use the Developer tool to import metadata from the data source.

Note: Do not use the metadata models directly to define the native metadata objects for the adapter. Use the Informatica Connector Toolkit to define native metadata objects.

Metadata Model Components

The metadata model contains components that represent the metadata of the data source.

The metadata model consists of the following components:

Flat Record

A flat record represents a structure that contains columns, unique keys, and primary keys. The structure of a flat record is similar to a database table that contains columns and keys.

A flat record contains attributes that store the following information:

- Name of the native metadata object
- Type of native metadata object

- Related records for the flat record
- Primary key for the flat record
- Unique keys for the flat record
- Indexes for the flat record
- Any additional record attributes specific to the data source

Field

A field is a data structure for a single unit of data in a data source.

A field contains attributes that store the following information:

- Name of the field
- Default value of the field
- Precision of the field
- Scale of the field
- Boolean value that indicates whether the field can contain a null value
- Any additional field attributes specific to the data source

Constraints

Constraints represent the primary key and unique keys for a flat record.

The primary key and unique key contain attributes that store the following information:

- Name of the key
- Native name of the key defined in the native metadata
- List of fields that form the key

Index

Index represents a native index that orders the flat records or uniquely identifies a row in the flat record.

An index contains attributes that store the following information:

- Name of the index
- Native name of the index
- Boolean value that indicates whether the index is unique
- List of index fields
- Index order to retrieve the data

The Informatica Connector Toolkit internally uses the metadata model components to represent the data source metadata and persists the metadata in the Model repository.

Metadata Patterns

The metadata pattern of an adapter describes the metadata structure of the native data source. You can choose a metadata pattern template to define the metadata pattern of an adapter.

A metadata pattern template is a set of packages such as catalog, container, flat record, and so on. Sets of related classes that define the metadata pattern are organized as packages.

Though an adapter can have more than one metadata pattern, the Informatica Connector Toolkit currently supports only Type A template. The type A template provides the metadata patterns for flat records. You can use the metadata catalog that the Type A template provides to describe the metadata structure of the data source.

Features of Type A Metadata Template

The Informatica Connector Toolkit uses the Type A metadata pattern template to define the metadata pattern for flat records.

The Type A metadata pattern template supports the following features:

Catalog

A catalog is a container for metadata objects that you can import.

Container

A container is a collection of packages.

Package

A package contains flat records. A package may also contain other packages.

Flat record

A flat record represents the structure of the native metadata object that you can import. The flat record structure contains fields, relationships, index, and constraints.

Field

A field is a data structure for a single piece of data in a data source.

Record relationship

A record relationship represents the relationship between flat records.

Index

An index represents a native index that orders the flat records or uniquely identifies a flat record.

Constraints

A constraint represents the unique key and primary key for a flat record.

APPENDIX B

ASO Model

This appendix includes the following topics:

ASO Model Overview

The Informatica Connector Toolkit internally uses an Adapter Specific Object (ASO) model to represent operations on native metadata objects.

The ASO model serves as a container for native metadata objects and associates operations such as read, write, or lookup with the native metadata objects. A native metadata object represents the native importable metadata of a data source such as a flat file, relational data source, and nonrelational data source. An ASO model contains references to multiple native metadata objects and operations.

Note: Do not use the ASO models directly to define operations on native metadata objects. Use the Informatica Connector Toolkit to define operations on native metadata objects.

ASO Model Components

Use the Informatica Connector Toolkit to define the components that comprise the application specific object.

The ASO model consists of the following components:

ASO

The ASO object represents the generic adapter specific object. The ASO object contains references to the following components that comprise the ASO object.

- ASO Operation
- Catalog
- Projection

ASO Operation

The ASO operation object contains references to capabilities, capability attributes, and complex types that you associate with an operation.

Catalog

The catalog object represents the native metadata object. The native metadata object is the native importable metadata of a data source. The catalog object contains references to components of native metadata objects such as columns, unique keys, and primary keys.

Projection

The projection object represents the operations that you can perform when you read or write data. The projection object contains references to the list of operations, projection type, and basic projection view.

ASO Projections

An ASO projection is a sequence of operations that you perform on data when you read data from a data source or write data to a data source.

Use the Informatica Connector Toolkit to define a basic ASO projection model or an advanced projection model according to your requirements. The basic ASO projection model provides a simple view of the advanced projection model. The Informatica Connector Toolkit uses the BasicProjectionView interface to provide a simple and basic model to the adapter developer. Use the get methods in the BasicProjectionView interface to get information on native metadata object, platform types, scale, precision, and conditions like filter and join.

To define the advanced ASO projection model, define the following projection operation interfaces as required.

Sink operation

There are two types of sink operations: native sink operation and platform sink operation. The native sink operation gets data from the platform operation when you write to a data source. The platform sink operation gets data from the native source operation when you read from a data source.

Source operation

There are two types of source operations: native source operation and platform source operation. The native source operation inputs the native data to the platform sink operation in a read projection operation. The platform source operation inputs the platform data to the native sink operation in a write projection operation.

Join operation

Use the join operation to join data from two related sources or from the same source.

Filter operation

Use the filter operation to filter data based on one or more conditions.

Projection operation

Use the projection operation to select a subset of attributes to rename or to drop fields.

For more information about interfaces that define the ASO projection model, see the *Informatica Connector Toolkit API reference documentation*.

APPENDIX C

Adapter Project Migration

This appendix includes the following topics:

Adapter Project Migration Overview

You can run the `ict.bat` file to migrate an adapter project from Windows platform to UNIX or AIX platform.

The following table lists the available command options that you can use to migrate the adapter project and deploy the adapter:

Command Option	Description
`generateCode`	Generates the adapter code and `build.xml` using the configurations defined in `ict_metadata.xml`.
`buildAdapter`	Compiles and deploys the adapter project.
`backupProject`	Creates a backup file for the adapter project.
`restoreProject`	Restores the project by using the project backup file.

The ICT command uses the following syntax:

- On Windows, `ict.bat <option> <argument1> [argument2] [...]>`
- On UNIX, LINUX and AIX, `ict.sh <option> <argument1> [argument2] [...]>`

Migrating the Adapter Project from Windows Platform to other Platforms

Before you migrate the adapter project, verify that the project is available on the Windows platform.

1. Open the command prompt on the Windows platform and navigate to `%ICT_HOME%\tools\scripts`.

2. Run `ict.bat backupProject <source_dir> <destination_dir\FileName.zip>` to back up the adapter project.

   Specify the following arguments:
   `source_dir`

   The source directory of your adapter project .

   `destination_dir`

   The directory where you want to save the backup file.

   `FileName`

   File name for the backup file.

   The `<FileName>.zip` backup file is created in the specified directory.
3. Copy the `<FileName>.zip` file from the Windows platform to the target platform.
4. To restore the adapter project, run `ict.bat restoreProject <source_dir> <destination_dir \FileName.zip>` from the target platform command prompt.

   Specify the following arguments:
   `source_dir`

   The path of your `<FileName>.zip` file.

   `destination_dir`

   The directory where you want to save the adapter project.

   `FileName`

   File name for the extracted adapter project.
5. Set the environment variables `ICT_HOME`, `ECLIPSE_HOME`, `JAVA_HOME` and `ANT_HOME` on the target platform.
6. Run `ict.bat generateCode <source_dir>` to generate the adapter code and the `build.xml` file.

   Specify the path of the adapter project in `<source_dir>`.
7. Run `ict.bat buildAdapter <source_dir>` to compile and deploy the adapter project.

   Specify the path of the adapter project in `<source_dir>`.

INDEX