CA Identity Manager

Connector Xpress Guide

r12.5 SP2

CA Product References

This document references the following CA products:

- CA Directory
- CA Identity Manager
- eTrust Admin

Contact CA

Contact Technical Support

For your convenience, CA provides one site where you can access the information you need for your Home Office, Small Business, and Enterprise CA products. At http://ca.com/support, you can access the following:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Provide Feedback

If you have comments or questions about CA product documentation, you can send a message to techpubs@ca.com.

If you would like to provide feedback about CA product documentation, complete our short customer survey, which is also available on the CA Support website, found at http://ca.com/docs.

Contents

Chapter 1: Connector Xpress

This section contains the following topics:

Connector Xpress Overview

Connector Xpress is a CA Identity Manager utility for managing dynamic connectors, mapping dynamic connectors to endpoints, and establishing routing rules for endpoints. You can use it to configure dynamic connectors to allow provisioning and management of SQL databases and LDAP directories.

Connector Xpress lets you create and deploy custom connectors without the technical expertise required when creating connectors managed by the Provisioning Manager.

You can also set up, edit, and remove a connector server configuration (both Java and C++) using Connector Xpress.

The primary input into Connector Xpress is the native schema of an endpoint system. For example, you can use Connector Xpress to connect to an RDBMS and retrieve the SQL schema of the database. You can then use Connector Xpress to construct mappings from those parts of the native schema that are relevant to identity management and provisioning. A mapping describes how the provisioning layer represents an element of the native schema.

Connector Xpress generates metadata that describes, to a dynamic connector, the runtime mappings to a target system.

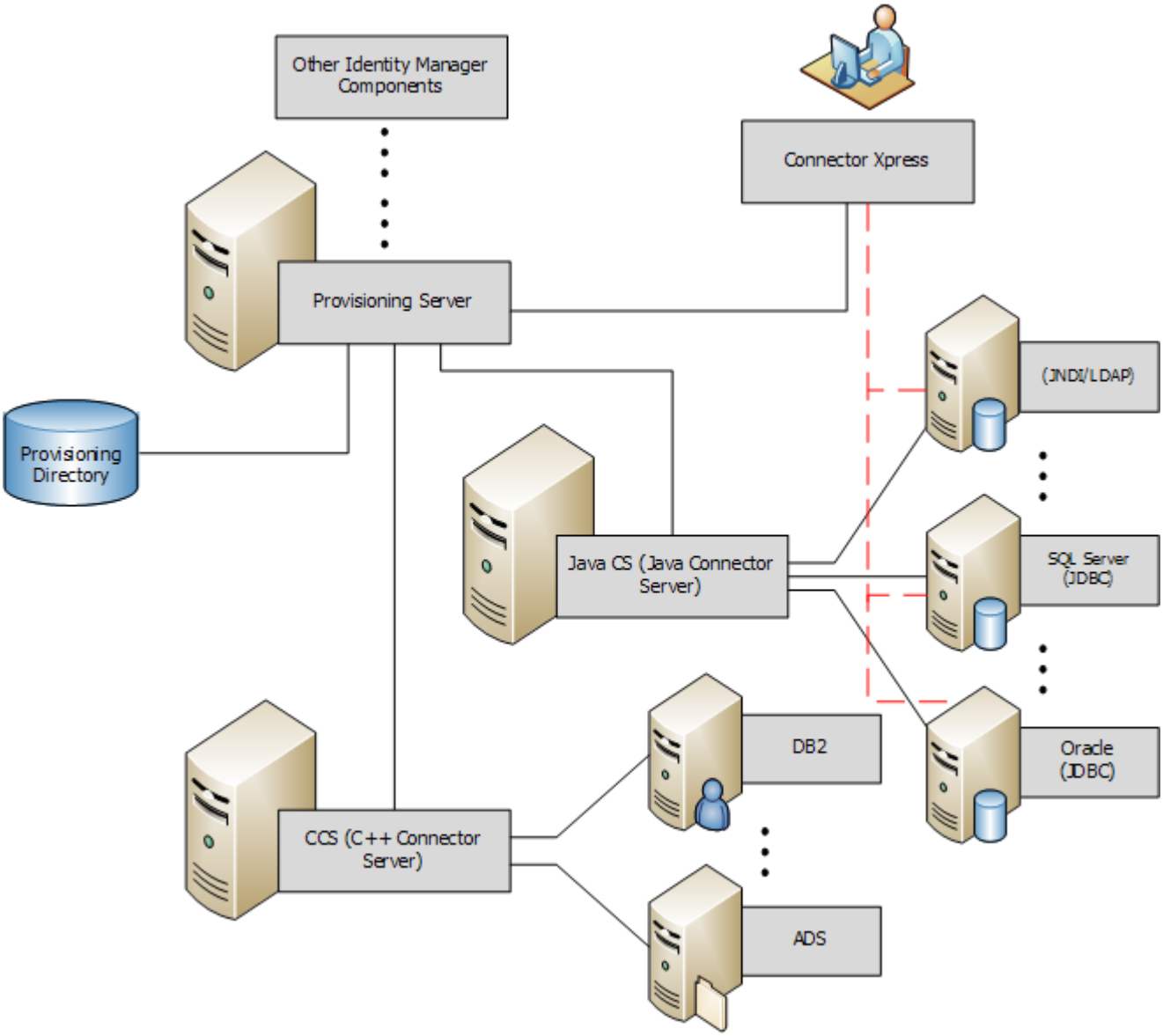
The output of Connector Xpress is a metadata document produced when you complete your mappings. The metadata is an XML file that describes the structure of your connector to the Java CS.

The output describes the provisioning server classes and attributes and how they are mapped to the native schema.

The metadata is used to create dynamic endpoint types on one or more Provisioning Servers.

Communication with Other Components

The following diagram shows the direct and indirect communications between Connector Xpress and other components in the distributed system:



Note: The dashed red lines indicate that the direct communications between Connector Xpress and endpoints are read-only. These direct communications occur when setting up data sources.

Supported Directories

Connector Xpress supports the following directories:

- CA Directory
- Sun One Directory
- Novell Directory
- Oracle Internet Directory (OID) 10.x
- ADAM for 2003
- LDS for 2008

Note: For the up-to-date list of supported versions, see the CA Identity Manager support matrix on the CA Support Site.

Supported Databases

Connector Xpress supports the following databases:

- Oracle
- Microsoft SQL Server
- Ingres
- DB2
- DB2 for z/OS
- DB2 for IBM i
- Sybase
- MySQL
- Informix

  Note: For the up-to-date list of supported versions, see the CA Identity Manager support matrix on CA Support.

Data Sources

A data source is a reference to a server or other source of schema information that Connector Xpress can construct a connector for.

Connector Xpress supports the following data sources from supported vendors:

- **JDBC**—Any service accessible through the JDBC API, typically a relational database server.
- **JNDI**—Any service accessible through the JNDI API, typically an LDAP server.

Set Up Data Sources

You can connect to a data source to read its schema and then use Connector Xpress to map the schema to a new Provisioning Server endpoint type. Connector Xpress maintains a list of data sources that you configure. You can set up JNDI or JDBC data sources.

Note: When you create a Connector Xpress project, the Connector Xpress prompts you to set up your data sources.

To set up data sources

1. Click Tools, Data Sources.

   The Select Data Source for new Project dialog appears.

2. Click Add.

   The Source Types dialog appears.

3. Select an available data source type from the list, then click OK.

   The Edit Source dialog appears, specific to the type of data source you are adding.

4. Complete the fields on the dialog, then click Test to verify your settings and authentication details.

   Depending on the authentication method chosen, a subsequent authentication dialog can appear.

   Complete the fields on this dialog to continue the test.

5. When the test has completed successfully, click OK.

   You have successfully configured the data source type you selected.

Password Caching

Connector Xpress caches the passwords used with each Provisioning Server only when the application is open. When the application is closed, Connector Xpress does not permanently record passwords. When Connector Xpress needs a password, and the password is not cached, or the cached password causes a failed authentication attempt, Connector Xpress prompts you to enter the password.

Connector Xpress Window

The Connector Xpress window contains the following three panes:

Mapping tree

Displays an overview of the mappings you made for your endpoint. Each node and child node represent an aspect of your endpoint mapping. For example, attributes you have mapped, associations between classes, and operation bindings for your endpoint. Clicking a node displays a dialog that allows you to edit the parameters of the node.

Editor panel

Displays a dialog that lets you edit the parameters for the node you have selected.

Provisioning Servers tree

Displays the endpoints and connectors of multiple Provisioning Servers.

Mapping Tree

The Mapping Tree lets you define the mappings, associations, and operation bindings for your connector. The mapping tree contains the following nodes:

Endpoint type node

Lets you create a description of your endpoint.

Classes parent node

Lets you map a new class and view a read-only list of the classes that you have mapped and the native classes they are mapped to.

Class node

Lets you specify the JDBC database tables or the JNDI native object classes that you want map a class too.

Attributes parent node

Lets you logically group the attributes you have mapped into groups and subgroups that you want to appear as tabs and pages in the CA Identity Manager User Console account screens.

Attributes node

Lets you configure extended details of the selected attribute.

Associations parent node

Lets you specify the classes you want to create direct and indirect association with.

Containers parent node

Lets you specify the object classes which you want to use as containers in your connector.

Custom Types

Lets you define arbitrary strings (flexi-strings) you want to use as metadata types in attribute mappings.

Operation Bindings

Lets you specify operation binding information and display summary of existing operation bindings.

Scripts parent node

Lets you name the script you want to bind to the operation bindings you have mapped.

Scripts node

Lets you specify the parameters of the script you want to bind to the operation bindings you have mapped.

Provisioning Servers Tree

The Provisioning Servers tree (in the lower left pane) lets you manage the endpoints and connectors of multiple Provisioning Servers. The Provisioning Servers tree contains the following fields:

Refreshes the currently selected node in the Provisioning Tree.

Note: The Refresh button is only available when you select a tree node that has a changeable list of child nodes. For example, the button is not available for a particular endpoint, as this type of node always has Policies and Endpoint child nodes only. For the endpoints node of a particular endpoint, the button is available.

Displays the actions you can perform on the currently selected object in the Provisioning Tree. Clicking this button is equivalent to right-clicking the currently selected object in the Provisioning Tree.

Provisioning Servers node

Lets you add and configure Provisioning Server connection details. When you add a Provisioning Server, Connector Xpress maintains the details of the Provisioning Server so that you can access it each time you run Connector Xpress.

Servers Object node

Displays each registered Provisioning Server on the network. You can use this node to edit or remove the Provisioning Server connection details.

Domains Node

Displays the domain configured for the Provisioning Server. For example, when you have configured multiple alternative provisioning servers for a single domain in an eTrust Admin 8.1 deployment.

Endpoint Types node

Displays a list of endpoint types. You can use this node to create an endpoint type.

Endpoints Object node

Lets you do the following:

- Deploy and edit metadata
- Delete an endpoint type
- Clean an endpoint type for DYN endpoint types
- Set the managing connector server only for non-DYN endpoint types

Endpoints node

Groups endpoints of a particular endpoint type.

Endpoints Name node

Displays an endpoint that you have acquired on the Provisioning Server. You can use this node to:

- Perform an explore or correlate of the endpoint for DYN endpoints
- Delete an endpoint
- Set the managing connector server only for non-DYN endpoints

Policies node

Displays groups of policies of a particular endpoint type.

Note: A default policy is provided for all endpoint types.

CS Config Object node

Displays a list of CS configurations.

You can use this node to:

- Edit the existing configuration for the server's branch DN routing rules
- Delete the configuration entry for the C++ Connector Server or Java Connector Server
- Set the managing connector server password

CS Configs node

Displays logical groupings of C++ Connector Server and Java CS configurations. These configurations contain the DSFConfig style routing rules for managing branch DNs. You can use this node to add a new routing rules configuration object for a C++ Connector Server or Java CS service.

Chapter 2: Installing Connector Xpress

This section contains the following topics:

Installation Prerequisites

The following are the prerequisites for installing Connector Xpress;

- Identity Manager Server
- CA Identity Manager Provisioning Server

Install Connector Xpress

You do not need to install Connector Xpress on the same computer as the Java CS or any other server component, including the Provisioning Server.

Important! We recommend that you disable all antivirus software before installing Java CS, Java CS SDK, and Connector Xpress.

To install Connector Xpress

1. Locate the CA Identity Manager installation download or other media.
2. Extract the product files (ZIP or TAR).
3. Navigate to the following subfolder and double-click the setup file.
   Provisioning\ConnectorXpress
4. Follow the onscreen instructions to complete the installation.

Install Connector Xpress Silently

Connector Xpress supports a silent mode of installation. To run a silent install, it is necessary to create a response file.

To install Connector Xpress silently

1. In a command window, navigate to where you extracted the Connector Xpress files, then to the following subfolder:

   Provisioning/ConnectorXpress

2. Do either of the following depending on whether you want to install the component as you create the response file, or whether you want to create a response file template to use with a silent install at a later time:

   - To create a response file and install the component at the same time, enter the following command and then run through the installation:

     setup -options-record *install_response_file*

   - To create a response file, but not install the component, enter the following command and then enter the required values in the template:

     setup –options-template *install_response_file*

   Note: Use fully qualified path names when generating and running response files. For example, responsefile.txt is not valid but C:\responsefile.txt is valid.

3. Start the silent installation using the following command:

   setup –options *install_response_file* -silent

Uninstall Connector Xpress Silently

Connector Xpress supports a silent mode of uninstallation. To run a silent uninstall you must create a response file.

To uninstall Connector Xpress silently

1. In a command window, navigate to the uninstall directory which is one of the following:

   - Windows

     C:\Program Files\CA\Identity Manager\Connector Xpress_uninst

   - UNIX

     /opt/CA/IdentityManager/ConnectorXpress/_uninst

2. Do one of the following depending on whether you want to uninstall the component as you create the response file, or whether you want to create a response file template to use with a silent uninstall at a later time.

   - To create a response file and uninstall the component at the same time, enter the following command and then run through the uninstallation:

     uninstaller -options-record *uninstall_response_file*

   - To create a response file, but not install the component, enter the following command and then enter the required values in the template:

     uninstaller –options-template *uninstall_response_file*

   Note: Use fully qualified path names when generating and running response files. For example, responsefile.txt is not valid but C:\responsefile.txt is valid.

3. Start the silent installation using the following command:

   uninstaller –options *uninstall_response_file* -silent

Upgrading Metadata

Connector Xpress automatically upgrades your existing metadata when you open your old project into the new version of Connector Xpress. As you take advantage of the new features in this release, they are automatically added to your existing metadata.

Note: When you open an old project for the first time in Connector Xpress, redeploy the metadata (see page 89) and generate the CA Identity Manager User Console Account Screens (see page 62).

Where User Preferences Are Stored

Connector Xpress stores user preferences through the Java Preferences API. On Windows, this stores data in the following registry key:

HKEY_CURRENT_USER\Software\JavaSoft\Prefs

On UNIX, this stores data in the users home directory under the following folder:

.java/.userPrefs

Note: The uninstaller does not remove this data, so subsequent installs preserve user preferences.

Set Preferences

You can change various aspects of the way Connector Xpress looks and behaves, such as its look and feel, search limits, and command parameters.

To set preferences

1. Click Tools, Preferences.

   The Connector Xpress Preference dialog appears.

2. Compete the fields on the dialog, then click OK.

   Connector Xpress applies and saves your preferences.

More Information:

Preferences Dialog (see page 159)

Where Settings Are Stored

Settings are stored using the Java Preferences API. The API stores the values in the registry on Windows and in a file under the user's home directory on Solaris.

On Windows, the value is stored in the following registry key:

HKEY_CURRENT_USER\Software\JavaSoft\Prefs\com\ca\iam\conman

Chapter 3: Creating Connectors

This section contains the following topics:

Projects

Connector Xpress lets you create projects that specify how connectors are configured and deployed to the Provisioning Server.

A project represents a connector in Connector Xpress. A project includes:

- The schema retrieved from a data source.
- The configuration of metadata that describes how Connector Xpress maps the schema to the endpoint.

You can save, edit, and reopen projects for later use.

You can also create projects based on existing metadata, then modify the class mappings and operation bindings and then redeploy the connector back to the endpoint.

How you Create a Connector Xpress Project

You can create a project using the following methods:

- Create a project specifying a JNDI or JDBC data source (see page 26)

  This lets you create projects that specify how connectors are configured and deployed to the Provisioning Server.

- Create a project without specifying a data source

  This allows you to use Connector Xpress as a generic metadata editor (see page 90) and create new attributes (see page 91).

- Create a project based on a template (see page 27)

  This lets you use templates as starting points for mapping common endpoint schemas.

- Create a project based on a template using wizard mode (see page 28)

  The wizard helps you start a new project based on a template, and steps you through the basic process of mapping an account and group class.

Templates

You can use a template as a starting point for mapping common endpoint schemas. For example, starting a project with a template is useful for LDAP, where standards such as RFCs define widely used schemas, and purely custom schemas are less common. Templates also include information that CA Identity Manager uses to render account management screens.

The following table highlights some of the more commonly used templates. For a full list of all templates, select Project, Create New from Template in Connector Xpress, then click on each template for a brief description.

Important! To help ensure correct performance, we recommend that you use the vendor-specific template supplied with Connector Xpress as a starting point when you create a mapping for a specific vendor.

Project Name Setting	Endpoint type	Description	Metadata File Name
JNDI NIS NetGroup	JNDI	For use with LDAP endpoints supporting NIS Netgroup Schema. This template demonstrates advanced association handling.	jndi_assoc_nisnetgroup_metadata
JNDI inetOrgPerson (Common)	JNDI	LDAP inetOrgPerson. This template should be used when no vendor-specific template is required.	jndi_inetorgperson_common_metadata
Lotus Notes Domino	JNDI	Lotus Notes Domino Server. This template allows easy mapping of eTLNDCustomAttribute* and eTLNDCustomCapabilityAttribute* attributes (the latter set are relevant for account template synchronization).	lnd_metadata

Project Name Setting	Endpoint type	Description	Metadata File Name
SDK DYN Compound	Any	Like SDKDYN but demonstrates use of Compound Values. This template uses compound values which allow complex data to be represented as a single string in JSON syntax, for instance '{"attr1": 42, "attr2": ["a", "b"], attr3: { "objName" : "jack" } }' represents a top level object with three attributes, the first is an integer (42), the next is an array of strings and the last a nested object.	sdkcompound_metadata
SDK DYN	Any	Software Development Kit demo connector. This template is a flat (i.e. non-hierarchical) case-sensitive connector that uses the recommended eTDYN* schema to save provisioning information to local files on the JCS host computer. Because it is flat, its containers are Virtual Containers not actually stored on the endpoint.	sdkdyn_metadata
SDK DYN Script	Any	Like SDKDYN but implemented in Java Script. This template demonstrates how to implement an entire connector (all operation bindings) in JavaScript, as well as configuration information usually found in a connector.xml file, using the connectorXML metadata setting on the top-level namespace.	sdkscript_metadata
SDK DYN UPO Script	Any	Like SDK DYN Script but sends emails rather than writing to local files. This connector has similar	sdkuposcript_metadata

Project Name Setting	Endpoint type	Description	Metadata File Name
		functionality to the deprecated C++ UPO connector except that it sends emails rather than writing information to local files.	

Template Locations

All template files are stored in the *Connector Xpress install directory*/conf/templates.

Connector Xpress searches this directory recursively to find all applicable templates.

Create a Project

To create a connector, you can create projects that specify how connectors are configured and deployed to the Provisioning Server.

To create a project

1. Click Project, New.

   The Select Data Source for new Project dialog appears.

2. If the data source you want to use is displayed in the list, then go to step 4.

3. If the data source you want to use is not displayed in the list, do the following:

   a. Click Add.

      The Source Types dialog appears.

   b. Select an available data source type from the list, then click OK.

      The Edit Source dialog appears, specific to the type of data source you are adding.

   c. Complete the fields on the dialog to configure the data source type you selected, then click OK.

      The Select Data Source for new Project dialog appears and the data source you added appears in the list.

4. Select the data source you want to use from the list, then click OK.

   The Enter Password for Data Source Dialog appears.

5. Enter the password for the selected data source, then click OK.

   Connector Xpress does the following:

   - Displays a Mapping Tree containing a top-level Endpoint Type node.
   - Creates a class named User Account by default in the Mapping Tree.
   - Displays the Endpoint Type Details (see page 129) dialog.

6. Create a mapping to create the appropriate endpoint data.

7. Click Project, Save, or Save As.

   The Save Project As dialog appears.

8. Specify the folder to save the connector project in, the file name and the file type for the saved connector, and click Save.

   Connector Xpress saves the project.

   Note: We recommend that you use the default file extension (.con) and that you keep a copy of the project. You cannot view the stored procedure bindings after you deploy a connector, unless you open the Connector Xpress project file where your bindings are stored.

More Information:

Select Data Source for new project Dialog (see page 164)
Source Types Dialog (see page 165)
Projects (see page 23)

Create a Project Using a Template

You can start a new project using a template as a starting point for mapping common endpoint schemas.

To create a project using a template

1. On the Project menu, click Create New from Template.

   The Select Template dialog (see page 164) appears.

2. Select the template (see page 24) you want to uses as a basis for your project.

   The Select Data Source for a new project dialog appears.

3. Complete the details on the Select Data Source for a new project dialog, then click OK.

   The Endpoint Type Details (see page 129) dialog appears.

4. Complete your attribute mappings as required.
5. Save the project (see page 31).

Wizard Mode

You can also start a project by opening a template in wizard mode. When you start a project using the wizard, the wizard steps you through the basic process of mapping an account class, or an account class and group class, depending on the template you selected. You can use the wizard to start a project, and then use the other dialogs to complete your mappings.

The wizard prompts you to complete the following mapping details:

1. The endpoint type name and description.
2. The native classes for the account mapping.
3. The mappings for account attributes.
4. For templates projects that contain a group class, the wizard prompts you to complete the following mappings:
   a. The native classes for the group mappings.
   b. The mappings for group attributes.
   c. The association between the account class and the group class.

Create a Project Using the Wizard

You can use the wizard to complete the basic details of a mapping, such as mappings for account and group attributes. You can use the wizard to start a project, and then use the other dialogs to complete your mappings.

To create a project using the wizard

1. On the Project menu, click Create New from Template.

   The Select template dialog appears.
2. Select the template (see page 24) you want to uses as a basis for your project.
3. Click Open in Wizard Mode.

   The Select Data Source for a new project dialog appears.
4. Complete the details on the Select Data Source for a new project dialog, then click OK.

   The Endpoint Type Details dialog appears.

5. Follow the prompts in the wizard and complete your mappings as required.

   Note: While the wizard is active, the mapping tree is read-only. As you step through the wizard, the relevant node in the mapping tree is expanded and highlighted.

6. Review the mapping summary, then click Finish.
7. Complete your attribute mappings as required.

Projects Based on Existing Metadata

You can create a project based on the contents of eTMetadata and eToperation binding set for both dynamic and static connectors. Creating a project based on existing metadata allows you to:

- Review or modify the class mappings and operation bindings for a connector type in Connector Xpress.
- Deploy (see page 34) the changes back to the endpoint.

Creating a project based on existing metadata is useful when:

- The original project file that was used for creating the connector is no longer available.
- A project file for the connector does not exist, for example, for static connector types.
- A project file that reflects the latest changes for the connector type is required.

Create a Project Based on Existing Metadata

To review or modify the class mappings and operation bindings for a static or dynamic connector where the original project for the connector does not exist, you can create a project based on existing metadata.

To create a project based on existing metadata

1. In the Provisioning Servers tree, right click the endpoint that you want to base a project on, then click Create Project.

   The progress dialog appears. If you selected a dynamic connector, Connector Xpress displays the Select Data Source for New project dialog.

2. If prompted, select a data source for your project, then click OK.

   Connector Xpress creates the project and populates the mapping tree with the metadata.

3. Map the native attributes in the Account Class (see page 47) to the relevant provisioning attributes.

4. Save the Connector Xpress project. (see page 31)
5. Redeploy the metadata if necessary. (see page 89)

Open an Existing Project

To open an existing project, click Project, Open, and double-click the file you want to open from the Open Project dialog.

Connector Xpress opens the project and displays the mappings in your project in the Mapping Tree (see page 15) of the Connector Xpress window.

Reopen a Recent Project

To reopen a recent project, click Project, Open Recent, and select the project you want to open.

Connector Xpress opens the project and displays the mappings in your project in the Mapping Tree (see page 15) of the Connector Xpress window.

Edit a Project

To change the details of a project, you can open a saved project and edit the metadata as required.

To edit a project

1. Open an existing project (see page 30).

   The Select Data Source for new project dialog appears.

2. Complete the details on the dialog, then click OK.

   Connector Xpress opens the project, and displays the metadata in the mapping tree of the Connector Xpress window.

   The Enter Password for Data Source dialog appears.

3. Complete the fields on the dialog, then click OK.
4. Edit the metadata as required.
5. Save the connector. (see page 31)
6. Deploy the connector (see page 34) if necessary.

More Information:

Projects (see page 23)
Enter Password for Data Source Dialog (see page 129)

Save a Project

After you have created a new dynamic connector, you can save the connector project for future deployment.

To save the project

1. Click Project, Save, or Save As.

   The Save Project As dialog appears.

2. Specify the folder to save the connector project in, the file name and the file type for the saved connector, and click Save.

   Connector Xpress saves the project.

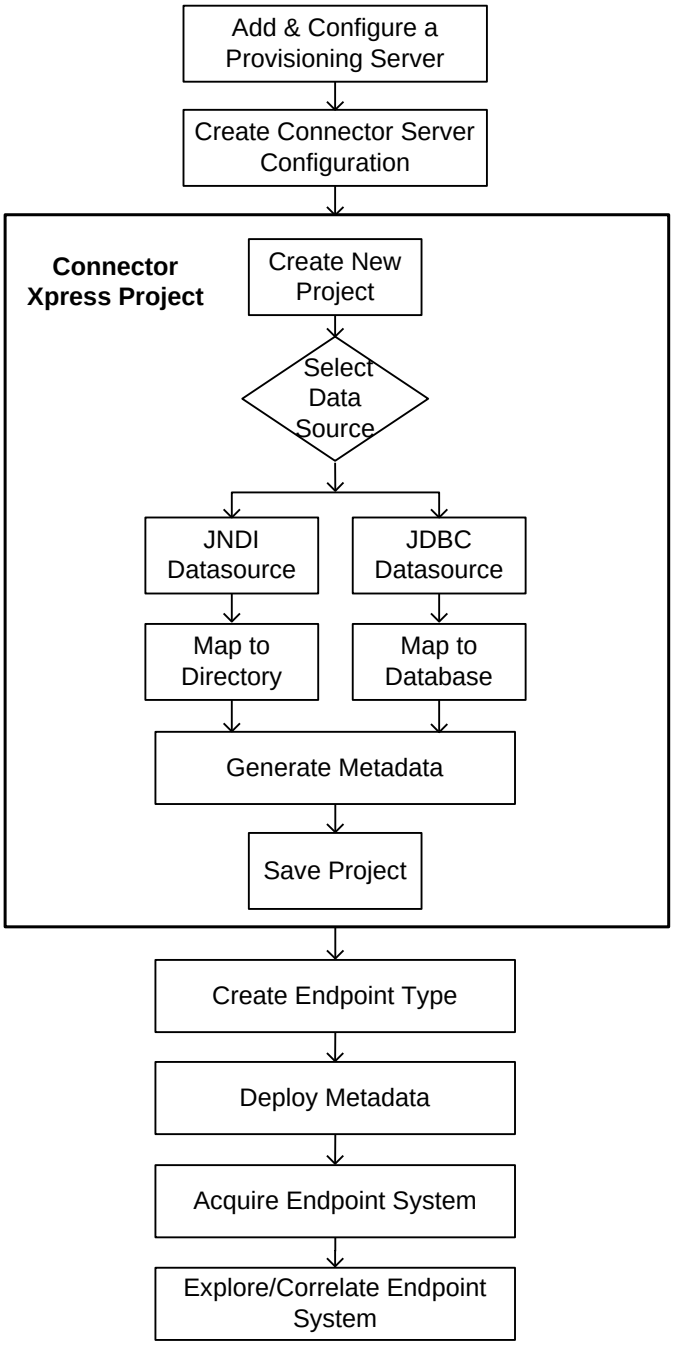
   Note: We recommend that you use the default file extension (.con) and that you keep a copy of the project. You cannot view the stored procedure bindings after you deploy a connector, unless you open the Connector Xpress project file where your bindings are stored.

More Information:

Projects (see page 23)

How To Create and Deploy Connectors

The following diagram shows the process you follow to create and deploy a connector using Connector Xpress:



Example: How you create and deploy a basic JNDI connector

The following steps detail the creation and deployment of a basic dynamic JNDI connector. To create a basic JNDI Connector, do the following.

1. Start Connector Xpress.

   The Connector Xpress main window appears.

2. Add and configure the connection details of the Provisioning Server (see page 73) where the endpoint you want to manage is located.

3. If the Provisioning Server does not have a Connector Server configured, create (see page 75) and set a managing Connector Server (see page 76) for the endpoint.

   The configuration determines how your Provisioning Server routes individual endpoints to Connector Servers.

4. Create a project (see page 26).

5. Set up a JNDI data source (see page 15), and test the connection and authentication details of your JNDI data source.

   Connector Xpress retrieves the schema from this JNDI data source, which lets you map to the schema to your new endpoint on the Provisioning Server.

   Note: When you create a project, Connector Xpress automatically creates a class named User Account in the Mapping Tree.

6. Map the native attributes in the user account class (see page 47) to the relevant provisioning attributes.

7. Save the project. (see page 31)

   You can save the project for future deployment.

8. Create an endpoint type (see page 84) on the Provisioning Server and specify the managing connector server (see page 76) you want to manage your endpoint type.

   Creating an endpoint type and specifying the managing connector server does the following:

   - Creates the endpoint type on the Provisioning Server you want to manage and deploy the connector to.
   - Specifies how your Provisioning Server routes endpoint types or individual endpoints to connector servers.

9. Deploy the connector to the Connector Server (see page 34).

   Deploying the metadata creates the new dynamic JNDI Connector.

10. Acquire, explore, and correlate the endpoint (see page 83).

    Connector Xpress populates the Provisioning Server with accounts and other objects found in the acquired endpoint.

    Correlating an endpoint is the process of examining accounts on it, and potentially creating users automatically and setting global user attributes from account attributes.

    To manage your objects, you explore the endpoint for your objects and correlate the objects to global users.

11. Generate CA Identity Manager User Console Account Screens (see page 62).

Connector Deployment

You can use Connector Xpress to configure metadata settings and to deploy the connector. Connector Xpress lets you save metadata settings to a project file (see page 23) and then deploy a connector from that file. Saving metadata to a project file is useful in cases where the same settings apply on multiple Provisioning Servers. For example, when you move connectors from a test environment to a production environment after testing is complete.

Deploy the Connector

Once you have created a connector, you can deploy the metadata for that connector to a Provisioning Server.

Note: This procedure assumes that you have created the connector server configuration (see page 75).

To deploy the connector

1. Open the existing project (see page 30) that contains the metadata you want to deploy.

   Connector Xpress opens the project and displays the metadata in the Edit pane of the Connector Xpress window.

2. In the Provisioning Servers tree, expand the Provisioning Servers node and then choose the server where you want to deploy the connector.

   The Provisioning Server Password Required dialog appears.

3. Complete the fields on the dialog to specify the password for the server, and click then OK.

4. Expand the server, and then right-click Endpoint Types, then click Create New Endpoint Type.

   The Create New Endpoint Types dialog appears.

5. Complete the fields on the dialog to define the name of your new endpoint type, then click OK.
6. Right-click on your new endpoint and select Acquire Endpoint.

   The Create New Endpoint dialog appears.
7. Complete the fields on the dialog to specify the name and password for your new endpoint, then click OK.
8. (Optional) Under the endpoints node, right-click on your new endpoint and choose Explore/correlate endpoint.

   The Explore/Correlate Endpoint dialog appears.
9. Complete the fields on the dialog to specify how Connector Xpress explores and correlates the endpoint, then click OK.

   Connector Xpress deploys the connector to the Provisioning Server.

More information:

Provisioning Servers Tree (see page 16)
Create New Endpoint Dialog (see page 115)
Explore/Correlate Endpoint Dialog (see page 130)
Projects (see page 23)

Connector Undeployment

You can use Connector Xpress to undeploy connectors. Undeploying a connector deletes the connector from the Provisioning Servers persistent storage and from the Java CS. You can also delete connectors (or instances of their parent endpoint types) using the Provisioning Manager user interface. In both cases, you do not need to restart the Java CS, and Connector Xpress does not delete data on the endpoint.

Multitable Support for JDBC Connectors

A compound class is an unmanaged class that you can use as the data type of an attribute.

To provide support for multitable JDBC Connectors, you map an unmanaged class to an endpoint object and use the compound class as a new data type for an account class attribute.

This means that values from multiple columns from a table, rather than from a single column, can populate a single attribute value.

Compound Class Mapping

You can map compound classes like normal classes, except that a naming attribute is not required.

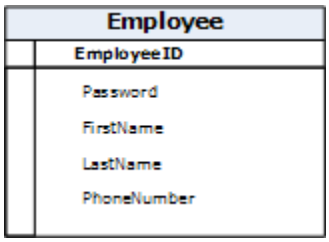
Note: For more information about compound types, see the *Programming Guide for Java Connector Server*

Compound Class for Multitable Support Example

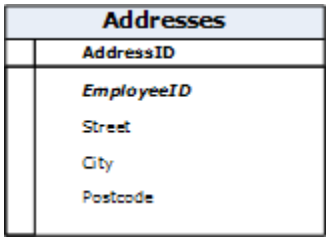
This example shows you how to create a simple connector named My_MultiTable_Connector that combines two database tables into the user account class.

This example uses a schema named Multi_Table schema that contains an Employees table and an Address table.

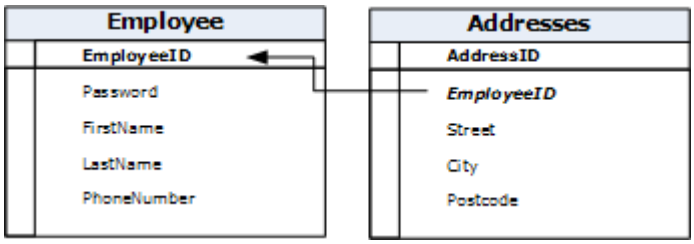
The Employees table is the main table for employee records. The table uses EmployeeID as its primary key as shown in the following diagram:



The Addresses table is a separate table of addresses that uses EmployeeID as a foreign key as shown in the following diagram:



The following diagram shows the relationship between the tables:



Example: Create a compound class for multitable support

This example combines the Employees table and Addresses table in the Multi_Table schema so that they are jointly accessible from the user's endpoint account in the CA Identity Manager User Console account management screens.

To create a compound class for multitable support

1. Create a JDBC project named My_MultiTable_Connector and specify the JDBC data source you want to use.
2. In the mapping tree, click the User Account class node.

   The Mapped Class and attributes (see page 136) dialog appears.
3. Select the Multi_Table schema from the Schema drop-down list, then select the Employees table from the Table drop-down list.
4. Map the the columns in the Employees table to the attributes in the Name column.
5. Click the Custom Types node.

   The Custom Types (see page 119) dialog appears.
6. Under Compound Type classes, click Add, then type the name of the new class, for example, Address, in the table.

   Connector Xpress creates a compound class called Address and adds it to the mapping tree. Connector Xpress sets the Address compound class to Unmanaged by default.

   Connector Xpress creates an account attribute named Addresses, adds it to the mapping tree, and selects the multivalued attribute check box on the Attribute details dialog. Connector Xpress also creates a direct association from the Address class to the User Account class and adds a node named *with User Account* to the Mapping tree. The Addresses attribute and direct association incorporates the Address compound class into the User Account class.

   Note: The Address class has the special *is compound value* metadata flag set to true on the Map Compound Class and Attributes dialog. This flag specifies that the class is a compound class.

7. In the Mapping tree, click the Address class node.

   The Map Compound Class and Attributes (see page 144) dialog appears.

8. Select the Multi_Table schema from the Schema drop-down list, then select the Address table from the Table drop-down list.

9. Map the columns in the Addresses table to the attributes in the Name column.

10. Click the With User Account node under the Address node.

    The Direct Association with User Account (see page 120) dialog appears.

11. In the Address Attribute field, select Employee ID.

    Connector Xpress matches this attribute with the User Account naming attribute.

    Note: The Employee ID attribute in the Address class serves as the association attribute. The attribute has a special association type metadata property (Assoc Type=COMPOUND_PARENT) set that defines it as the attribute linking this compound class to its parent class. To display this property on the Attribute Detail dialog, display the extended metadata properties. (see page 91)

12. Under the User Account node, click the Attributes node.

    The Attributes Summary (see page 112) dialog appears.

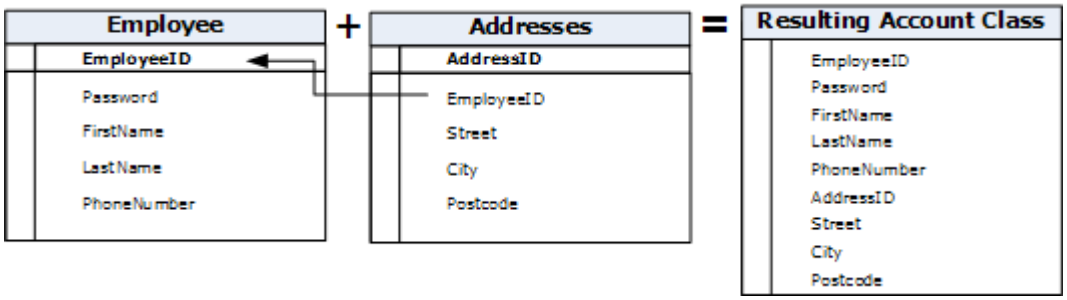
13. Group the mapped attributes into the logical groups and subgroups that you want to appear as tabs and page sections in the CA Identity Manager User Console account management screens by creating the presentation metadata in Connector Xpress. (see page 65)

14. Save your project and deploy the new connector.

15. Generate CA Identity Manager User Console Account Screens. (see page 62)

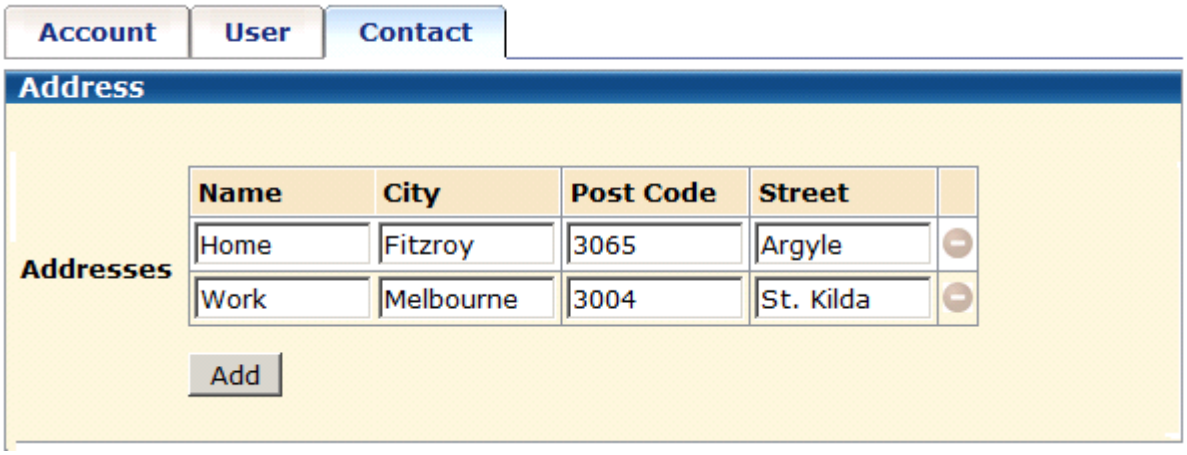
Example: Results of combining the Employees and Addresses table

The following diagram shows the results of combining the Employees table and Addresses table:



Example: Generated account screens

This example shows how the account management screens look after you import the My_MultiTable_Connector-RoleDef.Xml file into CA Identity Manager:



Managing Accounts and Groups

In earlier releases of Connector Xpress mapping a group object type implicitly created the association between groups and accounts. However, in this release of Connector Xpress, you have fine-grained control over the associations between accounts and groups on the endpoint system. To define groups and define group membership, you explicitly create associations between classes. You can create direct, reverse, or indirect associations. Creating an association between classes defines the class as a group class.

For JNDI connecters, the group class member attribute is hardwired to contain values of type DN (Distinguished Name). Values of this type are expressed relative to the root of the endpoint directory and enumerate the accounts belonging to each group.

The member attribute for the group class is virtual, meaning that its value is expensive to retrieve, as it has to be computed from group.member rather than being directly looked up. We therefore recommend that you request it with caution.

Some JNDI vendors, notably Novell eDirectory, actually expose the account.memberOf attribute in their schema. However to guarantee consistent behavior across all vendors, you are prohibited from mapping it explicitly. Instead, the Java CS implements it as a virtual attribute.

Mappings

Mappings consist of the following:

- One or more **class mappings**—Each class mapping describes a single class of provisioning objects in the Provisioning Server's DIT. For JDBC, a class mapping maps to a single database table. For LDAP, a class mapping can map to multiple native LDAP object classes. Account class mappings have some special handling because they are important to Provisioning Server operations like policy syncing.
- Multiple **attribute mappings—**Each class mapping can have multiple attribute mappings, which map individual values in the Provisioning Server's provisioning objects to single values in objects on the managed endpoint system.

Multi-attribute Mappings

Connector Xpress supports multi-attribute (many-to-one) mappings, which means you can map a native attribute to multiple provisioning attributes. The LDAP DYN template (see page 24) contains an example of multi-attribute mapping. In the template, the account class has both Common Name and Account ID mapped to the endpoint's *cn* attribute. This is useful because the Common Name is a common LDAP attribute that you should include in the account object and Account ID is the provisioning naming attribute required by the Identity Manager common attribute set.

Also, the Account ID and *uid* are both ambiguously mapped to the endpoint's *uid* attribute in the template.

Duplicate mappings to a native attribute within the same class are not allowed. For example, if *cn* is mapped you cannot map *cn* again. However you can map cn again as part of an *ambiguous* mapping. For example, together with *uid* as shown in the LDAP DYN template.

Note the multi-mapping of *accountname, cn, uid,* and a separate mapping directly to *cn* is required to fully manage JNDI endpoints. This allows either *cn* or *uid* to be used in the naming attribute, and satisfies the condition where *cn* is also required on an endpoint even when *uid* is used as the naming attribute.

Map Multi-attributes

As JNDI endpoints can contain accounts named using *cn* or *uid* attributes, map Account ID to both *cn* and *uid* attributes, and also provide single unambiguous mappings to *cn* and *uid t*o manage such endpoints.

To map multi-attributes

1. On the Project menu, click New.

   Connector Xpress automatically creates a user account provisioning class node in the Mapping tree when you create a project.

   The Select Data Source for new project dialog appears.

2. Select the data source you want to use for the project.

   The Endpoint Types dialog appears.

3. On the Endpoint Type Details dialog, specify a name, description, and version for your connector.

   Note: These fields are for descriptive purposes only.

4. In the Mapping Tree, click the User Account node.

   The Mapped Class and Attributes dialog appears.

5. Select the endpoints type's object class that you want to map from the Add structural class list. For example, inetOrgPerson.

6. In the Map Object Class Attribute mapping table, select the provisioning attribute you want to map then select Multi Attribute in the name list.

   A dialog appears that displays the provisioning attributes you can map in the Available list.

7. Select the attributes you want to map to the Available list and move them to the Selected list, then click OK.

8. Map the endpoints mandatory attributes to the provisioning attributes.

9. Save the Project. (see page 31)

Ambiguous Mappings

Connector Xpress supports ambiguous (one-to-many) mappings, which means you can map one provisioning attribute to multiple native attributes.

For example, in an organization where a merge of two employee databases has resulted in a large employee directory, and the employee objects are either an "inetOrgPerson" object or an "account" object, you can ambiguously map the Identity Manager account class to both object classes. As a result, when you search for accounts, the search finds both object types, and you can select which object type you want to use.

At the attribute level, the account class can be represented by only one object class, but a certain value can be populated by one of two attributes. For example, inetOrgPerson has both a *postalAddress* and a *registeredAddress* attribute. If some employees have their address saved in *postalAddress* and other employees in *registeredAddress,* for example due to a directory merge, you can ambiguously map the Identity Manager address attribute to both of these endpoint attributes and use them interchangeably.

Create an Ambiguous Mapping for a Naming Attribute in Connector Xpress

When you create an ambiguous mapping for a naming attribute in Connector Xpress and you want to specify both the ambiguous mapping choices simultaneously, create accounts with both ambiguous choices (for example, *uid* and *cn* attribute values). Use the same set of mappings and any additional unambiguous mappings for the same choices (for example, for *cn* and *uid*) separate to the naming attribute so that Connector Xpress generates correct mappings, before you use the mappings in Provisioning Manager.

As a single ambiguous account name mapping serves as an OR combination between two choices, that is, *cn* or *uid*, you can use Connector Xpress to generate an ambiguous mapping for account name and an additional straight or unambiguous mapping for *cn*. This means that when you use *uid* as the naming attribute you can still set a *cn* value for that account using supplementary, that is, unambiguous, mappings.

To create an ambiguous mapping for a naming attribute in Connector Xpress

1. Map account ID to each choice, for example, *cn* and *uid*.
2. For each of the choices (for example, *cn* or *uid*) you want to make available when you map the naming attribute, to your other choice, do the following:
   a. Click the Attributes node.

      The attribute editor panel appears

   b. Add the new attribute mapping, and specify native connectorMapTo value matching the choice (for example, *cn*).

      When you map eTDYNAccountName to *cn* or *uid* the combination of the attributes is an OR combination – that is, *cn* OR *uid*.
3. If you want to make the combination an AND combination, that is *cn* AND *uid*, make another attribute mapping where connectorMapTo=cn.
4. Check that the property mapping you created and save the metadata.

Types of Associations

You can create the following types of associations in Connector Xpress:

- (JNDI and JDBC) Direct association
- (JNDI and JDBC) Reverse association

  Note: Reverse associations are not supported for associations between compound classes and user account classes when creating multitable support JDBC connectors.

- (JDBC only) Indirect association

Direct Associations

A *direct* association is an association between any two classes of objects where the association values are stored on one of the objects directly.

A direct association can be in the forward direction, that is from group to account, and in the reverse direction, that is, from account to group. Creating a direct association in the forward direction lets you manage the accounts that belong to a group from the group side of the association. Creating a direct association in the reverse direction, that is a reverse association (see page 44), lets you manage the association from the account side of the relationship.

Direct associations in the forward direction coincide with the natural representation of the associative information about the endpoint objects, that is, where the group stores account members. Direct associations in the reverse direction are the reverse of this representation, as they define groups to which an account belongs, even if the native system does not store such information.

Typically, it is common to establish both directions of an association at the same time, that is, both the direct and reverse associations. You can use the Direct Association (see page 120) dialog to create and edit both associations at the same time.

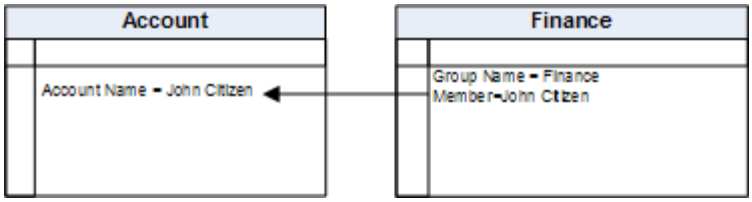
In a direct association, a group directly stores the directory-relative DNs of the accounts which belong to it. Therefore you create a direct association between classes when you map groupOfNames.member to inetOrgPerson.cn.

Note: Both of these objectclasses are part of the inetOrgPerson schema. They differ only in minor implementation details on the endpoint, that is, groupOfUniqueNames stores its members as a set rather than a list helping ensure that the associated entries are unique.

In direct associations, references are persisted directly into a multivalued attribute on the endpoint. For example, in LDAP, a group's member attribute directly stores reference to the accounts it contains.

Example: Direct Association

The following example shows a direct association that has been mapped between an account class and a finance class. The finance class stores the accounts that belong to it in its member attribute.



Reverse Associations

A reverse association is a direct association between two classes of objects of the type *from* Class1 *to* Class 2 (the direct association) and *from* Class 2 *to* Class 1 (the reverse association). Reverse associations are by definition bi-directional.

Typically, most endpoints only let you manage only one side of the association between accounts and groups from the group side of the association. For example, you can manage the accounts that belong to a group from the group side of the association. Creating a reverse association lets you provision and manage which groups an account is a member of, from the account side of the association.

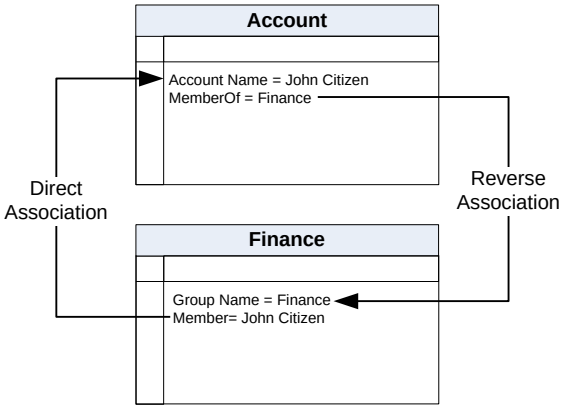
As most bi-directional associations have a physical attribute on one class and a virtual attribute on the other class, we recommend that you define the physical association attribute first.

Typically it is common to establish both directions of an association at the same time, that is, both the direct and reverse associations. You can use the Direct Association (see page 120) dialog to create and edit both associations at the same time.

Reverse associations appear in the mapping tree under the node of the class you have specified a reverse association with.

Example: Reverse Association

The following diagram shows a direct and reverse association between the account and group finance class, that is, a bi-directional association, created when you map the attribute in the account that contains the groups the account belongs to, for example, the *memberOf attribute*, to the group's naming attribute.



Indirect Associations

An *indirect a*ssociation occurs when there is a third entity defines the association between any two classes of object. For example, an intermediate table that binds two other database tables together.

In an indirect association, the association is stored as an independent entity rather than as a property on one of the objects.

For example, there is an indirect association between an account object and a group object if there is an intermediate table, such as a *membership table,* that identifies the associations between individual accounts and groups.

In a *direct* association, objects have an attribute that points directly to the other object. However in an *indirect* association, both of the related objects have attributes that point not to each other, but to the membership table.

Membership tables define the members of each group. They contain the association mappings that identify the individual accounts and groups that are related, for example, the groups an account belongs to, and a list of accounts in each group. In a membership table, associations between objects are stored in a many-to-many mapping table.

Membership tables let you specify a separate lookup table for associations between account and group objects and map the relevant attributes.

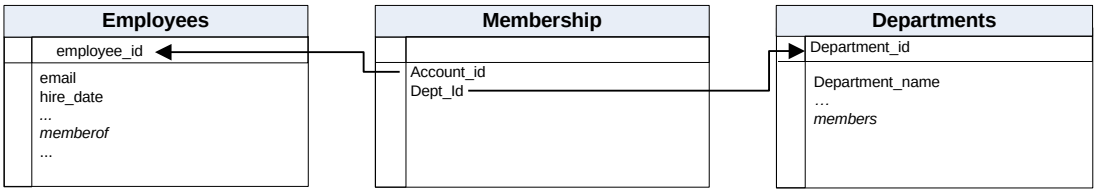
To define an indirect association, you specify the membership table that contains references to both of the related objects.

When you create an indirect association between two objects, Connector Xpress automatically creates the reverse association, that is; indirect associations are by definition bi-directional.

Note: You can only create indirect associations for JDBC mappings.

Example: Indirect Association

The following diagram shows the indirect association that has been mapped between the employee and department classes using a membership table. The following is an example schema and database.



The following are the three tables in the database:

- **Employees table**–lists the employees in the organization.

  The virtual attribute memberOf contains Group IDs, that is the departments the employee is a member of.

- **Membership table**–defines the associations between Employees and Departments. It contains the association mappings that identify the individual employees and departments that are related, that is, which departments an employee belongs to, and a list of employees in each department.

- **Departments table**–lists the departments in the organization.

  The virtual attribute members contains Account IDs, that is,the list of employees in the department.

More Information:

How to Define Group Membership (see page 46)
How to Create an Indirect Association (see page 50)
Indirect Association Example (see page 51)

How to Define Group Membership

To define group membership and create an association between classes that defines the class as a group class, you do the following:

1. Create an account class and map its attributes.

   Note: When you create a project, Connector Xpress creates a user account class by default.

2. Create the class you want to define as a group class, and map its attributes.

   Note: If you use the wizard to map account and group classes, the wizard automatically creates a group class and the details of the association that you need to complete.

3. Create a direct, reverse, or indirect association between the classes and map the groups group.member or group.uniqueMember attributes to the accounts naming attribute.

   Creating an association defines the group membership and creates the association between the classes.

Account Class Mapping Example

This example shows you the procedure you would follow if you were an administrator that wanted to map an account class for a JNDI data source. This example shows you how an administrator maps an endpoint's account class to the provisioning account class. This example assumes that the administrator has set up setup a JNDI data source.

To map an account class

1. On the Project menu, click New.

   Connector Xpress automatically creates a user account provisioning class node in the Mapping tree when you create a project.

   The Select Data Source for new project dialog appears.

2. Select the data source you want to use for the project.

   The Endpoint Types dialog appears.

3. On the Endpoint Type Details dialog, specify a name, description, and version for your connector.

   Note: These fields are for descriptive purposes only.

4. In the Mapping Tree, click the User Account node.

   The Mapped Class and Attributes dialog appears.

5. Select the endpoints type's object class that you want to map from the Add structural class list. For example, inetOrgPerson.

6. Map the endpoints mandatory attributes to the provisioning attributes in the Map Object Class Attribute mapping table. For example, cn and sn to Account ID and Last Name.

   Note: If the table list fails to populate when mapping tables, verify that your database does not have outstanding transactions or locks on the schema metadata.

7. Map any other required attributes, for example, the user password, street, and title.

8. Click the Classes node in the mapping tree.

   The Mapped Classes dialog displays a summary of the classes you have mapped. You can use this dialog to revise the native class to provisioning mappings you have made.

9. In the Mapping Tree, click the Account Id node under the Attribute node.

   The Attribute Details dialog appears.

   The dialog displays the LDAP attribute assigned to each field, its datatype, the JavaBean property name that JIAM uses, and whether the field is required (allows null values) and any length constraints.

10. Click the Last Name node under the Attributes node.

    The Attribute Details dialog appears with the default policy value set.

    Note: When you map a required attribute to a well-known provisioning attribute, Connector Xpress sets a default account template value by default.

11. Save the project.

How you Create a Direct and Reverse Association

To create a direct association between two classes, for example, an account class, and a group class, you do the following:

1. Start a new project and specify a data source for your project.
2. Create and map an account class.
3. Create a class that you want to define as a group and map it to the class on the endpoint that defines the entries for a group of names, for example, *groupOfNames*.
4. Map the group's name and its member attribute.
5. Specify that you want to create a direct association between the group class and the account class.

6. Map the group's member attribute to the account's naming attribute.

   Mapping the group's member attribute specifies that the group's membership attribute is populated by the account's naming attribute.

   This describes the association between the group class and the account class and creates a direct association between the account class and group class.

7. Specify that you want to create a reverse association between the account class and the group class.

8. Map the account class memberof attribute to the group's naming attribute.

   Note: If the native account class does not have a memberOf attribute, to create a virtual memberOf attribute and map it to the group's naming attribute.

   Connector Xpress creates the direct and reverse associations between the account and group class you have mapped and automatically creates and displays the association under the User Accounts node.

Direct and Reverse Association Example

This example shows you the procedure an administrator would follow to create a direct and reverse association between an account class, and a group class. In this example, the administrator defines an association that describes the relationship between the group class and the user account class.

This example assumes that the administrator has setup a JNDI data source, and created and mapped an account class named User account (see page 47).

To create a direct and reverse association between an account class, and a group class defines an association that describes the relationship between the group class and the user account class.

To create a direct and reverse association

1. In the mapping tree, click the Classes node.

   The Map Class and Attributes dialog appears.

2. Click Add on the Mapped Classes dialog, and type a name and for your class, for example, Group of Names.

   Connector Xpress adds the new class to the mapping tree.

3. In the mapping tree, click the Group of Names node.

4. In the Add structural class list, select the native class that you want to map, for example, groupOfNames.

   The attributes for the class appear in the Map Object Class Attributes table.

5. Map the groups name and the group's member attribute to the provisioning attributes. For example, map the native attributes *cn* and *member* to the provisioning attributes objectname and member.
6. Select the Multivalued check box for the member attribute.

   Selecting the check box specifies that the member attribute is multivalued and can hold multiple account names.
7. Click the Associations node under the Group of Names node.

   The Class Associations dialog appears.
8. In the Create direct association with list, select the User Account class.

   Connector Xpress adds a node named with User Account to the mapping tree.
9. Click the with User Account node under the Associations node, under the Group of Names node.

   The Direct Association with User Account dialog appears.

   Note: Connector Xpress selects the group's naming attribute in the Group of Names By Attribute field as the attribute to map to by default.
10. In the Group of Names Attribute, select member.

    Selecting the member attribute maps the groups member attribute to the account's naming attribute. That is, you have specified that the groups member attribute is populated by the accounts naming attribute, and defined the Group of names class as a group class.
11. Select the Include a Reverse Association check box.

    The Reverse Association dialog appears.
12. In the New Virtual Attribute field, type memberof.

    In this example, the native account class does not have a memberOf attribute, so create a virtual memberOf attribute and map it to the group's naming attribute.

    Note: Connector Xpress selects the group's naming attribute in the By Attribute field as the attribute to map to by default.
13. Click Project, Save.

    Connector Xpress creates the direct and reverse associations between the account and group class you have mapped.

How to Create an Indirect Association

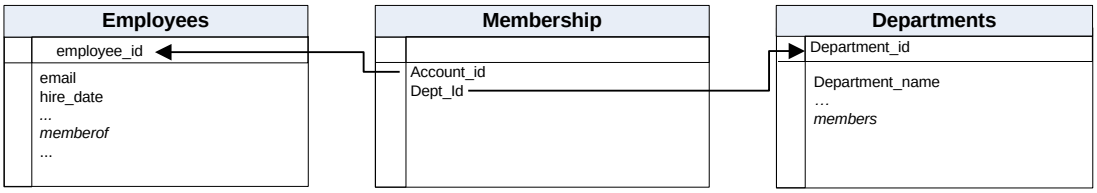
This example shows you the process you use to define an indirect association between two classes, for example, an employee class, and a department class using a membership table. To create the indirect association:

1. Create a project (see page 23) and specify a JDBC data source (see page 14) for your project.
2. Create and map class that holds a list of employees.
3. Create and map a class that holds a list of departments.
4. Specify that you want to create an indirect association between the employee class and the department class.
5. Specify the membership table that contains the association mappings that identify the individual employees and departments that are related.
6. Specify the membership table column that defines the association between employees and departments, and the membership table column that defines the association between departments and employees.
7. Create a virtual *memberof* attribute in the employee class and map and it to the membership table column that contains the list of departments the employee is a member of.
8. Create a virtual *members* attribute in the department class and map and it to the membership table column that contains the list of employees in each department.
9. Save the project.

Indirect Association Example

This example shows you the steps you would follow if you are an administrator that wants to create an indirect association between an Employees class, and a Departments class using a membership table.

This example uses an example schema, *HR*, and an example membership table, *Membership.* The membership table columns *Account_Id* and *Dept_id* contain the association mappings that identify the individual employees and departments that are related, as shown in the following example:



To identify individual employees and departments that are related, create an indirect association between them.

Note: This example assumes that you have already setup a JDBC data source (see page 14), and created and mapped an account class named Employees.

To create an indirect association

1. Click the Classes node.

   The Mapped Classes dialog appears.

2. Click Add, then specify a name for your class, for example, Departments.

   Connector Xpress adds a node named Department to the mapping tree.

3. Click the Departments node in the mapping tree.

   The Map Class and Attributes dialog appears.

4. Specify a name and description for your class.

   Note: These fields are for descriptive purposes only.

5. Select the schema and table you want to map. For example, HR and Departments.

6. In the Name column, map the account naming attribute to the objectname attribute.

7. Click the Associations node under the Employees node.

   The Class Associations dialog appears.

8. Create an indirect association with the Departments class.

   Connector Xpress adds a node named with Departments under the Associations node for the Employees class in the mapping tree.

9. In the Mapping tree, click the *with Departments* node under the Associations node.

   The Indirect Association dialog appears.

10. Specify the schema that contains the classes you want to map, for example, *HR.*

11. In the Membership Table list, select the membership table, *Membership*.

    This table specifies the association mappings that identify the individual employees and departments that are related.

12. In the Employees Attribute list, select the Employees class naming attribute.

13. In the Membership Table Columns list, select AccountID and DeptID respectively.

    The result is the following:

    - Maps the Employees class naming attribute, employee_id, to the membership table column, AccountID, and defines the association between employees and departments.
    - Maps the Department class naming attribute, Department_id, to the membership table column, DeptId. That defines the association between departments and employees.

    Note: Connector Xpress selects the naming class attributes by default in the Employees and Departments Attributes list.

14. In the Employees Attribute and Departments Attributes fields, type *member* and *memberof* respectively.

    Note: In this example, because the Employees and Department classes do not have a memberOf or *member* attribute, create a virtual *member* and memberOf attributes.

    The virtual attributes you create describe the association between the Employee class and Department class.

    These virtual attributes are a virtual representation of the association between the employee class and department classes naming attributes, and the membership table columns you mapped in step 11. The connector uses these virtual attributes to find the employees in a department, and the departments an employee belongs to.

    Connector Xpress automatically does the following:

    - Creates a *memberof and member* node under the Attributes node under the Employees and Department class node in the mapping tree.
    - Automatically makes the attribute multivalued and selects the multivalued check box by default on the Provisioning Details dialog.
    - Automatically creates the indirect association and displays it under the Associations node under the Departments node.
    - Automatically creates the association between the Employees and Departments class from the Departments class point of view, under the Employees node. Attributes viewed from the Associations node under the Employees class therefore appear reversed.

15. Click, Project, Save.

    Connector Xpress creates the indirect association between the Employees and Departments class you have mapped.

Container Classes

Specifying a container class is similar to mapping an ordinary class except that the only attribute that you can map is Container Name.

Additionally, you can specify the classes that are the children of this container ("Contained Classes"). For example, the container Employee Groups can only allow Staff Group and Executive Group classes and not individual account classes.

Input Validation

Connector Xpress validates the entries you make in the fields on the Endpoint Type, Map Class and Attributes, Provisioning Attributes, and Defined and Indirect Associations dialogs. Connector Xpress displays a warning icon next to any field that has invalid input, and a warning icon next to the corresponding node in the mapping tree. Connector Xpress displays details of the violation when you mouse-over the warning icon. The warning icon disappears after you correct your input and click another node in the mapping tree.

Operation Bindings

An operation binding is additional logic, such as a stored procedure or a script, that you can bind to a particular operation to specialize the handling of that operation. You can specify the timing of the logic invoked by the stored procedure in relation to the operation, that is, will it run before, after, or instead of the operation.

When the Java CS performs any operation, the Java CS verifies whether there are any operation bindings that tell it to invoke some logic before, instead of, or after that operation.

For example, imagine you want to add a user to an endpoint. You have the user's given name and family name. On the endpoint system, however, the record for that user is an attribute made up of a particular combination of their given name and family name. To resolve this situation, you can create a script that combines the two names to match the endpoint format and then, using an operation binding, specify that this script must run before any search you perform on the endpoint system.

- Stored Procedures (JDBC endpoints only)

  An operation binding can invoke a stored procedure for Add, Modify, Delete, and Rename operations.

- Scripts (JDBC and JNDI endpoints)

  An operation binding can invoke a script for Add, Modify, Delete and Rename operations, and an additional 16 operations such as Search, Lookup, Modify-Assocs, Delete-Assocs, Activate, Deactivate, and such.

  A script in this instance can either be a called function stored in a global script, or a stand-alone scriptlet. Calling functions that exist within a global script is best practice because you can then reuse those functions for other operation bindings.

You can apply an operation binding to any object class. For example, if you want to record all modify operations to either an account object or a group object in a log file, you could use a single operation binding and apply it to both.

You can apply multiple operation bindings with the same timing (before, after, instead of) to a single object. For example, invoking two stored procedures to run before a particular operation.

Stored Procedures

Stored procedures are located on JDBC endpoints. They are code that the Java CS can invoke before, after, or instead of an operation. Stored procedures are only relevant for relational database endpoints and are written in the language specific to that endpoint.

More Information:

Bind Operations to Stored Procedures (see page 56)

Scripts

Scripts are located in the Connector Xpress project file and are written using JavaScript. As with stored procedures, Java CS can invoke a script before, after, or instead of any Identity Manager operation.

You can create global scripts or individual scripts. A global script contains JavaScript functions that any number of operation bindings can invoke. Global scripts are an excellent way to store and reuse common functions.

An individual script is a piece of JavaScript code that is only used by a single operation binding. You would typically use an individual script for a simple specialization of the operation. If you want to bind several functions within a global script, you can create an individual script that invokes selected functions.

More Information:

Bind Operations to Scripts (see page 57)

Bind Operations to Stored Procedures

You can bind operations to stored procedures on JDBC databases to specify actions that you want to occur, before, after, or instead of, standard account CRUD operations. For example, Add, Modify, and Delete. Other types of operations are available, depending on the type of operation binding you select.

To bind operations to stored procedures

1. Create a Connector Xpress JDBC Project.
2. Map the user account class, and any other class as required.
3. Click the Operations Bindings node.

   The Operation Bindings Editor appears.
4. In the Object Classes Filtering list, select an object class.

   You have specified the object class you want to create the operation binding for.
5. Click Create.

   The Create Operation Binding dialog appears.
6. Select a class in the Available object classes list and add it to the Added object classes list.

   You have specified the object classes you want to apply the operation binding to.
7. Under Type, select Stored Procedure.
8. In the Available Operations list, select an operation.

   You have specified the type of operation that you want to bind the operation to.
9. In the Timing list, select a timing.

   You have specified when the operation binding is executed.
10. Click OK.

    A node is added to the mapping tree which displays the type of operation, the timing you selected and the name of class you want the operation binding to apply to.

11. Click the node that displays the information about the operation binding you created.

    The Stored Procedure dialog appears.

12. In the Procedure list, select a stored procedure.

    You have bound the procedure to the operation you specified in step 8.

    Note: If the procedure list fails to populate verify that your database does not have outstanding transactions or locks on the schema metadata.

13. Edit any other details of the stored procedure as required.

    You have specified the parameters for a stored procedure style operation binding.

14. Save the project.

More Information:

Operations Bindings Editor (see page 153)
Operation Bindings – Stored Procedure Editor (see page 154)
Create Operation Binding Dialog (see page 116)

Bind Operations to Scripts

You can bind operations to scripts to specify actions that you want to occur, before, after, or instead of, standard account CRUD operations, such as Add, Modify and Delete. Other types of operations are available, depending on the type of operation binding you select. You can bind an operation to a specific function in a global script, or bind an operation to an individual script. You can bind operations to scripts for any endpoint types which permit them (such as JNDI and JDBC).

To bind operations to scripts

1. Create a Connector Xpress JDBC or JNDI project.
2. Map the user account class, and any other class as required.
3. Click the Operations Bindings node.

   The Operation Bindings Editor appears.

4. In the Object Classes Filtering list, select an object class.

   You have specified the object class you want to create the script binding for.

5. Click Create.

   The Create Operation Binding dialog appears.

6. Select a class in the Available object classes list and add it to the Added object classes list.

   You have specified the object classes you want to apply the script binding to.

7. Under Type, select Script.

8. In the Available Operations list, select an operation.

   You have specified the type of operation that you want to bind the script to.

9. In the Timing list, select a timing.

   You have specified when the script binding is executed.

10. Click OK.

    A node is added to the mapping tree which displays the type of operation, the timing you selected and the name of class you want the script binding to apply to.

11. Click the node that displays the information about the script binding you created.

    The Script Editor dialog appears.

12. To bind a function in a global script to an operation, do the following:

    a. Select Execute a function in a global script.

    b. Select a global script from the Global Script list.

       Selecting a global script specifies the script where the function you want to bind to the operation is located.

    c. In the Function name field, type the name of the function.

       You have specified the function you want to bind to the operation.

13. To bind an operation to an individual script, do the following:

    a. Select Execute an individual script.

    b. Click Edit Script.

       The Edit Script dialog appears.

    c. Load or paste the script into the Edit Script dialog as required.

    d. Click OK.

14. Edit any other details of the script binding as required.

    You have specified the parameters for a script binding style operation binding.

15. Save the project.

Import Operation Bindings

If you created operation bindings in another project and you want to reuse them, or if you simply want to restore operation bindings from a backup, you can import operation bindings from an XML file.

When you import operation bindings, Connector Xpress imports operation bindings and any references to stored procedures.

To import operation bindings

1. Select Metadata, Import Operation Bindings.

   The Import Operation Bindings dialog appears.

2. Navigate to the operation bindings XML file and select Open.

   The imported operations bindings and all associated scripts or references to stored procedures appear under the Operation Bindings node.

Export Operation Bindings

If you plan to reuse operation bindings in another project or if you simply want to back up your operations bindings, you can export operation bindings to an XML file. When you export operation bindings, Connector Xpress makes a copy of all operation bindings and any scripts or references to stored procedures.You can create a copy of a live endpoint type and that contains all operation bindings.

To export operation bindings

1. Select Metadata, Export Operation Bindings.

   The Export Operations Bindings dialog appears.

2. Specify the folder to save the XML file in and the name of the exported operation bindings.

   Your operation bindings are saved.

   When you export operation bindings, Connector Xpress automatically encapsulates scripts in CDATA sections in the exported XML file. You can easily cut-and-paste from exported XML files without having to worry about XML quoting issues such as "<" => "<" and "&" => "'".

Stored Procedure and Column Considerations

This section covers some recommendations for your database schema to ensure smooth operations with stored procedures.

Stored procedure arguments that do not have a common SQL type reported by their drivers (that is, the driver returns java.sql.Type.OTHER = 1111) are treated as Type.VARCHAR (16), assuming that the driver converts from string to the argument's desired native type. For example, this works for Oracle NVARCHAR2 arguments. If this does not work, your stored procedure is not invoked and a failure message ending with "Invalid column type" appears.

We recommend that you use basic types (for example, basic types related to VARCHAR) for stored procedure arguments where possible, and verify other argument types against a single stored procedure against your desired vendor and version before proceeding to wide spread usage.

Verify that the native type of account/group table column you select as the key when creating a Group-Account association matches the type of corresponding columns in your chosen membership table. If they do not match, then membership information is not retrieved successfully. We recommend that you use strict constraints as much as possible. For example, if the column you select as the group naming attribute is of type NVARCHAR2 and the matching column is of type VARCHAR2 in the membership table, then looking up the groups an account belongs to returns an empty list (or at least is missing groups which have multibyte characters in their names).

We recommend that you do not use the percent (%) and underscore (_) characters because they act as wildcard characters when searching for database objects like schemas, tables, table columns, and stored procedure arguments. These values are quoted where they do appear, but this is an area of considerable divergence between vendors and versions. For example, some vendors do not report the quotation character used in some releases correctly. There are no longer any known problems with %/_ for any of our supported vendors.

Important! Connector Xpress and the Java CS read and write data from stored procedures through their arguments. However, they do not verify the validity of the code of any stored procedures you bind to. We recommend that you verify the validity of the code of any stored procedures you bind to.

Pure Scripted Connectors

The fully functional SDK script connector bundled with this SDK is an example of a 100 percent scripted connector.

If you do not want to use a ready-made connector, you can create your own pure scripted connector using a templates provided by Connector Xpress. You can use the following two templates to create a pure scripted connector:

- SDK DYN Script
- SDK DYN UPO Script

Using this template as a starting point, you can invoke your own JavaScript functions for each mandatory operation. This functionality relies on the 'instead of' operation binding. Before the Java CS performs any operation, it checks to see whether there are any operation bindings that tell it to invoke some logic before, after, or instead of the operation. Use the 'instead of' operation binding to invoke a JavaScript operation for your own pure scripted connector.

You must create an 'instead of' operation binding for each of the mandatory operations; Add, Delete, Modify, Search, and Lookup.

You can 'hot deploy' a pure scripted connector, which means that you do not have to change the Java Connector Server for the new connector to become operational.

A hot deployed connector specifies its connector.xml content as part of its metadata 'connectorXML' at the namespace level (accounting for proper XML encoding of this value). This means that a scripted connector can be created on the fly on a running Java CS without needing to restart it, or adding a static connector.xml on the Java CS host. Also, any connector configuration changes that are typically specified in connector.xml can be made active without a Java CS restart such as the connection pool settings.

How to Create a Pure Scripted Connector

Using templates as a starting point, you can create your own pure scripted connector that invokes your own JavaScript functions for each mandatory operation (Add, Delete, Modify, Search, and Lookup). Do the following:

1. Create a project from a template. (see page 27)

   The two pure scripted connector templates are SDK DYN Script and SDK DYN UPO Script.

2. Map the attributes. (see page 47)

   The template gives you some attributes as a starting point but you can add more.

3. Create operation bindings for each of the mandatory functions. (see page 57)

   The template includes the five mandatory operations (Add, Delete, Modify, Search and Lookup), but you can add more.

4. Add your JavaScript operations.

   Note: The template provides basic scripts, but you will need to customize these.

5. Click the Endpoint type node at the top level of the mapping tree.

   The Endpoint Type Details dialog appears.

6. Click Load, then select the connector.xml configuration you want to load.

   Connector Xpress loads the connector.xml configuration in the Connector XML field.

7. In the Connector XML field, configure the following Connector XML settings:

   a. Change the value of "name" from the template default to a unique name.

   ```
   <Property name='name'>
   <Value>UniqueName</value>
   </Property>
   ```

   b. Change the value of connectorTypeName from the template default to a unique name.

   ```
   <Property name='connectorTypeName'>
   <Value>UniqueName</value>
   </Property>
   ```

   c. (Optional) We recommend that you look at the other configuration items in the Connector XML to fine-tune your connector configuration. For example:

   ```
   <Property name='defaultConnectorConfig'>
   ```

   Note: If you keep the template name for your new pure scripted connector, the Java Connector Server recognizes it. You can now deploy the pure scripted connector. It is best practice, however, to give your connector a new name which means you need to hot-deploy it to help ensure that the Java Connector server recognizes it and loads its settings. For more information on hot-deployment, see the *Java CS Programming Guide*.

8. Save the Connector Xpress project.
9. Deploy the connector.

How you Generate CA Identity Manager User Console Account Screens

CA Identity Manager supports metadata-based generation of role and screen definitions for the CA Identity Manager User Console.

You can create the account management screens for a specific dynamic endpoint type in the CA Identity Manager User Console. The account management screens let you manage the accounts, account templates, and endpoints on a specific endpoint type. To create the account management screens, you do the following:

1. Use Connector Xpress to create the presentation metadata (see page 66) that defines the layout of the account management screens in the User Console. You create the presentation metadata by grouping mapped attributes into logical groups and subgroups.

   These attributes appear as tabs and page sections in the account management screens. Connector Xpress saves the groupings you make in the metadata.

   The presentation metadata defines:

   - Where on the User Console account management screen the input control appears
   - The input control's label

2. Use the Role Definition Generator to generate:

   - The field, screen, tab, task, and role definitions from the presentation metadata you created in Connector Xpress.
   - The files required by the Identity Manager Server to provide account management for a specific endpoint type through the User Console.

3. Deploy the Identity Manager Server Configuration Files required by the Identity Manager Server.

4. Import the field, screen, tab, task, and role definitions into CA Identity Manager.

   Importing the field, screen, tab, task, and role definitions makes the account management tasks available in the User Console.

Role Definition Generator

The Role Definition Generator is a stand-alone utility that generates the files needed by the Identity Manager Server to provide account management for a specific endpoint type through the User Console.

The Role Definition Generator is installed with the Identity Manager Server in the following directories:

- (Windows) %PROGRAMFILES%\CA\Identity Manager\IAM Suite\Identity Manager\tools\RoleDefinitionGenerator\bin
- (UNIX) /opt/CA/Identity_Manager/IAM_Suite/Identity_Manager/tools/RoleDefinitionGenerator/bin

Role Definition Generator Command

Valid on Windows and UNIX

The Role Definition Generator command parses the endpoint type metadata generated from Connector Xpress and generates the following file:

- **endpoint type.jar**–Contains the JIAM mapping files, framework, managed object definition files, resource bundle file and task role and screen definition file.

  This command has the following format:

  - (Windows) RoleDefGenerator.bat [d *domain* -l -h *hostname* -o *directory* -n -p *port* -u *username* -y *password_file.txt*] [endpoint_type ...]
  - (UNIX) RoleDefGenerator.sh [d *domain* -l -h *hostname* -o *directory* -n -p port -u *username* y *password_file.txt*] [endpoint_type ...]

  -d *domain*

  Specifies the CA Identity Manager domain. If not specified, the role definition generator defaults to the CA Identity Manager domain.

  -l

  Specifies that the Role Definition Generator lists endpoint types, but does not generate role definitions.

  -h *hostname*

  Defines the host name of Provisioning Server.

  -o *directory*

  Defines the output directory.

  Default: '.' that is, the current working directory.

  -n

  If specified, TLS is not used. TLS communication is enabled by default.

  -p *port*

  Specifies the Provisioning Server port number. If not specified, then 20390 is used, or 20389 is used if *–n* is specified.

-u *username*

Defines the Provisioning Server admin user name.

-y *password_file.txt*

Specifies the file that contains the Provisioning Server admin user password. If not specified, the utility prompts you for the password. The password file is in UTF-8 format. The first line of the file is used as the password.

Endpoint_*type*

Defines the name of the endpoint type (long form).

Example: List all endpoint types on a Provisioning Server

This example lists all endpoint types on a Provisioning Server:
RoleDefGenerator.bat -d *EXAMPLEDOMAIN* -h *im.example.com* -u *adminusername* -l

Example: Generate role definitions for a dynamic endpoint type

This example generates role definitions for *YourDynamicEndpointType.*
RoleDefGenerator.bat -d *EXAMPLEDOMAIN* -h *im.example.com* -u *adminusername YourDynamicEndpointType*

Account Screen Creation Example

This example shows you how to create the presentation metadata that defines the tabs and page sections in the account management screens in the User Console for a simple JNDI connector. This example creates account management screens for a dynamic endpoint type named MyJNDIEndpointType.

How you Generate Account Screens

To generate account management screens for the dynamic endpoint type MyJNDIEndpointType, do the following:

1. Use Connector Xpress to create the project that describes MyJNDIEndpointType. (see page 66)
2. Create the presentation metadata in Connector Xpress by grouping the mapped attributes into the logical groups and subgroups that you want to appear as tabs and page sections in the account management screens. (see page 66)
3. Use Connector Xpress to deploy the connector to the Provisioning Server. (see page 34)

4. Use the Role Definition Generator to generate the MyJNDIEndpointType.jar files.

   Identity Manager Server requires this file to provide account management for MyJNDIEndpointType.

5. Deploy the MyJNDIEndpointType.jar file to the Identity Manager Server.
6. Import the role and task definitions into the Identity Manager environment you want to manage the accounts for a specific endpoint type.

Presentation Metadata Example

The following example shows you how to group the attributes you have mapped for MyJNDIEndpointType into the logical groups and subgroups you want to appear as tabs and page sections in the account management screens in the User Console.

This example assumes that you have:

1. Created a project and specified a JNDI data source such as CA Directory for your connector. (see page 26)
2. Created a User Account class and mapped the Account ID and Last Name attributes in *inetOrgPerson* to the provisioning attributes *cn* and *sn* respectively.
3. Created a Group class and mapped the Account ID and Member attributes in *groupOfNames* to the provisioning attributes *cn* and *member* respectively. (see page 46)
4. Created a direct and reverse association between the Group and the User Account class and mapped the group's member attribute to the account's class naming attribute. (see page 48)

Example: Create the presentation metadata

To group the attributes you have mapped into the logical groups and subgroups you want to appear as tabs and page sections in the account management screens in the User Console, use Connector Xpress to create the presentation metadata.

To create the presentation metadata

1. In the Mapping Tree, click the Attributes node under the User Account node.

   The Attributes Summary dialog appears.

2. Under Account Screens, click Account.

   The Login page section appears.

3. Select the Account Id attribute from the drop-down list on the Login page section.

4. Under Account Screens, click User.

   The Name page section appears.

5. Select the Last Name attribute from from the drop-down list on the Name page section.

6. Under Account Screens, click Membership.

   The Membership page section appears.

7. Select the Member attribute from from the drop-down list.

8. Deploy the MyJNDIEndpointType connector to the Provisioning Server, then save the project.

9. Next, use the Role Definition Generator to convert the presentation metadata to the files required by CA Identity Manager to provide account management screens for MyJNDIEndpointType.

Role, Task, and Screen Definition File Example

Use the Role Definition Generator to generate the field, screen, tab, task, and role definitions from the presentation metadata you created in Connector Xpress and the files required by the Identity Manager Server to provide account management for a specific endpoint type through the User Console.

Example: Generate role, task, and screen definition files

To convert the presentation metadata to the files required by CA Identity Manager to provide account management screens for MyJNDIEndpointType, use the Role Definition Generator.

Valid on Windows and UNIX

To generate role, task, and screen definition files

1. Navigate to one of the the following directories according to your operating system:

   - (Windows) %PROGRAMFILES%\CA\IAM Suite\Identity Manager\tools\RoleDefinitionGenerator\bin
   - (UNIX) /opt/CA/IAM_Suite/Identity_Manager/tools/RoleDefinitionGenerator/bin

2. Open a command prompt window or a terminal window according to your operating system, then enter one of the following commands:

   - (Windows) RoleDefGenerator.bat -d *exampledomain* –h *im.exmaple.com* –p *password* –u *adminusername* MyJNDIEndpointType
   - (UNIX) RoleDefGenerator.sh -d *exampledomain* –h *im.exmaple.com* –p *password* –u *adminusername* MyJNDIEndpointType

(Windows and UNIX) The command generates the MyJNDIEndpointType.jar file.

The role, task, and screen definitions generated from the metadata include a basic Manager role and an Auditor (read-only) role the endpoint type you specified.

3. Next, deploy the MYJNDI.EndpointType.jar file to the Identity Manager Server.

Identity Manager Server Configuration Files Deployment Example

The following example shows you how to Deploy Identity Manager Server Configuration Files to the Identity Manager Server.

Example: Deploy Identity Manager server configuration files

To provide account management for MyJNDIEndpointType, deploy the MyJNDIEndpointType.jar file to the Identity Manager Server.

Valid on Windows and UNIX

To deploy Identity Manager server configuration files

1. Copy MyJNDIEndpointType.jar to one of the following directories:
   - (Windows) *app server home*/IdentityMinder.ear/user_console.war/WEB-INF/lib
   - (UNIX) *app server home*\IdentityMinder.ear\user_console.war\WEB-INF\lib

     Note: For WebSphere, copy the JAR file to: WebSphere_home/AppServer/profiles/Profile_Name/config/cells/Cell_name/applications/IdentityMinder.ear/deployments/IdentityMinder/user_console.war/WEB-INF
2. Repeat the preceding step for each node if you have a cluster.
3. Restart the Identity Manager Server.
4. Import the role and tasks settings into the Identity Manager environment.

Import Role and Task Settings into Identity Manager

The following example shows you how to import the role and task settings generated by the Role Definition Generator into CA Identity Manager.

Example: Import the role and task settings

To provide account management for MyJNDIEndpointType, import the role and task settings generated by the Role Definition Generator into CA Identity Manager.

To import the role and task settings

1. From the CA Identity Manager Management Console, click Environments.
2. Select the environment from which you want to manage accounts for a given endpoint type.
3. Click Role and Task Settings.
4. Click Import.
5. Select the endpoint types for which you want to import the screen, role, and task definitions, then click Finish.

   The status is displayed in the Role Configuration Output window.
6. Restart the CA Identity Manager environment.

   The account management screens for MyJNDIEndpointType are available in the User Console when you perform account management tasks such as creating and modifying accounts on an endpoint.
7. Identify users that are not members of the System Manager admin role.

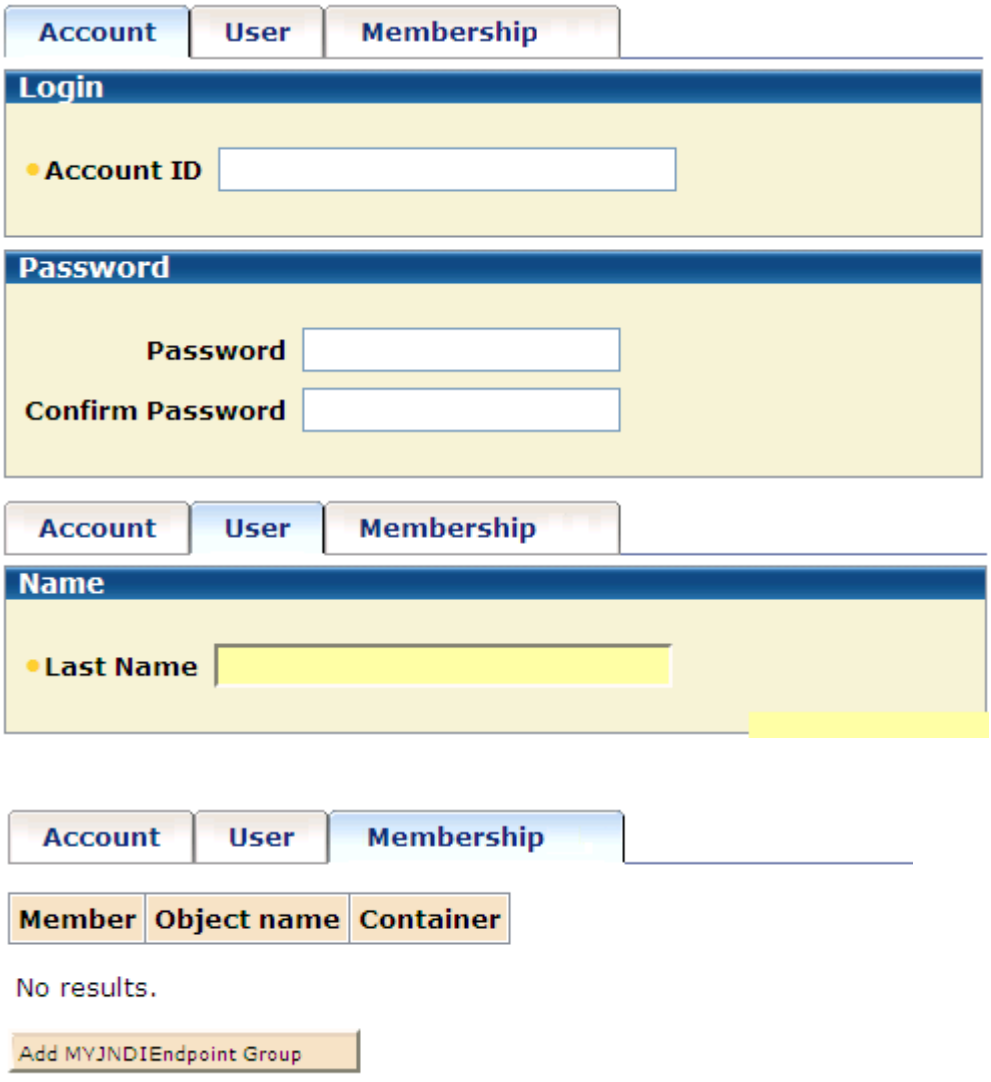
8. In CA Identity Manager, grant users that are not members of the System Manager admin role membership of the newly created auditor or manager admin roles for the specific endpoint type.

   This grants the users access to the Account tasks and Accounts tab.

   Members of the System Manager admin role see the new Accounts tab in the Modify User's Accounts and View User's Account admin tasks automatically.

Example: Generated account screens

This example shows you how the account management screens for the account management task look after you import the role and task definitions into CA Identity Manager.



Undeploy Role Definitions for an Endpoint Type

You can undeploy the role definitions for a given endpoint type from a CA Identity Manager environment where you previously imported role definitions

To undeploy role definitions for an endpoint type

1. Remove the endpoint-type-specific .jar from <IdentityMinder.ear>\user_console.war\WEB-INF\lib folder.
2. Restart the CA Identity Manager server.

The endpoint type is unregistered from the CA Identity Manager server and no longer appears in the CA Identity Manager User Console. You can no longer manage accounts or account templates for that endpoint type in the CA Identity Manager User Console. Removing the endpoint-type-specific .jar has no effect on objects which are on the Provisioning Server side, for example, account templates, endpoints and such for the endpoint type.

Promote a Connector from a Test to a Production Environment Example

In this example, Forwardinc has completed creating and testing a new Oracle connector that maps Oracle database tables to provisioning accounts and groups. Forwardinc has deployed the connector to their test and staging provisioning server, forwardinc_Test_PS. They now want to deploy the connector to a new production provisioning server, forwardinc_Prod_PS, and remove the connector from the test environment after deploying the connector.

Example: How forwardinc Promotes a Connector from a Test to a Production Environment

The following example shows you the process the administrator of forwardinc's CA Identity Manager installation follows to promote the connector from the test environment to the new production environment. The Administrator:

1. Saves the Oracle connector as a Connector Xpress project. For example, ORA_Connector.con.
2. Configures the new production provisioning server, forwardinc_Prod_PS.

   The connection details of the production provisioning server to which forwardinc wants to deploy the connector are specified.
3. Opens the Connector Xpress project ORA_Connector.con.
4. Creates an endpoint type on the forwardinc_Prod_PS provisioning server.

5. Specifies the managing connector server that manages the new endpoint type.

   Specifying the managing connector server helps ensure that requests for the endpoint type are routed to it.

6. Explores and correlates the new endpoint type.

   The exploration and correlation populates the forwardinc_Prod_PS provisioning server with data found in the endpoint.

7. Deletes the old endpoint type from the test provisioning server, forwardinc_Test_PS.

   The endpoint type, the associated account templates, and any child endpoints are deleted, however data on the endpoint is unchanged.

Chapter 4: Connector Xpress Utilities

This section contains the following topics:

What You Can Use Connector Xpress Utilities to Do

You can use Connector Xpress utilities to:

- Set up a Provisioning Server configuration (see page 73)
- Set up a Connector Server configuration (see page 75)
- Associate an endpoint with a connector server using different methods (see page 76)
- Update a Java CS password (see page 78)
- Export a mapping summary HTML file for an endpoint (see page 79)
- Add, populate, clean and remove dynamic endpoints (see page 82)
- Edit and deploy metadata (see page 87)
- Set managing CS for endpoint types (see page 76)
- Acquire, explore, correlate, and remove endpoint systems (see page 83)

Provisioning Server Configuration

Connector Xpress lets you manage connectors on a server by adding, editing, and removing Provisioning Servers.

Add and Configure a Provisioning Server

To manage connectors on a server, add and configure a Provisioning Server.

To add and configure a Provisioning Server

1. In the Provisioning Servers tree, right-click the Provisioning Servers node, then click Add Server.

   The Provisioning Server Details dialog appears.

2. Complete the fields in the dialog, then click OK.

   Connector Xpress adds the Provisioning Server to the Provisioning Servers tree.

   Note: If the Provisioning Server already has a Connector Server configured (see page 75), it appears under the Provisioning Server node.

More Information:

Provisioning Servers Tree (see page 16)
Provisioning Server Details Dialog (see page 159)

Edit Provisioning Server Details

If the details of your Provisioning Server change, for example, when you upgrade your hardware, edit the details of your Provisioning Server if necessary.

To edit Provisioning Server details

1. In the Provisioning Servers tree, right-click the server whose details you want to edit, then click Edit Server.

   The Provisioning Server Details dialog appears.

2. Complete the fields in the dialog, then click OK.

   Connector Xpress changes the details of the Provisioning Server.

More information:

Provisioning Servers Tree (see page 16)
Provisioning Server Details Dialog (see page 159)

Remove a Provisioning Server

If the details of your Provisioning Server change, for example, when you remove or replace your hardware, remove the details of your Provisioning Server from the Provisioning Servers tree if necessary.

To remove a Provisioning Server

1. In the Provisioning Servers tree, right-click the Provisioning Server that you want to remove, then click Remove Server.

2. When prompted, confirm that you want to remove the Provisioning Server.

   Connector Xpress removes the Provisioning Server.

More information:

Provisioning Servers Tree (see page 16)

Connect to a Provisioning Server

To connect to a Provisioning Server, provide a password for the Provisioning Server if necessary.

To connect to a Provisioning Server

1. Expand the Provisioning Servers tree, and then double-click the Provisioning Server you want to connect to.

   The Provisioning Server password dialog appears.

2. Type the password, then click OK.

   Note: If you enter an invalid password when attempting to connect to the Provisioning Server, the connection fails and Connector Xpress displays an error node on the Provisioning Servers tree. To reconnect with a different password, refresh the Provisioning Servers tree.

More information:

Provisioning Server Password Required Dialog (see page 160)
Provisioning Servers Tree (see page 16)

Connector Server Configuration

Connector server configuration lets you configure how your Provisioning Server routes individual endpoints to connector servers.

You can create multiple configurations and then associate the appropriate configuration with a given endpoint.

Note: Create the connector server configuration (see page 75) for the endpoint before deploying the connector.

Create a Connector Server Configuration

To specify how your Provisioning Server routes endpoint types or individual endpoints to connector servers, you create a connector server configuration.

To create a connector server configuration

1. Expand the server in the Provisioning Servers tree where you deployed your connectors.
2. Right-click on the CS Configs node and then select New CS Config.

   The Connector Server Configuration dialog appears.
3. Complete the fields on the dialog, then click OK.

   You have specified how your Provisioning Server routes endpoint types or individual endpoints to connector servers.

More information:

Provisioning Servers Tree (see page 16)
Edit Connector Server Configuration Dialog (see page 122)

How you Set a Managing Connector Server

You can select a server to manage a particular endpoint type or endpoint. When you have alternate Provisioning Servers and you have a different CS configured for each Provisioning Server, you can also associate two or more Connector Servers with an endpoint type.

You can associate an endpoint with a Java CS in either of the following ways:

- From the endpoint type node (see page 76)
- From the Java Connector Server on the CS Configs node (see page 77)

Set a Managing Connector Server from the Endpoint Type Node

You can select a connector server to manage a particular endpoint type or endpoint from the endpoint type node.

To set a managing connector server from the endpoint node

1. In the Provisioning Servers tree, right-click the endpoint type you want to manage, then click Set Managing CS.

   The Select Connector Servers dialog appears.
2. Select the Connector Server or Connector Servers from the list, then click OK.

   You have specified the Connector Server or Connector Servers you want to manage. Connector Xpress adds the endpoint type or endpoint to the managed branches of the selected Connector Servers.

Set a Managing Connector Server from the CS Configs Node

You can select a connector server to manage a particular endpoint type or endpoint from the CS Configs Node.

To set a managing connector server from the CS Configs Node

1. Expand the CS Configs node.
2. Right-click the CS Config you want to associate with the endpoint, then click Edit CS Configuration.

   The Edit Connector Server Configuration dialog appears.
3. Complete the following fields on the dialog, then click OK.

   Object Handle

   Defines the object handles of the endpoint types being managed on the selected Provisioning Server.

   Example:
   Namespace=dyn_jndi_democorp,Domain=forward,Server=Server

Edit a Connector Server Configuration

To change how your Provisioning Server routes endpoint types to a connector server, you can edit the connector server configuration.

To edit a connector server configuration

1. Expand the server in the Provisioning Servers tree where you deployed your connectors.
2. Expand the CS Configs node.
3. Right-click the server configuration you want to edit, and then select Edit CS Config.

   The Connector Server Configuration dialog appears.
4. Complete the fields on the dialog, then click OK.

   You have changed how your Provisioning Server routes endpoint types or individual endpoints to connector servers.

More information:

Provisioning Servers Tree (see page 16)
Edit Connector Server Configuration Dialog (see page 122)

Remove a Connector Server Configuration

To remove a connector server configuration, you can delete it from the Provisioning Servers tree.

To remove a connector server configuration

1. Expand the server in the Provisioning Servers tree where you deployed your connectors.
2. Expand the CS Configs node.
3. Right-click the server configuration you want to edit, and then select Delete CS Config.
4. When prompted, confirm that you want to delete the connector server configuration.

   The connector server configuration is removed.

More information:

Provisioning Servers Tree (see page 16)

Update a Java CS Password

To help ensure good security practice, you can update the Java CS password periodically. This also helps ensure that a Java CS is only being used by a known set of Provisioning Servers. For example, if you update a Java CS password and only update the provisioning servers that are currently using the JCS, then any other provisioning servers that are accessing the Java CS will no longer be able to access the Java CS after the next restart of the Java CS.

To update a Java CS password

1. In the Provisioning Servers tree, expand the CS Configs node.
2. Right click the CS Config you want to edit, then click Edit CS Config.

   The Edit Connector Server Configuration dialog appears.
3. Complete the fields on the dialog, then click OK.

More information:

Provisioning Servers Tree (see page 16)
Edit Connector Server Configuration Dialog (see page 122)

Mapping Summary File

You can use the mapping summary file to help you debug and read logs. For example, you can use the mapping summary file if the attribute names in the generic schema are not descriptive, for example, eTDYN-str-i-01.

Generating a mapping summary file is also important when you want to write batch scripts for automation.

The mapping summary file is an HTML file that displays the following information for a selected endpoint type, or a currently open project:

- Provisioning LDAP attribute name
- Logical name
- Native name
- Display name
- Attribute values for each class defined in the dynamic endpoint

Export a Mapping Summary File

You can export a mapping summary file to help you debug and read logs. For example, you use the file to look up endpoint mappings between Provisioning LDAP attribute names and native attribute/column names.

You can export a mapping file for the currently loaded project, or a selected endpoint type in the Provisioning Severs tree.

To export a mapping summary file

1. Do *one* of the following:
   - In the Provisioning Servers tree, right-click the endpoint type you want to export a Mapping Summary file for, then click Export Mapping Summary File.
   - Open the project (see page 30) you want to export a Mapping Summary file for, then click Metadata, Export Mapping Summary File.

     The Export Mapping Summary File dialog appears.
2. Specify the location where you want to save the file, then click OK.

   Connector Xpress saves the mapping summary file in HTML format to the location you specified.
3. Open the file in web browser and review the endpoint mappings.

Import XML Data Model File

To edit previously exported XML data, you can import an XML data model file. Importing an XML data file into an existing project replaces the existing metadata of the endpoint.

To import an XML data model file

1. Click Metadata, Import.

   The Import Data Model from XML file dialog appears.

2. Navigate to the location of the XML file you want to import, then click OK.

   Connector Xpress deploys the new data model and replaces the existing metadata of the endpoint.

Example: How an Administrator Imports and Deploys Metadata using an XML Data File

In this example, the administrator wants to replace the metadata on the endpoint *dyn_jdbc_endpoint* with metadata in the XML data file *export_dyn_jdbc_endpoint2.xml.* The administrator does the following:

1. Creates new metadata in a Connector Xpress project.
2. Saves the metadata in project named *dyn_jdbc_endpoint2.con*.
3. Exports the metadata as an XML data file named *export_dyn_jdbc_endpoint2.xml*.
4. Opens the *dyn_jdbc_endpoint.con* project file.

   This file contains the metadata for the endpoint *dyn_jdbc_endpoint* that the administrator wants to replace.

5. Imports the *export_dyn_jdbc_endpoint2.xml* data file into the *dyn_jdbc_endpoint.con* project.
6. Deploys the merged data model to the endpoint named *dyn_jdbc_endpoint.*

   Connector Xpress deploys the new data model and replaces the existing metadata on the *dyn_jdbc_endpoint*.

Export Data Model to an XML File

You can export the metadata for the currently open project into an XML file.

To export a data model to an XML File

1. Open the project (see page 30) you want to export the metadata for.

2. Click Metadata, Export.

   The Export Data Model to XML file dialog appears.

3. Specify the location where you want to save the file, then click OK.

   Connector Xpress exports and saves the metadata in the file you specified.

Merge a Modified Data Model

You can select a data model XML file to load and merge into the currently loaded metadata. Merging an XML data file adds the additional account attributes to metadata for the endpoint.

To merge a modified data model

1. Open the existing project (see page 30) you want to merge the metadata for.
2. Click Metadata, Merge.

   The Merge Data Model from dialog appears.

3. Navigate to the location of the file you want to merge the metadata with, then click OK.

   Connector Xpress overwrites or appends the data model file of the file you opened to the currently loaded data model.

Example: How an Administrator Merges the Metadata on an Endpoint using an XML Data File

In this example, the administrator wants to add some additional account attributes in one project to the metadata in another project. The administrator merges the metadata in the *dyn_jdbc_endpoint2.con* project with the metadata in the *dyn_jdbc_endpoint.con* project.

The *dyn_jdbc_endpoint.con* project contains the metadata for the endpoint named *dyn_jdbc_endpoint*. The administrator does the following:

1. Exports the metadata for the *dyn_jdbc_endpoint2.con* project to an XML data file named *export_dyn_jdbc_endpoint2.xml.*
2. Opens the *dyn_jdbc_endpoint.con* project.
3. Imports *export_dyn_jdbc_ns2.xml* into the *dyn_jdbc_endpoint.con* project.
4. Uses the Metadata Editor to verify that Connector Xpress has successfully merged the additional account attributes created in the *dyn_jdbc_endpoint2.con* project with the metadata in *the dyn_jdbc_endpoint.con* project.

5. Deploys the merged data model to the endpoint named *dyn_jdbc_endpoint.*
6. Uses the Provisioning Manager to verify that the new mappings are correct.

Managing Endpoints

To manage the accounts, users, groups, and other resource objects in your environment, you do the following:

- Acquire the endpoints
- Explore the endpoint for the objects you want to manage
- Correlate the objects to global users

You can specify how an endpoint is going to be explored and correlated. For example, you can specify how many levels in the container are explored, one level or the only the current subtree.

You can choose whether the correlate function creates any users that are not present or whether it associates accounts with no matching users to the [default user] global user.

You can also create, clean, and delete endpoint types.

How You Acquire and Manage Endpoints

Acquiring an endpoint automatically populates the Provisioning Server with accounts and other objects found in the acquired endpoint. An acquired endpoint is any application or computer managed by the Provisioning Server.

Acquiring an endpoint consists of creating it, exploring it, and correlating it. Correlating an endpoint is the process of examining accounts on it, and potentially creating users automatically and setting global user attributes from account attributes.

To manage an endpoint using Connector Xpress, do the following:

1. Create the endpoint in the Provisioning Server. When you register an endpoint, you are declaring it as an endpoint managed by the Provisioning Server.
2. Explore its contents. When you explore an endpoint, Connector Xpress finds the objects in the endpoint and stores a record of them in the Provisioning Server.

3. Correlate the explored accounts. When you correlate accounts on an endpoint, the Provisioning Server associates them with a global user in the Provisioning directory. You can choose whether the correlate function creates any global users that are not present or whether it associates accounts with no matching global user to the [default user] global user.
4. Update global user fields from accounts.

   Whether you chose to let correlate function create missing users, you can optionally run an additional function that sets (or refreshes) selected user attributes using the account's attribute values that you select.

More Information:

Explore/Correlate Endpoint Dialog (see page 130)
Create New Endpoint Dialog (see page 115)

Acquire, Explore, and Correlate an Endpoint

To acquire an endpoint, first register it in the Provisioning Server, that is, specify the connection details and credentials of the endpoint so that the connector can communicate with it. Then explore the endpoint for the objects you want to be managed, and correlate the objects to global users.

To acquire, explore and correlate an endpoint

1. In the Provisioning Servers tree, right click the Endpoint node that you want to explore or correlate, then click Acquire Endpoint.

   The Create New Endpoint dialog appears.
2. Complete the fields on the Create New Endpoint dialog, then click OK.

   You have specified the connection details and credentials of the endpoint.
3. In the Provisioning Servers tree, right click the endpoint node that you created then click Explore/Correlate endpoint.

   The Explore/Correlate endpoint dialog appears.
4. Complete the fields on the Explore/Correlate Endpoint dialog, then click OK.

   You have specified how Connector Xpress explores and correlate the endpoint.

More Information:

Create New Endpoint Dialog (see page 115)
Explore/Correlate Endpoint Dialog (see page 130)

Create an Endpoint Type

You can create an endpoint type using the loaded metadata.

Note: You can only create an endpoint type when a Connector Xpress project is open.

To create an endpoint type

1. Open an existing project (see page 30).
2. In the Provisioning Servers tree, right-click the Endpoint Types node, then click Create new Endpoint Type.

   The Create New Endpoint Type dialog appears.
3. Complete the fields on the dialog, then click OK.

   You have defined the name of your new endpoint type and added the endpoint type to the Endpoint Types node of the Provisioning Server you specified.

More information:

Create New Endpoint Type Dialog (see page 114)

Delete an Endpoint Type

You can delete an endpoint type using Connector Xpress when you no longer want to manage that endpoint type and the endpoints under it in the Provisioning Server.

Important! We strongly recommend that you only use Connector Xpress to remove endpoint types associated with dynamic connectors, to help ensure successful deletion.

To delete an endpoint type, expand the Endpoints Type node in the Provisioning Servers tree, right-click the endpoint type you want to delete, then click Delete Endpoint Type.

Connector Xpress deletes the endpoint type, the associated account templates, and any child endpoints. Data on the endpoint is unchanged.

Clean an Endpoint Type

When you want to delete all account templates and endpoints from a selected endpoint type, you can clean the endpoint type. Cleaning an endpoint type removes all endpoint objects from the Provisioning Server, and recreates the default account template object.

To clean an endpoint type

1. In the Provisioning Servers tree, right-click the endpoint type you want to clean, then click Clean Endpoint Type.

   The Clean Endpoint Type Objects dialog appears.

2. Confirm that you want to clean the endpoint type.

   Connector Xpress deletes all account templates and endpoints of the selected endpoint type from the Provisioning Directory. Data on the endpoints is not affected.

Delete an Endpoint

You can remove a connector using Connector Xpress when you no longer want to manage that connector in the Provisioning Manager.

Important! We strongly recommend that you only use Connector Xpress to remove endpoints associated with dynamic connectors, to help ensure successful deletion.

To delete an endpoint, expand the Endpoints node in the Provisioning Servers tree, right-click the endpoint you want to delete, then click Delete Endpoint.

Connector Xpress deletes the connector, and removes the endpoint and associated account templates, but the data on the endpoint data is unchanged.

More information:

Managing Endpoints (see page 82)

Configuration Data Location

The configuration data created using Connector Xpress is stored in various locations throughout the distributed system as described in the following sections.

Java CS Routing Rules Location

The Java CS routing rules are located in the following location:

eTSAName=[CSNAME],eTSAContainerName=SAs,eTNamespaceName=CommonObjects,dc=[DOMAIN],dc=eta

Endpoint Type Metadata Location

The endpoint type metadata is located in the following location:

eTNamespaceName=[NAMESPACE],dc=[DOMAIN],dc=eta

Endpoint Type Metadata and Java CS Routing Rules

The endpoint type metadata and Java CS routing rules are stored on the Provisioning Server.

View Endpoint Type Metadata and Java CS Routing Rules

To view the endpoint type metadata and Java CS Routing Rules, connect to the Provisioning Server using an LDAP browser such as JXplorer.

Java CS Routing Rules Hierarchy

Each CS has an entry under the following:

eTSAName=*CS name*,eTSAContainerName=SAs,eTNamespaceName=CommonObjects,dc=*server*,dc=eta

For each endpoint type managed by the CS, there is an entry similar to the following:

eTSABranchDN : eTNamespaceName=Oracle Server,dc=*server*

Endpoint Type Distinguished Name

The endpoint type's distinguished name is as follows:

eTNamespaceName= *MyEndpointType*,dc=*Domain*,dc=*TopLevelProvisioningServer*

The following attributes are used to store metadata on each dynamic endpoint type object:

eTMetaData

(Mandatory) Stores the XML document containing all wizard mappings, excluding stored procedures and operation bindings.

eTOpBindingsMetaData

(Optional) Stores the XML document containing stored procedures and operation bindings.

Connector Xpress Local Storage

Connector Xpress project files (.con files) are stored on the local file system.

Connector Xpress data source configurations are stored under Connector Xpress user preferences.

Note: You cannot export Connector Xpress data source configurations.

More Information:

Projects (see page 23)
Set Preferences (see page 22)
Where User Preferences Are Stored (see page 21)

Edit Metadata

You can edit the metadata for a project.

Important! Use the metadata editor at your own risk. Removing or altering important objects in the metadata can render it inoperable.

To edit metadata

1. Open an existing project (see page 30).

2. In the Provisioning Servers tree, right-click the endpoint type you want to edit the metadata for, then click Edit Metadata.

   The Edit Metadata for Endpoint Type dialog appears and displays the metadata in an XML tree.

3. Edit the metadata as required, then click Save.

   Connector Xpress redeploys the metadata.

Example: How an Administrator Modifies Metadata Manually

In this example, an administrator wants to change the display name mapping from last name to first name by manually modifying the metadata without using the mapping tree. The administrator:

1. Opens the dyn_jdbc_namespace.con project file.
2. Right-clicks the dyn_jdbc_namespace.con endpoint, then selects Edit Metadata.
3. Expands the dyn_jdbc_namespace node, and navigates to Classes, eTDYNAccount, Properties, eTDYN-str-01 in the Metadata Editor.
4. Modifies the displayName attribute from Last name to First name.
5. Modifies the beanPropertyName attribute from lastName to firstName.
6. Redeploys the modified metadata.
7. Checks that the changes have been made successfully by validating them in the Provisioning Manager.

More information:

Projects (see page 23)
Provisioning Servers Tree (see page 16)
Edit Metadata for Endpoint Type Dialog (see page 123)

Enumerated Values

For static enumerations, defining a new type involves:

- Creating a new unique type name
- Constructing a list of known values, each consisting of an internal value, and an optional display value

Note: Naming your new enum type int or integer does not cause any problems, however it can make maintenance of the metadata more difficult.

For query-based enumerations, define:

- A mapping to an unmanaged class

- The attributes the objects of that class have available for querying
- An association between a managed class and unmanaged class, through an appropriately chosen attribute in the former class

Metadata and Operations Bindings Editors Considerations

Important! You can use the Metadata editor to verify that the mappings you have made, however we recommend that you use the Metadata editor at your own risk. Removing or altering important objects in the metadata can render it inoperable.

Redeploy Metadata

When a connector project is open and metadata is available, you can push the metadata to the endpoint type on the Provisioning Server. You can only deploy metadata to dynamic endpoint types.

Note: If the endpoint type already has metadata, the new metadata cannot alter fundamental aspects of the data model such as adding attribute names or types.

To redeploy metadata

1. Open an existing project (see page 30).
2. In the Provisioning Servers tree, right-click the endpoint type you want to deploy data to, then click Deploy Metadata.

   The Deploy Metadata dialog appears.
3. Confirm that you want to deploy the metadata.

   Connector Xpress deploys the metadata to the selected endpoint type and overwrites any existing metadata.

More information:

Projects (see page 23)
Provisioning Servers Tree (see page 16)

DYN Schema Extensions

The DYN Schema has been extended for the following attributes:

- Container objects (eTDYNContainerXXX) are extended to 10.
- Generic objects (eTDYNObjectXXX) are extended to 35.

- All capability attributes (for example, eTDYN-str-multi-ic-, eTDYN-str-ic-, eTDYN-str-c-, eTDYN-bool-c-, eTDYN-int-multi-c-, eTDYN-int-c-) are extended to 99, with the exception of the multivalued case-sensitive attributes (eTDYN-str-multi-c-), which are extended to 500.
- The noncapability multivalued case-sensitive or case-insensitive attributes (for example, eTDYN-str-multi-, eTDYN-str-multi-i-) are extended to 500.
- The generic cached (that is, data location equals BOTH) multivalued case-sensitive attributes (eTDYN-str-multi-ca-) are extended to 99. These attributes are included in every DYN object.

  Cached attributes have their values stored on the provisioning server and are therefore accessible without contacting the endpoint system. Where their values are provided by the endpoint system, an explore operation is needed to update the values stored for them to match that on the endpoint system.
- Ninety-nine connection-specific cached multivalued case-sensitive attributes (eTDYN-str-multi-ca-sec-) are added for DYN Directory only. These attributes are added in the "encryptwith" line of the DYN password attribute, which can be used to obfuscate sensitive settings.

Metadata Descriptions

For more information about the metadata used in all layers of the architecture including Connector Xpress, JIAM, and the Java CS, see the JavaDocs in the CA Identity Manager Bookshelf.

Using Connector Xpress as a Generic Metadata Editor

You can use Connector Xpress as a generic editor for non-JNDI or JDBC endpoint types. Using Connector Xpress as a generic editor lets you edit generic metadata that is not intended for a particular implementation bundle. For example, to edit metadata for the Scripting connector, create classes and attributes from the ground-up that are not mapped to a particular schema.

You can create and edit generic metadata in the following ways:

- Create new attributes (see page 91), rather than selecting discovered attributes in the attribute mapping table.
- Load a set of predefined custom operational attributes (see page 92) from a predefined dictionary file.
- Load selected operational attributes from a predefined dictionary file (see page 92).
- Load an extended set of metadata properties from a data file. (see page 91)

Create New Attributes

You can use Connector Xpress as an editor for generic metadata, and create new attributes, rather than selecting discovered attributes in the attribute mapping table. You can use this procedure to define compound type attributes or any other connector-specific attribute that is not mapped to an endpoint attribute.

To create new attributes

1. Create a project (see page 26), and when prompted to select a Data Source, select No source.

   The Endpoint Type Details dialog appears.

2. Complete the fields on the Endpoint Type Details (see page 129) dialog.
3. Expand the Classes node, then click the Attribute node of the class you want to map.

   The Attributes Summary (see page 112) dialog appears.

4. Click Add, and type a name for the attribute.

   Connector Xpress adds the new attribute to the mapping tree.

5. Expand the Attributes node in the mapping tree, then click the new node for the attribute.

   The Attribute Details dialog appears.

6. Complete the fields on the Attribute Details dialog.

Display Extended Set of Metadata Properties

You can display an extended set of metadata properties on various dialogs.

To display extended set of metadata properties

1. Create a project (see page 26), or open an existing project (see page 30).
2. Click Tools, Preferences.

   The Connector Xpress Preferences dialog appears.

3. Select the Show Extended Metadata check box.

   Connector Xpress displays the extended metadata properties and contains all additional metadata that is relevant to this data model item.

   Note: For information about the fields displayed, see Extended Metadata Properties.

More Information:

Endpoint Type Details Dialog (see page 129)
Map Class and Attributes Dialog (JNDI) (see page 140)

How you Load Operational Attributes

Connector Xpress does not automatically load operational attributes, however for JNDI endpoint types, you can load operational attributes using either of the following methods:

- Load a dictionary of custom operational attributes from a dictionary (see page 92)

  Each dictionary is based on a particular vendor or standard. You can select the following dictionaries:

  - CA Directory
  - RFC 2252
  - X.500

- Add a selected operational attribute (see page 92) to the list of all attributes of the selected structural native class.

Load Operational Attributes from a Dictionary

For JNDI endpoints, you can load a set of predefined custom operational attributes from a dictionary.

You can map operational attributes like other attributes.

To load operational attributes from a dictionary

1. Create a project (see page 26), or open an existing project (see page 30).
2. Click Project, Settings.

   The Edit Project Settings dialog appears.
3. Select the dictionary you want to load the operational attributes from, then click OK.
4. Click the Class node of the class you are mapping.

   The Map Class and Attributes dialog appears. Connector Xpress displays the operational attributes in the Map Object Class Attributes table.
5. Map the operational attributes as required.

Load Selected Operational Attributes from a Dictionary

For JNDI endpoints, you can load selected operational attributes from a dictionary.

To load selected operational attributes from a dictionary

1. Create a project (see page 26), and when prompted to select a Data Source, select No source.

   The Endpoint Type Details (see page 129) dialog appears.

2. Click Project, Settings.

   The Edit Project Settings dialog appears.

3. Select the dictionary you want to load the operational attributes from, then click OK.

4. Expand the Classes node, then click the Endpoint Class node.

   The Endpoint Class (see page 128) dialog appears.

5. Select the dictionary attribute you want to load from the Add Dictionary Attribute list, then click Add.

   The attribute you selected is added to the mapping tree.

6. Click the attribute you created in the mapping tree.

   The Attribute Details dialog appears.

7. Complete the fields on the Attribute Details dialog.

Chapter 5: Troubleshooting

This section contains the following topics:

Setting a Managing Connector Server for a Dynamic Endpoint Fails

Valid on Windows and Unix

Symptom:

Choosing a C++ Connector Server to manage your DYN endpoint fails, and an error message is displayed.

The most common reason for this failure is a mismatch of the type of endpoint type (for example, DYN) and the managing Connector Server (for example, C++ CS).

Solution:

Select a Java Connector Server to manage the dynamic endpoint type.

ID/Sequence Column Support

Connector Xpress does not currently provide mapping support of any ID/Sequence columns, for example, auto-increment Object ID column.

Failed to Insert because IDENTITY_INSERT is Set to Off

Valid on MSSQL only

Symptom:

The message Failed to Insert because IDENTITY_INSERT is Set to Off appears when you create an account. This message occurs because your database table has IDENTITY_INSERT set to off.

Solution:

Write a stored procedure to create an account, and write scripts that set IDENTITY_INSERT on before the insert operation and then sets IDENTITY_INSERT off. Do the following:

1. Write a one-line stored procedure with timing=PRE (Before) that sets IDENTITY_INSERT [database. [owner.]] { table } on.
2. Write another one-line stored procedure with timing=POST (After) that sets IDENTITY_INSERT [database. [owner.]] { table } off.

Multiple Foreign Key Constraints

If the database table mapped to accounts is associated with multiple foreign-key constraints, do either of the following:

- Write a stored procedure to delete the account and the associations the account has with other tables.
- Configure a cascading delete in the database schema.

For example, in a company inventory database, the foreign keys in the customer group table and customer order table reference a customer account table. Customer account is mapped to account and customer group is mapped to account group in Connector Xpress. When you delete an account, Connector Xpress automatically deletes any entries in the account group that relate to this account. However, Connector Xpress does not delete the customer order table. The delete operation fails because of a violation of the foreign-key constraint.

To resolve the problem, do either of the following:

- Write a stored procedure that removes any entries that are associated with the account that you want to delete *before* the actual account is deleted.
- Configure a cascading delete on the customer account, if the database vendor or schema supports this feature.

The Role Definition Generator Will Not Run

Valid on Windows and UNIX

Symptom:

When I try to run the Role Definition Generator I get the following error message:

java.lang.UnsupportedClassVersionErrorException

This occurs when an older version of the Java JRE appears before the new version in the path.

Solution:

Verify that Java 5 JRE appears before older versions in the path.

Tuning the Database Used by the Provisioning Server

We recommend that you tune your Provisioning Server. If you do not tune the Provisioning Server, its performance can suffer when there are large numbers of objects stored in dynamic namespaces (JDBC and JNDI).

Use the CA Directory dxtunedb command if you add many objects to the Provisioning Directory. With the introduction of the dynamic namespace option, execute the following Ingres commands following each dxtunedb command:

- optimizedb –zr499 –zu240 <database name> -rsubsearch –acid
- optimizedb –zr499 –zu240 <database name> -rdit –aparent –ardnkey –aeid
- sql <database name> > set trace point RD010 \g

Support for Binary-type Attributes

If you are performing DYN mappings in Connector Xpress to perform account management using CA Identity Manager, be aware that CA Identity Manager does not support binary-type attributes for DYN endpoint types.

As a result, any attributes that you specified as having a binary data type in Connector Xpress are not displayed on CA Identity Manager account management screens.

We recommend that you use Provisioning Manager to manage binary attributes on DYN endpoint types.

Support for Mandatory Attributes on a JDBC Endpoint

To support attributes that the JDBC endpoint defines as mandatory (that is, NOT NULL), but do not need to be mandatory from the provisioning point of view, the JDBC connector allows you to use an empty value for the provisioning attribute.

When you create an account and are mapping a mandatory attribute on the endpoint to a provisioning attribute that is non-mandatory, for example, a description field, and you do not enter a value for the description field, the JDBC connector maps the empty provisioning attribute to the endpoint attribute by populating the NOT NULL column on the endpoint database with spaces.

To support mapping of non-mandatory fields in Provisioning Manager, the Java Connector Server contains a NullValueClassConverter that converts empty values in non-mandatory provisioning attributes to spaces in mandatory attributes on the endpoint.

For example, empty values can occur on a legacy database system has a description field that is NOT NULL on the table being mapped to a user account. However, the description field is not set to mandatory in the Provisioning Manager, which means you do not have to enter a description for a user to create an account. The NullValueClassConverter is used to store the empty value.

The property *nullValue* in the *pluginConfig* class in the NullValueClassConverter is set to a space by default. You can change *nullvalue* to other values, but changing *nullvalue* requires additional configuration of the endpoint database (see page 101).

For JDBC endpoints, the Java Connector Server loads the converter automatically. For other types of endpoints, for example, JNDI, manually configure the Java Connector Server to load the converter. (see page 100)

How to Enable Support for Mapping of Non-mandatory Fields in Provisioning Manager

To enable support for mapping of non-mandatory fields in Provisioning Manager to not-null columns in a JDBC database using the JDBC connector, add the metadata attribute *useSpecialNullValue* to the Connector Xpress Dyn mapping to each attribute that you want the *NullValueClassConverter* to convert. The Java Connector Server checks for the presence of this metadata attribute before enabling the converter.

Note: You can also use this procedure to enable support for other types of endpoints.

To enable support, do the following:

1. Use Connector Xpress or another editor to edit the metadata for your endpoint type.
2. Add the new Boolean metadata attribute *useSpecialNullValue* to the attribute that you want the *NullValueClassConverter* to convert, and set the Boolean value to true.

   Note: Add the metadata attribute *useSpecialNullValue* to the Connector Xpress Dyn mapping to each attribute that you want to convert. The Java Connector Server checks for the presence of this metadata attribute before enabling the converter.
3. Set the *isRequired* metadata attribute to false.
4. Repeat this procedure for all attributes that you want to convert.

Example: Enable support for mapping of non-mandatory fields in Provisioning Manager

This example uses Connector Xpress to edit the metadata. The Description attribute of your account is mapped to eTDYN-str-01.

To enable support for mapping of non-mandatory fields

1. In Connector Xpress, right-click the endpoint type you want to edit the metadata for, then click Edit Metadata.

   The Edit Metadata for Endpoint Type dialog appears.
2. Expand the eTDYN-str-01 node, then select the metadata subnode.
3. Click the Add button.

   Connector Xpress adds a new metadata attribute node to the tree.
4. Name the new metadata node *useSpecialNullValue*.
5. Select Boolean from the type drop-down list.
6. Select the check-box next to the drop-down list.

   The value of the *useSpecialNullValue* attribute is set to *true.*
7. Select the *isRequired* metadata attribute on eTDYN-str-01.
8. Clear the check-box next to the drop-down list.

   The value of the *isRequired* attribute is set to *false*.

How to Enable Support for Mandatory Attributes on Endpoints other than JDBC Endpoints

To support endpoints other than JDBC endpoints that define attributes as mandatory (that is, NOT NULL), but do not need to be mandatory from the provisioning point of view, do the following:

1. Configure the Java Connector Server to load the NullValueClassConverter. (see page 100)
2. Enable support for mapping of non-mandatory fields in Provisioning Manager. (see page 98)

Configure the Java Connector Server to Load the NullValueClassConverter

The NullValueClassConverter plugin is shipped with the JDBC connector. To use the NullValueClassConverter for endpoints other than JDBC endpoints, configure the Java Connector Server to load the converter.

Note: The converter is loaded automatically for JDBC endpoints.

Note: The property *metadataPropNames* has a value of *useSpecialNullValue.* Add the *metadataPropNames* metadata attribute to the Connector Xpress Dyn mapping to each attribute that plugin handles. The Java Connector Server checks for the presence of this metadata attribute before enabling the plugin.

To enable the plugin, configure the plugin in an override connector.xml for your endpoint.

To configure the Java Connector Server to load the NullValueClassConverter

1. Rename the file SAMPLE.connector.xml in C:\Program Files\CA\Identity Manager\Connector Server\conf\override\jdbc to connector.xml.

2. Add the following to your class plugin config of your connector.xml file:

```
<property name="classPluginConfigs">
 <list>
  <bean class="com.ca.jcs.cfg.MetaPluginConfig">

   <property name="pluginClass">
    <value>com.ca.jcs.converter.meta.NullValueClassConverter</value>
   </property>

   <property name="pluginConfig">
    <bean
class="com.ca.jcs.converter.meta.NullValueClassConverter$NullValueConverterConfig">
     <property name="nullValue">
      <value> </value>
     </property>
    </bean>
   </property>

   <property name="metadataPropNames">
    <list>
     <value>useSpecialNullValue</value>
    </list>
   </property>

  </bean>
 </list>
</property>
```

Note: The property *nullValue* in the pluginConfig class in the NullValueClassConverter is set to a space by default. To change the null value the plugin uses, change the nullValue in the config bean.

Change the Default Value Used to Store an Empty Value

The property *nullValue* in the pluginConfig class in the NullValueClassConverter is set to a space by default. To change the default value used to store an empty value, change the nullValue in the config bean in the class plugin config of your connector.xml file.

Mapping Against One Endpoint and Acquiring Against Another Endpoint

If you perform mapping against one endpoint and acquire a directory referring to another, verify that they have the same tables and stored procedures (JDBC) or LDAP schema (JNDI).

Table Attribute Mappings

When you define your table and attribute mappings, you can make the various choices in the Type field on the Map Class and Attribute dialog (see page 136). The choices you make define which predefined attribute is used and what the attributes behavior is.

If you select the Synchronized check box on the Attribute details dialog Connector Xpress determines the synchronization type from the attribute's data type automatically.

For example, the choices you make set the limits on how many predefined attributes there are for that type and behavior. The following table shows the limits:

Type and behavior	Limit
single-valued non-capability integer	10
single-valued capability integer	99
multivalued non-capability integer	10
multivalued capability integer	99
single-valued sensitive non-capability string	30
single-valued insensitive non-capability string	30
single-valued sensitive capability string	99
single-valued insensitive capability string	99
multivalued sensitive non-capability string	500
multivalued insensitive non-capability string	500
multivalued sensitive capability string	500
multivalued insensitive capability string	99
single-valued non-capability Boolean	10
single-valued capability Boolean	99
single-valued binary	10

Type Mapping

In Connector Xpress, an *edittype=int* maps to a plain LDAP string attribute, unless you explicitly request integer sync.

MySQL and Informix Stored Procedure Support

Connector Xpress does not support the following stored procedures from the following vendors:

- **MySQL**–MySQL stored procedure parameters are not accessible through the required JDBC method and therefore are not supported.
- **Informix**–Informix supports stored procedures with the same name but different parameters. This makes listing the stored procedures and their parameters problematic, and therefore are not supported.

JDBC Naming Attribute

You can only map the account and group naming attributes to columns which are required in the native table they are mapped to. That is, an attribute that does not allow null values. You are advised to verify that these columns act as primary keys for the tables containing them, or at the least, have a unique index applied to them.

Sybase Stored Procedures Failure

Valid on Windows and UNIX

Symptom:

The following exception is reported when invoking Sybase stored procedures:

Nested exception is com.sybase.jdbc3.jdbc.SybSQLException: Stored procedure 'AddObject_Instead_Parameter' may be run only in unchained transaction mode. The 'SET CHAINED OFF' command will cause the current session to use unchained transaction mode.

Solution:

The Java CS requires that Sybase stored procedures have the set chained option to on.

Set the chained option to on for all Sybase stored procedures.

Connector Xpress Logging

Connector Xpress log messages are written to following file:

im_home\Connector Xpress\logs\conxp-log.txt

Note: The generated metadata is usually the primary resource about the problem being diagnosed. We recommend that you analyze the generated metadata using the Connector Xpress log files.

Chapter 6: Screens and Dialogs

This section contains the following topics:

Attribute Details Dialog

The Attribute Details dialog lets you define logical parameters that control how a specific provisioning attribute is handled.

This dialog contains the following fields:

Name

Defines the name of the attribute. This label appears in user interfaces to represent the attribute.

Description

(Optional) Describes the attribute.

Connector Map To

Specifies which name to map an object class (including the connector itself) or attribute to in connector-speak. For a dynamic connector, this attribute specifies the name of the native system item to map the attribute to. For example, a JDBC-based connector would probably map an object class to the name of a database table, and each property within it to the name of one of its columns.

Default Value

(Optional) Specifies the default value for the attribute.

Note: Only clients use this value (for example, user interfaces). If the client does not supply a value, the Provisioning Server does not use it by default.

Required

If selected, specifies that the endpoint requires that this attribute is present in all objects of this type.

Note: If the endpoint schema requires this attribute to be present, Connector Xpress automatically selects the check box, and you cannot clear the check box.

Minimum Length

Specifies the minimum byte length of values for this attribute value.

This value is used for input validation in CA Identity Manager account screens.

Maximum Length

Specifies the maximum byte length of values for this attribute value.

This value is used for input validation.

Allowed Operations

Specifies the operations you can perform on this attribute. Select at least one of the following options:

Create

Specifies whether this attribute's value is set when creating an object instance of the parent class.

Read

Specifies whether this attribute's value is read when viewing an object instance of the parent class.

Modify

Specifies whether this attribute's value is set when modifying an object instance of the parent class.

Minimum Value

(Numeric attribute types) Specifies the minimum value you can set for this attribute.

Maximum Value

(Numeric attribute types) Specifies the maximum value you can set for this attribute.

Data type

Specifies the data type of the provisioning attribute that you have mapped to the native attribute.

Binary Data

Defines an attribute whose value is arbitrary binary data.

Boolean

Specifies logically true or false in XML, but represented by the Provisioning Server and JIAM APIs as 1 or 0 in LDAP attribute values.

Date

Specifies a date.

Example: 1999-05-31

Note: The Dynamic Namespace plug-in to Provisioning Manager supports the years from 1800 through 9999, but other components of the solution impose no such restrictions and can represent virtually any year in recorded history.

Date & Time

Specifies a particular time on a particular day.

Example: 1999-05-31T13:20:00

Note: The Dynamic Namespace plug-in to Provisioning Manager supports the years from 1970 through 2036, so you must use Date to represent days falling outside of this range.

Note: Vendor differences complicate how Connector Xpress handles time-related columns. For example, MSSQL "DATETIME" signifies a DateTime value whereas other vendors use the standard "TIMESTAMP", and MSSQL TIMESTAMPs are automatically generated binary values. Also, Oracle does not support a "TIME" type and its "DATE" type is also effectively a TIMESTAMP. Therefore, to remain vendor-neutral, Connector Xpress allows you to map to any of Date/DateTime/Time whenever it makes sense for you to do so.

Double-precision floating-point

Specifies a double-precision 64-bit floating-point value.

Enumeration - enumeration type name

Specifies an attribute with a fixed list of enumerated values.

Flexi-DN

Specifies a distinguished name string format.

For example, "cn=Bob,ou=Sales,o=ExampleCorp". The connector enforces this.

Flexi-Email

Specifies an email address string format.

Flexi-Quoteless

Specifies that quotes are removed from attribute values.

Floating Point

Specifies a single-precision 32-bit floating-point number.

Integer

Specifies a 32-bit value between -2147483648 and 2147483647.

Long Integer

Specifies a 64-bit value from 9223372036854775808 through 9223372036854775807.

String

Specifies an unrestricted field.

Time

Specifies an offset of between 0 seconds and 23:59:59.

Example: 13:20:00

Force Case

Specifies the case of the connector-speak value of a string.

Upper

Specifies that the case is set to upper case.

Example: If set to upper, *String* is set to *STRING*

Lower

Specifies that the case is set to lower case.

Example: If set to Lower, *STRING* is set to *string*.

Preserve

Specifies that the case of the string is preserved.

Note: The endpoint may not enforce case-sensitive searching, so a search for "EXAMPLE" may still match "eXaMpLe" on the endpoint system.

Multi-valued

If selected, specifies that this attribute is multi-valued.

JNDI: Selected only if the native attribute is multi-valued. Read-only if native attribute is single valued.

JDBC: Default is unselected.

Flattening Style

(JDBC only) Specifies the flattening style you want to use.

CSV (comma-separated value)

Specifies that the multiple values of the attribute are comma-separated.

SQL

Specifies that the multiple values of this attribute are in SQL format.

XML

Specifies that the multiple values of this attribute are in XML format.

Trim Whitespace

Specifies whether Connector Xpress trims leading and trailing whitespace from the attribute's value.

Sensitive

Specifies whether user interfaces use asterisks to mask the attribute's value.

Boolean Values

Overrides the default 0 and 1 symbolic values for false and true values for Boolean attribute types.

True

Specifies the symbolic value that is be used for *true.*

False

Specifies the symbolic value that is used for *false*.

Account Template Value (Mandatory for mandatory attributes in account classes)

Specifies the default value for this attribute used in the default account template.

Account Template Value

Specifies how the default value for this attribute is generated used in the default account template.

None

Specifies that this attribute does not have a value in the default account template.

By Value

Specifies the value of this attribute used in the default account template.

By Rule String

Specifies that the value of this attribute used in the default account template is generated by a rule string.

Note: For more information on Rule Strings, see the *Administration Guide.*

Synchronized

Specifies whether this attribute is updated when this account is synchronized with its templates.

Cached

Specifies that the attribute is stored by the Provisioning Server and can be retrieved without consulting the endpoint system.

For example, credentials used to connect to the endpoint system must be cached, otherwise the user might have to provide the credentials every time the connector is acquired. You may also need to cache a value because querying an endpoint system for the value can take a long time to compute.

Well Known Name

Specifies whether this attribute is a *well-known* attribute.

JavaBean Property Name

Specifies the JavaBean Property name used for this attribute in the JIAM API. If you select a value in the Well-Known Name list, you cannot edit this value.

Expensive Retrieval

Specifies that this attribute's value may be too expensive to return in general searches. If selected, searches only return the attribute value if explicitly asked for.

Converter Regex

Defines the regular expression used to convert the attribute value.

Converter Regex Style

Specifies the syntax of the regular expression used by Converter Regex

Values: perl5 or Java.

Validating Regex

Defines the regular expression used to validate user input.

Validating Regex Style

Specifies the syntax of the regular expression used by Validating Regex.

Values: perl5, Java

Validating Regex Hint

Specifies the help message displayed if the input value does not match the validating regular expression.

UI Field Length (String types only)

Specifies the recommended character width of the user interface input field for this attribute. If not used, Connector Xpress uses the attributes data type and maximum length to define an appropriate input field length.

Connector Generated

Specifies the generator expression used by the Java CS.

To pass the value to the endpoint system literally, enclose the expression in double quotes.

Example: \"NEXT VALUE FOR my_sequence\"\'

If you do not enclose the expression in double quotes, the connector must know how to interpret the value. For example, JDBC knows the value provided will be a sequence name.

Example: \'my_sequence\'

Is Connector Generated

Specifies that that the value for property is generated implicitly by the endpoint. For example, *true* for IDENTITY columns in JDBC.

Extended Properties

Displays an extended set of metadata properties. These fields are displayed when you select the select the Show extended set of metadata properties (see page 91) on the Connector Xpress Preferences dialog. (see page 159)

Note: For more information, see Extended Properties.

Attributes Summary Dialog

The Attributes Summary dialog allows you to group mapped attributes logically into groups and subgroups. These attributes appear as tabs and pages in the CA Identity Manager User Console account screens. You can use this dialog to design the account screens that you use to edit the endpoint's accounts in CA Identity Manager. Connector Xpress saves the groupings made on this screen in the metadata. You can then use this metadata in the Role Definition Generator to generate the appropriate screen definitions and import them into CA Identity Manager.

Note: For more information, see How you Generate CA Identity Manager User Console Account Screens (see page 62) in the *Connector Xpress Guide.*

This dialog contains the following fields:

Name

Displays the provisioning attributes that you have mapped.

Maps to

Displays the native name you have mapped your attributes to.

Add

Adds an attribute to the mapping tree for the selected class.

You can use this to define compound type attributes or any other connector-specific attribute that is not mapped to an endpoint attribute.

Remove

Removes the selected attribute from the mapping tree.

Account Screens

(JNDI and JDBC only) Lets you specify how CA Identity Manager groups and displays the attributes you map in the User Console account screens.

The following fields define the way the attributes appear on the tabs and page sections in the account screens in the CA Identity Manager User Console.

Connector Xpress groups common attributes into the following default account screen layouts:

Account

Defines the login, password, and status information for the endpoint account.

Login page section

Defines the login information for the endpoint account.

Password page section

Defines the password information for the endpoint account.

Status page section

Defines the status information for the endpoint account.

User

Defines the personal identification details of the user.

Name page section

Defines the name of the user.

Organization page section

Defines the organization the user belongs to.

Contact

Defines all contact and address information for the user.

Internet page section

Defines internet contact details of the user.

Address page section

Defines the address of the user.

Phone page section

Defines the phone contact details of the user.

Membership

Defines all membership details of the user.

Add

Adds a new tab. This tab appears in CA Identity Manager User Console account screens.

Add Page Section

Adds a new page section to the selected tab. The page section appears on the specified tab in CA Identity Manager User Console account screens.

Add Attribute drop-down

Specifies the attribute you want to add to the selected page section. The attribute appears on the specified page and tab in CA Identity Manager User Console account screens.

Class Associations Dialog

The Class Associations dialog lets you specify the classes you want to create direct and indirect association with.

This dialog contains the following fields:

Create direct association with

Specifies the class you want to create a direct association with.

Create indirect association with

(JDBC only) Specifies the class you want to create an indirect association with.

Associated with

Displays the name of the associated class.

Association attribute

Displays a brief summary of the association.

Remove

Removes the association from the Mapping Tree.

Create New Endpoint Type Dialog

The Create New Endpoint Type dialog lets you define the name of the endpoint type and optionally, the connector server you want to use to manage the endpoint type.

This dialog contains the following fields:

New Endpoint Type

Defines the name of the endpoint type you want to create.

Connector Xpress automatically provides a name for the endpoint type based on the value entered in the Define Endpoint dialog. You can accept the default or enter another name.

Note: If you include any spaces in the name, they are converted to underscores.

Connector Servers

Specifies the connector server you want to manage the new endpoint type.

More Information:

Create an Endpoint Type (see page 84)

Create New Endpoint Dialog

The Create New Endpoint dialog lets you create an endpoint for JDBC and JNDI connectors.

This dialog contains the following fields:

LDAP URL (Required)

(JNDI only) Specifies the LDAP URL.

Base DN (Required)

(JNDI only) Specifies the top level of the DIT (Directory Information Tree) you want to manage within the directory endpoint.

Use TLS (Required)

(JNDI only) Specifies that the CS uses TLS connections between the CS and directory endpoint.

LDAP Version (Required)

Defines the LDAP version.

Note: This field only appears on existing endpoints that have this attribute set to *isRequired=true*. The attribute does not appear for any new projects created with Connector Xpress, as Connector Xpress sets the attribute *isRequired* to false by default in the JNDI metadata.

Endpoint Name (Required)

(JDBC and JNDI) Defines the endpoint name. We recommend that you use a descriptive name for your endpoint.

Password (Required)

(JDBC and JNDI) Specifies the password that is used to log on to the target endpoint.

Bind User DN (Required)

(JNDI only) Specifies the distinguished name of the user used to bind to the directory endpoint.

Connection URI (Required)

(JDBC only) Specifies the secure connection URI.

Username (Required)

(JDBC only) Specifies the username used to log on to the target database.

More Information:

Deploy the Connector (see page 34)
How You Acquire and Manage Endpoints (see page 82)
Acquire, Explore, and Correlate an Endpoint (see page 83)

Create Operation Binding Dialog

This dialog lets you specify operation binding information.

This dialog contains the following fields:

Available object classes

Displays the object classes you can create operation bindings for.

Added object classes

Lets you specify the object classes you want to apply the operation binding to.

Stored Procedure

(JBDC only) Specifies the operation binding is set to JDBC stored procedure type.

Script

Specifies the operation binding is set to scripting style type.

Available Operations

Specifies the available operations that this operation binding can be bound to.

Add

Creates an object on the managed system.

Modify

Modifies the object on the managed system.

Delete

Deletes the object on a managed system.

Modify-Rn

Renames the object on a managed system.

Move

Moves the object from one location to other location on the managed system.

Activate

Activate the connector making it visible to the outside world. This method is called when the connector has been fully initialized, and is ready to receive requests.

Deactivate

Deactivates the connector making it invisible to the outside world. This method is called when the connector is no longer needed and tidies any connections or resources that it holds.

Lookup

Looks up the object on the managed system.

Search

Searches objects on the managed system.

Move-Assocs

Updates association attributes after the target object is moved on the managed system.

Modify-Rn-Assocs

Update association attributes after the target object is renamed on the managed system.

Delete- Assocs

Delete all associations referencing target object on the managed system.

Add-Attr-Assocs

Adds association to the target object on the managed system.

Remove-Attr-Assocs

Move association from the target object on the managed system.

Lookup-Assocs

Lookups all associations for the target object on the managed system.

Search-Assocs

Search all associations for the target object on the managed system.

Modify-Update-Attrs-Assocs

Updates the reverse associative membership list based on changes to the target object on the managed system.

Assoc-Update-References-To

Updates all reverse associative pointers referring to the target object on the managed system.

Assoc-Search-For-References-To

Returns search results (in connector-speak) for each instance of association's class which references the target object, given a target object and an association.

Timing

Specifies when the operation binding is executed.

Before

Species that the connector executes the operation binding before the specified LDAP operation.

After

Species that the connector executes the operation binding after the specified LDAP operation.

Instead Of

Species that the connector executes the operation binding instead of the specified LDAP operation.

Note: If you do not select an object class, this operation binding is applied to all mapped object classes.

More information:

Operation Bindings (see page 54)

Custom Types Dialog

The Custom Types dialog lets you define arbitrary strings (formatted string types) you want to use as metadata types in attribute mappings. The strings you define appear as attribute types you can map in the Type list in the Map Class and Attributes dialog. This dialog also lets you define any static or dynamic enumerations for your attribute.

Some predefined formatted string types are available by default (for example, email, DN, and quoteless), because these formatted strings are already known to the Java Connector Server framework. Handlers for these types are on the connector server.

Note: For more information about creating new formatted string type code, and the steps required for adding support to a custom type, see the *Connector Programming Guide.*

This dialog contains the following fields:

Formatted String Types

Lists the custom strings you can use as metadata types in attribute mappings.

Name

Displays the names of the defined formatted string types.

DN

Specifies a distinguished name string format.

Quoteless

Specifies that quotes are removed from attribute values.

Email

Specifies an e-mail address string format.

Add

Lets you add a new custom string to the Formatted String Types list.

Remove

Removes a custom string type from the Formatted String Types table.

Enumerated Types

Lets you define static enumerated types for use as attribute.

Name

Displays the name of the static enumerated types you defined.

Add

Adds a new enumerated type.

Remove

Removes the selected enumerated type.

Values

Displays the value of the selected enumerated type.

Value

Defines the value of the enumerated type used on the endpoint system.

Display Name

(Optional) Defines the name of the enumerated type displayed in the CA Identity Manager User Console and Provisioning Manager.

Ordinal

(Optional) Defines the order of the enumerated values.

Add

Lets you define a value for a new enumerated type.

Remove

Removes a previously defined value for an enumerated type.

Compound Type Classes

Lets you map the contents of another class to a provisioning attribute.

Name

Displays the names of the compound classes you have added.

Add

Adds a compound class to the mapping tree.

Remove

Removes a compound class from the mapping tree.

Direct Association Dialog

The Direct Association dialog lets you specify a direct association between any two classes of objects. Associations of this type are always directional of the form *from Class1 to Class2.*

You can also use this dialog to establish reverse association that maps an association between the same classes, but in the opposite direction.

Most bi-directional associations have a physical attribute on one class and a virtual attribute on the other class. We recommend that you define the physical association attribute first.

This dialog contains the following fields:

Physical Attribute

Specifies the physical attribute through which an association is made. The attribute is physical in the sense that it exists on the native system.

Virtual Attribute

Specifies an attribute which does not exist on the native system, but rather is virtual and maintained by the Java CS (that is, computed by the Java CS and not persisted on the native system) which forms the part of an association.

Typically in JNDI, memberof is such attribute, as the native systems only store associations in one direction from group to account. A virtual attribute allows you to create an association from account to group instead.

By Attribute

Specifies the association attribute of the target class that the association attribute of the source class references.

Value Template

Specifies a formatted attribute value to match against. For example, in the cases where the target class's association attribute is more complex than the value held in the source class's association attribute.

By default, non-DN association attributes are assumed to hold values that exactly correspond to the values of the name or alternative key of the "to" class. If the association attribute has some internal structure, then you can use the template field to help you deal with the structure when creating an association.

The template value can contain the string ${name} or ${dn}. When creating an association, ${name} is replaced with the simple name of the related object, and ${dn} is replaced by its full DN.

By Filter

Specifies a particular LDAP filter to be used to match the target class's attribute. This is specified as an LDAP search filter, where a string of the form ${attributeName}replaces the value of the corresponding attribute of the *from* class at run time.

Use as a Base Association

Allows you to select an existing association and base the results of the new association on the existing association, but altering those results somehow, for example, adding association results that appear due to nesting.

Objects Must Exist

Specifies that the Connector Server checks that the referenced object exists before it adds it to an association. This check box also affects the reverse association if any. That is, this check box applies to both the source and target class, in either direction.

Use DNs in Attributes

Specifies the values that are persisted on the native system. If selected, flags that the full native DN values are stored. For example, cn=myaccount, ou=myorgunit, rather than just the naming attribute value from a DN, such as myaccount.

Association is Nested

Specifies whether associations between group classes are nested. For example, a single association definition may satisfy "groups of groups of groups" relationships.

Include A Reverse Association

Displays the Reverse Association part of the dialog.

More information:

Direct Associations (see page 43)

Edit Connector Server Configuration Dialog

The Edit Connector Server Configuration dialog lets you set the hosting, routing, and managed connector information for the server.

This dialog contains the following fields:

Connector Server Host Name

Defines the host name of the computer running the connector server.

Port

Defines the TCP port on the connector server used for non-TLS connections.

TLS Port

Defines the TCP port on the connector server used for TLS connections.

Use TLS

Specifies that the connections between the Provisioning Server and the connector server are encrypted TLS connections. If not selected, connections between the Provisioning Server and the connector server are in clear text.

Provisioning server (optional)

Lets you specify the Provisioning Server you want to operate on, as opposed to the Provisioning Server that is selected in the Provisioning Servers tree.

Password

Defines the password used to access the connector server. You usually configure this password when you install the server.

Note: If you edit the Connector Server Host Name, Ports, or Use TLS fields, reenter the password.

Make this the default CS

Specifies that the Provisioning Server uses this connector server as the default.

Note: Any branch (endpoint type or endpoint) that is not explicitly associated with a CS Config is routed through the default CS. As a result, a mismatch may happen if you create an endpoint type or endpoint without explicitly specifying the managing Connector Server. For example, when the default CS is the C++ Connector Server and you have created a dynamic endpoint type without explicitly specifying a Java CS as the managing Connector Server.

Important! You can select this check box for multiple connector servers on a single Provisioning Server. If you select this check box, a random CS is chosen when the request is routed.

Object Handle

Specifies the object handles of the endpoint types the selected Provisioning Server is managing.

Note: We recommend that you manage this list using the Set Managing Connector Server procedure.

More Information:

How you Set a Managing Connector Server (see page 76)
Edit a Connector Server Configuration (see page 77)
Create a Connector Server Configuration (see page 75)

Edit Metadata for Endpoint Type Dialog

The Edit Metadata for Endpoint Type dialog lets you manually edit metadata.

This dialog contains the following fields:

Node Type

Displays a list of possible node types that you can add using the Add button. The contents of the list depend on the type of node currently selected in the XML tree.

Add

Lets you add a new node to the XML tree of the type specified in the Node Type drop-down list.

Edit

Lets you edit the currently selected node in the XML tree.

Remove

Lets you remove the currently selected node from the XML tree.

XML Tree

Displays an XML tree view of the metadata that you can edit.

Export

Lets you export a data model to an XML file.

More Information:

Edit Metadata (see page 87)

Edit Project Settings Dialog

The edit project settings dialog lets you edit the settings for a project.

This dialog contains the following fields:

Project File Comment

Defines free-form comments for your project.

Metadata Dictionaries

Specifies the dictionaries you can use to load a set of predefined custom operational attributes.

Edit Script Dialog

This dialog lets you specify the parameters for scripts.

This dialog contains the following fields:

Script Language

Specifies the available scripting languages that Java Connector Server framework currently supports.

More information:

Scripts (see page 55)
Bind Operations to Scripts (see page 57)

Edit Source Dialog

The Edit Source dialog lets you specify the name and connection details for your data source. For JDBC data sources, this dialog lets you construct an appropriate JDBC URL for the database you select. For JNDI data sources, the dialog lets you specify the details of an LDAP connection.

Note: The fields in this dialog vary, depending on the type of data source selected.

This dialog contains the following fields:

Name

Specifies the name of the data source.

Note: We recommend that you chose a name that clearly describes what the data source is for.

Server Name

Specifies the host name of the server you want to access.

Host Name

(Informix Dynamic Server) Specifies the name of the Informix server.

Database Type

(JDBC only) Specifies the type of database connection you want to make:

- DB2
- DB2 for IBMi
- Informix Dynamic Server
- INGRES
- Microsoft SQL Server
- MySQL
- Oracle
- Sybase Adaptive Server Enterprise
- Other

Username

(JDBC only) Specifies the username the connector uses to log on to the target database endpoint.

Defaults:

Oracle: System

Microsoft SQL Server: sa

DB2: db2admin

Ingres: Ingres

Native

(JDBC and MS SQL Server only) Specifies that MSSQL Native Authentication on Windows is activated. The following is added to the JDBC URL used for the connection:

integratedSecurity=true

Server name

(JDBC only) Specifies the hostname of the database server.

SID

(JDBC and Oracle only) Specifies the Oracle SID.

Port

Specifies the port number the connector uses for communication with the endpoint.

Defaults

JNDI: 389

Oracle: 1521

Microsoft SQL Server: 1433

DB2: 6789

INGRES: II7

Informix Dynamic Server: 9088

MySQL: 3306

Database

(JDBC only) Defines the database name (most vendors including MS SQL) or its SID (Oracle).

Defaults

Microsoft SQL Server: master

Oracle SID: ORCL

JDBC URL

(JDBC only) Defines the JDBC URL. Connector Xpress automatically forms the URL as you complete the previous fields.

Note: To edit this field, select the Edit check box.

Bind DN

(JNDI only) Defines the distinguished name of the user used to bind to the directory endpoint.

Anonymous

(JNDI only) Specifies that no bind DN is used and instead, an anonymous bind to the directory endpoint is used.

Use TLS

Specifies that Connector Xpress uses TLS connections between Connector Xpress and the endpoint.

Base DN

(JNDI only) Defines the top level of the DIT (Directory Information Tree) you want to manage within the directory endpoint.

Edit

(JDBC only) Lets you edit the JDBC URL field.

Test

Attempts to connect to the specified directory or database using the information provided.

More Information:

Create a Project (see page 26)
Set Up Data Sources (see page 14)

Endpoint Class Dialog

The Endpoint Class dialog lets you create new attributes to map, rather than selecting discovered attributes in the attribute mapping table. You can also load a set of predefined attributes from a dictionary into the attribute mapping table.

This dialog contains the following fields:

Name

Defines the name of the class you are mapping.

Limits: Must begin with a letter.

Description

Describes the class you are mapping.

Connector Map To

Specifies which name to map an object class (including the connector itself) or attribute to in connector-speak. For a dynamic connector, this attribute specifies the name of the native system item to map the attribute to. For example, a JDBC-based connector would probably map an object class to the name of a database table, and each property within it to the name of one of its columns.

Managed

If unchecked, marks this class as mapped only for the purpose of establishing associations. As a result, Connector Xpress only maps its name and type. Instances of an unmanaged class can be listed and associated with other objects, but cannot be created, edited or deleted.

For compound classes, Connector Xpress selects this field by default, and cannot be cleared.

Add Dictionary Attribute

Loads a set of predefined custom operational attributes from a dictionary and displays the attributes in the mapping tree.

Note: This field is available when you specify that you want to load operational attributes from a dictionary. (see page 92)

Extended Properties

Displays an extended set of metadata properties. These fields are displayed when you select the select the Show extended set of metadata properties (see page 91) on the Connector Xpress Preferences dialog. (see page 159)

Note: For more information, see Extended Properties.

Enter Password for Data Source Dialog

Use the Enter Password for Data Source dialog to specify the password for the data source used for the basis of the connection.

This dialog contains the following fields:

Source name

Displays the name of the data source specified when Connector Xpress creates the data source.

Source URL

Displays the connection URL of the data source.

Source type

Displays the type of the data source (for example, JNDI, JDBC).

User name

Displays the name of the user used to connect to the data source.

Password

Specifies the password of the user Connector Xpress uses to connect to the data source.

More Information:

Edit a Project (see page 30)
Create a Project (see page 26)

Endpoint Type Details Dialog

The Endpoint Type Details dialog lets you specify the name, description, and version number of the endpoint type you want to define.

This dialog contains the following fields:

Name

(Mandatory) Specifies the name of the endpoint type.

Note: The name cannot contain the characters # / \ [ ] ; : | = . , + * ? < > @ ( ) and any spaces are converted into underscores.

If you enter any of these characters, Connector Xpress displays a warning. We recommend that you remove these characters from your existing metadata.

Description

Specifies the extended description of the endpoint type.

Version

Specifies the version number for the endpoint type. Specifying a version can be useful when modifying a Connector Xpress project file to record incremental changes. Specifying a version does not have a logical effect in the dynamic connector.

Extended Properties

Displays an extended set of metadata properties. These fields are displayed when you select the select the Show extended set of metadata properties (see page 91) on the Connector Xpress Preferences dialog. (see page 159)

Note: For more information, see Extended Properties.

Explore/Correlate Endpoint Dialog

The Explore/Correlate Endpoint dialog lets you specify how you want to explore and correlate the endpoint.

This dialog contains the following fields:

Explore Endpoint first endpoint

Specifies whether this endpoint is explored to find undiscovered accounts and groups.

Sub Tree

Specifies that all the containers in the subtree are included in the endpoint exploration.

One Level

Specifies that only one level of containers are included in the endpoint exploration.

Correlate Accounts with Users

Specifies whether global user accounts are correlated with information found in accounts and groups on the endpoint system.

Use existing Users

Specifies that the correlation links each account to the user with the same name. The correlation links only existing users; none are created.

Create Users as needed

Specifies that the correlation links each account to the user sharing the same name. If the user does not exist, the correlation process creates the user.

More Information:

Deploy the Connector (see page 34)
Acquire, Explore, and Correlate an Endpoint (see page 83)
How You Acquire and Manage Endpoints (see page 82)

Extended Metadata Properties

The following fields are displayed on Connector Xpress dialogs when you select the select the Show extended set of metadata properties (see page 91) on the Connector Xpress Preferences dialog (see page 159).

Connector Xpress displays the fields on the following dialogs:

- Endpoint Type Details dialog (see page 129)
- Map Class and Attributes dialog (JDBC) (see page 136)
- Map Class and Attributes Dialog (JNDI) (see page 140)
- Map Compound Class and Attributes dialog (see page 144)
- Attribute Details dialog
- Endpoint Class dialog (see page 128)
- Map Container Class dialog (see page 148)

Note: Not all fields are displayed on all dialogs.

Assoc Obj Key Attr

Targets any indirect association property. Alternate key used to name the parent class of the target property that is, not the referenced class.

Assoc Ref Object Class

Defines a relationship from an obj type to a referenced obj type, where a direct association through an attribute and an indirect association through a table external to both object types are supported.

Assoc Reverse Attr

Targets any direct association property, names attribute in the referenced class (that is, not the parent of target property) which contains association information in the reverse direction.

Assoc Table

Targets any indirect association property, names table which stores association links.

Assoc Table Obj Column

Targets any indirect association property, specifies column in table named in ${MD_ASSOC_TABLE} which contains references to class which is parent of target property.

Assoc Table Ref Column

Targets any indirect association property, specifies column in table named in ${MD_ASSOC_TABLE} which contains references to referenced class (that is, not the parent class of target property).

Assoc Type

When true specifies that target attribute specifies an association between a parent object and its compound value child objects.

Connector Generated Override

Specifies how to handle attempts to override generated attribute values in LDAP ADD requests.

Connector Generator

Generator name (or literal expression passed to the endpoint if surrounded in \"s) which specifies the value assigned to the property

Example: \'my_sequence\' or \'\"NEXT VALUE FOR my_sequence\"\' for JDBC.

Connector Map To

Specifies which name to map an object class (including the connector itself) or attribute to in connector-speak. For a dynamic connector, this attribute specifies the name of the native system item to map the attribute to. For example, a JDBC-based connector would probably map an object class to the name of a database table, and each property within it to the name of one of its columns.

Connector Map To Alias

Specifies which name to map an objectclass (including the connector itself) or attribute to in connector-speak. For example, where MD_CONN_MAP_TO is mapped to a complicated expression and therefore does not act as a useful reference name in the connector's code.

Connector Map To Ambiguous

Specifies the rare cases where a single LDAP attribute ambiguously maps to more the one connector objectclass or attribute. The order of the choices is important, as the first choice is the default when no explicit choice is made and a new object is created.

Connector Map To Ambiguous Choice Attr

Specifies that an attribute with ambiguous mappings can optionally name another LDAP attribute which is used to pass in an explicit choice when new instances of the objectclass are add()ed

Connector Map To Auxiliary

Identifies all auxiliary classes by name to which mappings are present.

Connector Map To Derived

Identifies all structural classes by name to which mappings are present with any classes that each of them derives from.

Connector Map To Lax

Specifies that MD_CONN_MAP_TO mappings are *lax*. The Boolean can appear on any objectclass which has MD_CONN_MAP_TO specified.

Connector Map to Multiple

Specifies multiple connector-speak classes or attributes. Used when you need to split a single LDAP class or attribute into multiple connector-speak attributes, or compose it from the values of multiple connector-speak attributes.

Connector Map To Same

Specifies that all properties defined for this attribute will use their LDAP names literally as their connector-speak names. This attribute appears on an object class which does not have MD_CONN_MAP_TO specified. Object classes without either MD_CONN_MAP_TO or this attribute are skipped.

Connector Map To Strict

Causes the framework to augment objectclass filters to include an explicit check for the existence of a particular connector-speak naming attribute. This is used in cases where the naming attribute is potentially ambiguous, but you have chosen to map only one choice.

Connector XML

Contains dynamic version of connector.xml configuration used for all connectors on this namespace.

Connector Map to Derived

Identifies all structural classes by name to which mappings are present with any classes that each of them derives from.

Default Value

(Optional) Specifies the default value for the attribute.

Note: Only clients use this value (for example, user interfaces). If the client does not supply a value, the Provisioning Server does not use it by default.

Implementation Bundle

Defines the name of connector implementation targeted by the connector type.

Is Compound Value

If selected, specifies that that the target class exists to define the format of values accepted by a compound or structured attribute.

Is Connection

Distinguishes connection-related attributes on the connector.

Is Connector Generated

Specifies that the value for property is generated implicitly by the endpoint (For example, true for IDENTITY column in JDBC).

Is Hidden

Specifies that the value for the property is not displayed in GUIs.

Is Naming

Specifies whether the attribute is the LDAP naming attribute.

Is Operational Attribute

Specifies whether the target property is operational (that is, calculated by the native endpoint rather then persisted).

Is Read Only Native

Specifies whether the property is read-only on the endpoint system.

isTransactionsEnabled

(DB2 IBM i only) Controls whether transactions are used when performing operations (for example, Add, Delete, Modify) on a JDBC endpoint. Applicable where transactions cannot be used for DB2 for IBM i database tables that are not journaled.

Is Well Known

Specifies whether this property is recognized as a member of the common set.

Metadata Item Name

(Read only) Defines the value for the name of the object within the metadata XML. For classes and attributes, this is usually the LDAP provisioning name.

Virtual

Flags the target property as *computed* rather than *persisted*.

Indirect Association Dialog (JDBC only)

You can use the Indirect Association dialog to specify a two-way association between any two classes of objects. The association between the two objects is bi-directional and contained in a third entity, that is a table, which holds the association links between the objects.

Each entry indicates an association between instances of each object. The node representing the association appears under the Associations node in the mapping tree.

Due to the symmetrical nature of containing the association in a third entity, lookups in either direction are equally fast and reverse associations are automatically assumed and created.

This dialog contains the following fields:

Schema

Specifies the schema (for example, tablespace or database) name for which the tables are displayed in the Membership Table field.

Membership Table

Specifies the tables that define the members of each group.

Note: The database table that you select requires at least two columns and cannot have more than two required columns.

***Source Class* Attribute**

Specifies the attribute of the source class whose value is held in the membership table.

Membership Table Columns

Specifies the membership table columns that refer to the source and target class attributes.

***Target Class* Attribute**

Specifies the attribute of the target class whose value is held in the membership table.

***Source Class* Attribute**

Specifies the name of the virtual attributes on the source class that contain the list of names of associated objects.

***Target Class* Attribute**

Specifies the name of the virtual attributes on the target object that contain the list of names of associated objects.

Objects Must Exist

Specifies whether the associated entries must exist.

Default: Selected.

Use DNs in Attributes

Specifies whether the virtual membership attributes contain a list of DNs, or just a list of the target object's naming attributes.

Default: Unselected.

More information:

Indirect Associations (see page 45)

Mapped Classes Dialog

The Mapped Classes dialog displays a read-only list of any classes that you have mapped and the native classes you have mapped them to.

This dialog contains the following fields:

Name

Displays classes that you have mapped.

Maps to

Displays the native classes you have mapped your classes to.

Add

Lets you add a new class node to the Mapping Tree.

Remove

Removes a mapped Class from the Mapping Tree.

Map Class and Attributes Dialog (JDBC)

The Map Class and Attributes dialog (JDBC) lets you specify the database tables that you want map this class to. Every class mapping that you create must be associated with exactly one database table.

This dialog contains the following fields:

Name

Defines the name of the class you are mapping.

Limits: Must begin with a letter.

Description

Describes the class you are mapping.

Managed

If unchecked, marks this class as mapped only for the purpose of establishing associations. As a result, Connector Xpress only maps its name and type. Instances of an unmanaged class can be listed and associated with other objects, but cannot be created, edited or deleted.

For compound classes, Connector Xpress selects this field by default, and cannot be cleared.

Schema (Mandatory)

Specifies the schema (for example, tablespace or database) name for the tables displayed in the Table list.

Table (Mandatory)

Specifies the table that represents account/group information in the selected schema.

You must associate every class mapping that you create with exactly one database table.

Map Columns

Displays an overview of the column mappings you have selected.

Note: This table does not show the full set of mapping options. For more rarely used options, expand the class node to display individual attribute detail nodes.

Native Name

Displays the name of the native attribute.

Bold entries indicate mandatory entries that you must map at least once per class.

The naming attribute of the class in question is displayed in bold.

Native Type

Specifies the data-model type depending on the SQL type for the JDBC case.

Native Size

Displays the size of the Account/Group Table column in characters.

Note: In some cases this value is a "best guess", such as for NVARCHARS or storage of time-related values in vendor-specific formats.

Name

Lists the provisioning attributes you can map to the native attribute.

Italic entries

Indicate that the attribute has already been mapped. For JNDI, these attributes can be mapped again.

Bold entries

Indicate mandatory entries that you must map at least once per class.

Custom attributes

Indicate that the attribute is given a default name based on its native attribute name. You can modify this attribute in the Provisioning Attribute Details dialog.

Blank entry

Lets you remove a mapping.

Note: For account classes, this list also displays the list of well-known attributes.

Type

Specifies the data-model type depending on the SQL type for the JDBC case.

Binary Data

Defines an attribute whose value is arbitrary binary data.

Boolean

Specifies logically true or false in XML, but represented by the Provisioning Server and JIAM APIs as 1 or 0 in LDAP attribute values.

Date

Specifies a date.

Example: 1999-05-31

Note: The Dynamic Namespace plug-in to Provisioning Manager supports the years from 1800 through 9999, but other components of the solution impose no such restrictions and can represent virtually any year in recorded history.

Date & Time

Specifies a particular time on a particular day.

Example: 1999-05-31T13:20:00

Note: The Dynamic Namespace plug-in to Provisioning Manager supports the years from 1970 through 2036, so you must use Date to represent days falling outside of this range.

Note: Vendor differences complicate how Connector Xpress handles time-related columns. For example, MSSQL "DATETIME" signifies a DateTime value whereas other vendors use the standard "TIMESTAMP", and MSSQL TIMESTAMPs are automatically generated binary values. Also, Oracle does not support a "TIME" type and its "DATE" type is also effectively a TIMESTAMP. Therefore, to remain vendor-neutral, Connector Xpress allows you to map to any of Date/DateTime/Time whenever it makes sense for you to do so.

Double-precision floating-point

Specifies a double-precision 64-bit floating-point value.

Enumeration - enumeration type name

Specifies an attribute with a fixed list of enumerated values.

Flexi-DN

Specifies a distinguished name string format.

For example, "cn=Bob,ou=Sales,o=ExampleCorp". The connector enforces this.

Flexi-Email

Specifies an email address string format.

Flexi-Quoteless

Specifies that quotes are removed from attribute values.

Floating Point

Specifies a single-precision 32-bit floating-point number.

Integer

Specifies a 32-bit value between -2147483648 and 2147483647.

Long Integer

Specifies a 64-bit value from 9223372036854775808 through 9223372036854775807.

String

Specifies an unrestricted field.

Time

Specifies an offset of between 0 seconds and 23:59:59.

Example: 13:20:00

Multi-valued

If selected, specifies that this attribute is multi-valued.

Note: If the native attribute is multi-valued, Connector Xpress automatically selects this check box. If the native attribute is single-valued, this option is cleared and read-only.

Extended Properties

Displays an extended set of metadata properties. These fields are displayed when you select the select the Show extended set of metadata properties (see page 91) on the Connector Xpress Preferences dialog. (see page 159)

Note: For more information, see Extended Properties.

Map Class and Attributes Dialog (JNDI)

The Map Class and Attributes dialog lets you specify the native object classes (JNDI) that you want to map a class too. You can map any attributes of this structural class, and associate zero or more auxiliary LDAP classes with the class mapping.

If you map to a specific provisioning attribute as a part of structural class mapping table, then Connector Xpress does not allow you to map to this attribute in any of the auxiliary class tables. This limitation is due to the nature of the auxiliary classes in general. The reverse is also true. For example, if you are mapping a well-known provisioning attribute, for example, eTSuspended to an auxiliary table, it is no longer present in the drop-down lists for other classes.

For JNDI-based endpoint types, you can map the account class to multiple endpoint object classes. When the user creates an account for this endpoint type through the Identity Manager User Console, they select which of the object classes they would like to use for the account from a list. This list is made possible by a choice attribute that is generated automatically by Connector Xpress whenever the account class is mapped to multiple structural classes. This attribute should be added to your user console account screens like any other attribute.

Likewise, any provisioning attribute that is mapped to multiple endpoint attributes will have a choice attribute generated for it. For ease of use, these should be placed on your user console account screens near the attribute in question.

This dialog contains the following fields:

Name

Defines the name of the class you are mapping.

Limits: Must begin with a letter.

Description

Describes the class you are mapping.

Managed

If unchecked, marks this class as mapped only for the purpose of establishing associations. As a result, Connector Xpress only maps its name and type. Instances of an unmanaged class can be listed and associated with other objects, but cannot be created, edited or deleted.

For compound classes, Connector Xpress selects this field by default, and cannot be cleared.

Search Container

Defines a particular location in the DIT where all objects of this class can be found. This can help the performance of certain search operations performed by the Connector Server.

Add structural class

Specifies all native LDAP object classes and displays the attributes in the attributes table.

You must associate every class mapping that you create with at least one structural class.

Add auxiliary class

Specifies any existing auxiliary classes of the structural class you selected and displays the attributes from other auxiliary LDAP classes in the attributes table.

Class Name

Displays the selected object classes for this provisioning class mapping.

Type

Displays whether the native object class is structural or auxiliary.

Derived From

Displays the inheritance hierarchy of this native object class.

Remove

Removes the selected object class.

Map Object Class Attributes

Displays an overview of the attribute mappings you have selected.

Note: This table does not show the full set of mapping options. For rarely used options, expand the class node to display individual attribute detail nodes.

Native Name

Displays the name of the native attribute.

Bold entries indicate mandatory entries that you must map at least once per class.

The naming attribute of the class in question is displayed in bold.

Native Type

Default: String for JNDI. You can change the default if necessary.

Name

Lists the provisioning attributes you can map to the native attribute.

Italic entries

Indicate that the attribute has already been mapped. For JNDI, these attributes can be mapped again.

Bold entries

Indicate mandatory entries that you must map at least once per class.

Custom attributes

Indicate that the attribute is given a default name based on its native attribute name. You can modify this attribute in the Provisioning Attribute Details dialog.

Blank entry

Lets you remove a mapping.

Note: For account classes, this list also displays the list of well-known attributes.

Type

Default: String for JNDI. You can change the default if necessary.

Important! The data model type is important because it drives data validation and conversion in clients and the Java Connector Server. The following types match syntaxes defined as part of the XML schema (XSD) specification. You can find information about the XML schema definition at the following website:

http://www.w3.org

Binary Data

Defines an attribute whose value is arbitrary binary data.

Boolean

Specifies logically true or false in XML, but represented by the Provisioning Server and JIAM APIs as 1 or 0 in LDAP attribute values.

Date

Specifies a date.

Example: 1999-05-31

Note: The Dynamic Namespace plug-in to Provisioning Manager supports the years from 1800 through 9999, but other components of the solution impose no such restrictions and can represent virtually any year in recorded history.

Date & Time

Specifies a particular time on a particular day.

Example: 1999-05-31T13:20:00

Note: The Dynamic Namespace plug-in to Provisioning Manager supports the years from 1970 through 2036, so you must use Date to represent days falling outside of this range.

Note: Vendor differences complicate how Connector Xpress handles time-related columns. For example, MSSQL "DATETIME" signifies a DateTime value whereas other vendors use the standard "TIMESTAMP", and MSSQL TIMESTAMPs are automatically generated binary values. Also, Oracle does not support a "TIME" type and its "DATE" type is also effectively a TIMESTAMP. Therefore, to remain vendor-neutral, Connector Xpress allows you to map to any of Date/DateTime/Time whenever it makes sense for you to do so.

Double-precision floating-point

Specifies a double-precision 64-bit floating-point value.

Enumeration - enumeration type name

Specifies an attribute with a fixed list of enumerated values.

Flexi-DN

Specifies a distinguished name string format.

For example, "cn=Bob,ou=Sales,o=ExampleCorp". The connector enforces this.

Flexi-Email

Specifies an email address string format.

Flexi-Quoteless

Specifies that quotes are removed from attribute values.

Floating Point

Specifies a single-precision 32-bit floating-point number.

Integer

Specifies a 32-bit value between -2147483648 and 2147483647.

Long Integer

Specifies a 64-bit value from 9223372036854775808 through 9223372036854775807.

String

Specifies an unrestricted field.

Time

Specifies an offset of between 0 seconds and 23:59:59.

Example: 13:20:00

Multi-valued

If selected, specifies that this attribute is multi-valued.

Note: If the native attribute is multi-valued, Connector Xpress automatically selects this check box. If the native attribute is single-valued, this option is cleared and read-only.

Extended Properties

Displays an extended set of metadata properties. These fields are displayed when you select the select the Show extended set of metadata properties (see page 91) on the Connector Xpress Preferences dialog. (see page 159)

Note: For more information, see Extended Properties.

Map Compound Class and Attributes Dialog (JDBC)

The Map Compound Class and Attributes dialog (JDBC) lets you create a Compound type class that you can use to you provide multi-table support for JDBC connectors.

Every class mapping that you create must be associated with exactly one database table.

This dialog contains the following fields:

Name

Defines the name of the class you are mapping.

Limits: Must begin with a letter.

Description

Describes the class you are mapping.

Managed

If unchecked, marks this class as mapped only for the purpose of establishing associations. As a result, Connector Xpress only maps its name and type. Instances of an unmanaged class can be listed and associated with other objects, but cannot be created, edited or deleted.

For compound classes, Connector Xpress selects this field by default, and cannot be cleared.

Schema (Mandatory)

Specifies the schema (for example, tablespace or database) name for the tables displayed in the Table list.

Table (Mandatory)

Specifies the table that represents account/group information in the selected schema.

You must associate every class mapping that you create with exactly one database table.

Map Columns

Displays an overview of the column mappings you have selected.

Note: This table does not show the full set of mapping options. For more rarely used options, expand the class node to display individual attribute detail nodes.

Native Name

Displays the name of the native attribute.

Bold entries indicate mandatory entries that you must map at least once per class.

The naming attribute of the class in question is displayed in bold.

Native Type

Specifies the data-model type depending on the SQL type for the JDBC case.

Native Size

Displays the size of the Account/Group Table column in characters.

Note: In some cases this value is a "best guess", such as for NVARCHARS or storage of time-related values in vendor-specific formats.

Name

Lists the provisioning attributes you can map to the native attribute.

Italic entries

Indicate that the attribute has already been mapped. For JNDI, these attributes can be mapped again.

Bold entries

Indicate mandatory entries that you must map at least once per class.

Custom attributes

Indicate that the attribute is given a default name based on its native attribute name. You can modify this attribute in the Provisioning Attribute Details dialog.

Blank entry

Lets you remove a mapping.

Note: For account classes, this list also displays the list of well-known attributes.

Type

Specifies the data-model type depending on the SQL type for the JDBC case.

Binary Data

Defines an attribute whose value is arbitrary binary data.

Boolean

Specifies logically true or false in XML, but represented by the Provisioning Server and JIAM APIs as 1 or 0 in LDAP attribute values.

Date

Specifies a date.

Example: 1999-05-31

Note: The Dynamic Namespace plug-in to Provisioning Manager supports the years from 1800 through 9999, but other components of the solution impose no such restrictions and can represent virtually any year in recorded history.

Date & Time

Specifies a particular time on a particular day.

Example: 1999-05-31T13:20:00

Note: The Dynamic Namespace plug-in to Provisioning Manager supports the years from 1970 through 2036, so you must use Date to represent days falling outside of this range.

Note: Vendor differences complicate how Connector Xpress handles time-related columns. For example, MSSQL "DATETIME" signifies a DateTime value whereas other vendors use the standard "TIMESTAMP", and MSSQL TIMESTAMPs are automatically generated binary values. Also, Oracle does not support a "TIME" type and its "DATE" type is also effectively a TIMESTAMP. Therefore, to remain vendor-neutral, Connector Xpress allows you to map to any of Date/DateTime/Time whenever it makes sense for you to do so.

Double-precision floating-point

Specifies a double-precision 64-bit floating-point value.

Enumeration - enumeration type name

Specifies an attribute with a fixed list of enumerated values.

Flexi-DN

Specifies a distinguished name string format.

For example, "cn=Bob,ou=Sales,o=ExampleCorp". The connector enforces this.

Flexi-Email

Specifies an email address string format.

Flexi-Quoteless

Specifies that quotes are removed from attribute values.

Floating Point

Specifies a single-precision 32-bit floating-point number.

Integer

Specifies a 32-bit value between -2147483648 and 2147483647.

Long Integer

Specifies a 64-bit value from 9223372036854775808 through 9223372036854775807.

String

Specifies an unrestricted field.

Time

Specifies an offset of between 0 seconds and 23:59:59.

Example: 13:20:00

Multi-valued

If selected, specifies that this attribute is multi-valued.

Note: If the native attribute is multi-valued, Connector Xpress automatically selects this check box. If the native attribute is single-valued, this option is cleared and read-only.

Extended Properties

Displays an extended set of metadata properties. These fields are displayed when you select the select the Show extended set of metadata properties (see page 91) on the Connector Xpress Preferences dialog. (see page 159)

Note: For more information, see Extended Properties.

More information:

Compound Class Mapping (see page 36)

Mapped Containers Dialog

This dialog lets you specify the object classes which you want to use as containers in your connector.

This dialog contains the following fields that are not self-explanatory.

Name

Specifies the name of the container.

Contains

Displays the list of all defined containers.

Add

Adds a new container class to the mapping tree.

Remove

Removes the container class definition.

Map Container Class Dialog (JNDI)

The Map Class and Attributes Containers dialog lets you specify the objectclasses that can act as containers for accounts and groups, and their naming attributes.

By default Connector Xpress populates the Object Class list with the following entries which are the containers for inetOrgPerson and groupOfNames in the inetOrg schema, which are common account and group choices:

- Organization – o
- Organizational Unit – ou

Note: The attributes displayed for inetOrgPerson account objects do not have a direct relation to the corresponding container. Although ideally they would match, however there is no guarantee that match do for any given account object.

This dialog contains the following fields:

Name

Defines the name of the class you are mapping.

Limits: Must begin with a letter.

Description

Describes the class you are mapping.

Managed

If unchecked, marks this class as mapped only for the purpose of establishing associations. As a result, Connector Xpress only maps its name and type. Instances of an unmanaged class can be listed and associated with other objects, but cannot be created, edited or deleted.

For compound classes, Connector Xpress selects this field by default, and cannot be cleared.

Contained Classes

Specifies the classes that can be children of this container. For example, you can specify that the container Employee Groups can only allow Staff Group and Executive Group classes and not an individual Account Class.

Add structural class

Specifies all native LDAP object classes and displays the attributes in the attributes table.

You must associate every class mapping that you create with at least one structural class.

Add auxiliary class

Specifies any existing auxiliary classes of the structural class you selected and displays the attributes from other auxiliary LDAP classes in the attributes table.

Class Name

Displays the selected object classes for this provisioning class mapping.

Type

Displays whether the native object class is structural or auxiliary.

Derived From

Displays the inheritance hierarchy of this native object class.

Remove

Removes the selected object class.

Map Object Class Attributes

Displays an overview of the attribute mappings you have selected.

Note: This table does not show the full set of mapping options. For rarely used options, expand the class node to display individual attribute detail nodes.

Native Name

Displays the name of the native attribute.

Bold entries indicate mandatory entries that you must map at least once per class.

The naming attribute of the class in question is displayed in bold.

Native Type

Default: String for JNDI. You can change the default if necessary.

Name

Lists the provisioning attributes you can map to the native attribute.

Italic entries

Indicate that the attribute has already been mapped. For JNDI, these attributes can be mapped again.

Bold entries

Indicate mandatory entries that you must map at least once per class.

Custom attributes

Indicate that the attribute is given a default name based on its native attribute name. You can modify this attribute in the Provisioning Attribute Details dialog.

Blank entry

Lets you remove a mapping.

Note: For account classes, this list also displays the list of well-known attributes.

Type

Default: String for JNDI. You can change the default if necessary.

Important! The data model type is important because it drives data validation and conversion in clients and the Java Connector Server. The following types match syntaxes defined as part of the XML schema (XSD) specification. You can find information about the XML schema definition at the following website:

http://www.w3.org

Binary Data

Defines an attribute whose value is arbitrary binary data.

Boolean

Specifies logically true or false in XML, but represented by the Provisioning Server and JIAM APIs as 1 or 0 in LDAP attribute values.

Date

Specifies a date.

Example: 1999-05-31

Note: The Dynamic Namespace plug-in to Provisioning Manager supports the years from 1800 through 9999, but other components of the solution impose no such restrictions and can represent virtually any year in recorded history.

Date & Time

Specifies a particular time on a particular day.

Example: 1999-05-31T13:20:00

Note: The Dynamic Namespace plug-in to Provisioning Manager supports the years from 1970 through 2036, so you must use Date to represent days falling outside of this range.

Note: Vendor differences complicate how Connector Xpress handles time-related columns. For example, MSSQL "DATETIME" signifies a DateTime value whereas other vendors use the standard "TIMESTAMP", and MSSQL TIMESTAMPs are automatically generated binary values. Also, Oracle does not support a "TIME" type and its "DATE" type is also effectively a TIMESTAMP. Therefore, to remain vendor-neutral, Connector Xpress allows you to map to any of Date/DateTime/Time whenever it makes sense for you to do so.

Double-precision floating-point

Specifies a double-precision 64-bit floating-point value.

Enumeration - enumeration type name

Specifies an attribute with a fixed list of enumerated values.

Flexi-DN

Specifies a distinguished name string format.

For example, "cn=Bob,ou=Sales,o=ExampleCorp". The connector enforces this.

Flexi-Email

Specifies an email address string format.

Flexi-Quoteless

Specifies that quotes are removed from attribute values.

Floating Point

Specifies a single-precision 32-bit floating-point number.

Integer

Specifies a 32-bit value between -2147483648 and 2147483647.

Long Integer

Specifies a 64-bit value from 9223372036854775808 through 9223372036854775807.

String

Specifies an unrestricted field.

Time

Specifies an offset of between 0 seconds and 23:59:59.

Example: 13:20:00

Multi-valued

If selected, specifies that this attribute is multi-valued.

Note: If the native attribute is multi-valued, Connector Xpress automatically selects this check box. If the native attribute is single-valued, this option is cleared and read-only.

Extended Properties

Displays an extended set of metadata properties. These fields are displayed when you select the select the Show extended set of metadata properties (see page 91) on the Connector Xpress Preferences dialog. (see page 159)

Note: For more information, see Extended Properties.

Merge XML Dialog

The Merge XML dialog lets you import an XML file that contains metadata into your project. Each project automatically contains metadata but you may want to import an external metadata file if, for example, you have one from another project that you want to reuse. When you import an XML metadata file, the system merges the new file with the existing one already in the project, but you must specify how the system handles any conflicts. This dialog lets you specify how the system deals with the conflict.

This dialog contains the following fields:

Retain existing project values

Maintain the existing metadata values in the project.

Overwrite existing project values with those in the XML file being selected

Replace the existing metadata in the project with the imported metadata.

More information:

Merge a Modified Data Model (see page 81)

Operations Bindings Editor

The Operations Bindings dialog displays a summary of existing operation bindings.

This dialog contains the following fields:

Available object classes

Displays the object classes you can create opbindings for.

Filtered object classes

Lets you filter the operation bindings displayed in the Opbindings summary list. Operation bindings for the classes displayed in this list are displayed in the Opbindings summary list.

The selections you make in this list are also displayed in all Filtered object classes lists in all Opbinding Operation editors. Also any changes to you make to this list in other Operation binding editors are also displayed in this list.

Opbindings summary

Displays a summary of existing operation bindings.

Create

Displays the Create Operation Binding dialog which lets you specify operation binding information.

Delete

Deletes the operation binding from the Opbindings summary list.

More information:

Operation Bindings (see page 54)
Stored Procedures (see page 55)
Bind Operations to Stored Procedures (see page 56)
Stored Procedure and Column Considerations (see page 59)

Operation Bindings – Stored Procedure Editor

This dialog lets you specify the parameters for a stored procedure style operation binding.

This dialog contains the following fields:

Available object classes

Displays the available mapped object classes in the following formats:

- **Object class format**

  Format: DisplayName [connectorMapToObjectClassName]

  Example: User Account [inetOrgPerson]

- **Ambiguous object classes format –** Displayed as a single item in the following format:

  Format: DisplayName [connectorMapToObjectClassName1 | connectorMapToObjectClassName2 | …]

  Example: Default Container [organizationalUnit | organization]

Note: If you choose any ambiguous object class for the concurrent Opbinding, the LDAP object class name is always used in the Opbinding xml content. As a result, Connector Xpress clears the Use Native Name check box and makes it unavailable.

Added object classes

Lets you specify the object classes you want to add to this script operation binding.

Execute this procedure

Specifies when you want the connector to execute the stored procedure.

Description

Describes the operation binding.

Abort operation if procedure fails

Specifies how errors are handled. For example, if the timing is set to Before or After, and this check box is selected, exceptions are thrown and the operation aborts if any errors occur. If this check box is cleared the connector logs the error but the operation continues if an error occurs.

If the timing is set to Instead Of, and an error occurs, the connector always throws exceptions and aborts the operation when executing the operation binding.

When the timing is set to Instead Of this field is set to true by default, and cleared by default when the timing is set to After.

Use native name

Specifies that the connector uses connector-map-to attribute names for the binding rather than DYN LDAP attribute names.

Lookup Level

Specifies whether the attributes or the deleted object are cached for the opbinding to function and controls the lookup of all an object's values. Caching is typically done preemptively for Post delete operations.

The default for scripts is Full. For methods, the framework verifies all the parameters mapped for the Post delete stored procedure and selects Full or Exists depending whether non-contextual attributes are present or not, respectively.

Note: The LookUpLevel attribute is important to the handling of Post delete stored procedure operation bindings, as attribute values may need to be cached before the target object is deleted.

Full

Specifies that all attribute values are cached. This is the default for scripts.

Exists

Specifies that the Java Connector Server checks that the object being deleted exists. If the object does not exist, the Java Connector Server throws an LdapNameNotFoundException.

None

Specifies that the method or scripting payload is responsible for determining whether the target object exists. If you select Full or Exits, the framework determines whether the target object exists automatically. This means that the Java Connector Server passes the call on, and that the script or stored procedure you call should determine if the object exists.

Procedure

Specifies the stored procedure you want to run before, instead of, or after a selected operation.

In / Out

Indicates the type of parameter.

In

Specifies that the parameter or type passes the value into the stored procedure.

InOut

Specifies the stored procedure passes the value both in and out.

Out

Specifies that the stored procedure passes out the value.

Parameter

Specifies that the name of the parameter as specified in the store procedure's definition.

Data Type

Displays the SQL type for the named parameter.

Attribute

Specifies the attribute that maps to the parameter for the stored procedure.

Note: The values in this list with asterisks on either side are runtime context values for the specified operation. For example, the Distinguished Name for the account targeted by an operation or operation-specific value such as the "new name" in a MODIFY RN rename operation. The remaining values are the attribute values you previously mapped for the specified class.

Name

Defines the target object's most nested RDN value.

DN

Defines the target object's full distinguished name.

ErrorStatus

Specifies that the connector uses the attribute to pass a descriptive error string back from a stored procedure. This can cause failure if *strict completion* is true for the binding.

AddModify_AttrsAsXML

Specifies that the entire Add or Modify operation is passed in as a single XML string.

Note: This can be useful for stored procedures bound to ADD or MODIFY operations where the endpoint has good in-built XML support. The entire request can be passed in a single XML string that can then be broken down within the stored procedure, if desired.

ModifyRn_NewRdn

Defines the new RDN.

Move_NewParentName

Defines the new parent name.

MoveRename_NewParentName

Defines the new parent name.

MoveRename_NewRdn

Defines the new RDN.

CLASS

Defines the name of the object type.

OPERATION

Defines the name of the operation, for example, Add, Delete.

ALIAS

Defines the alias for the script.

Note: If you select more than one object class, only the union set of attributes for those object classes and the runtime attributes (such as *NAME*, *DN*) is available in this field.

Multivalued

Specifies that the parameter can have multiple values and needs a flattening style.

Flattening Style

Specifies a flattening method used to represent multiple values in a single string literal. If the attribute is multivalued, use the Flattening Style column.

Flattening Mode

Displays the flattening mode for multivalued parameters where changes can be expressed in terms of values ADDed or REMOVEd, or alternatively, by passing in the complete new list of values (REPLACE).

More information:

Operation Bindings (see page 54)
Stored Procedures (see page 55)
Bind Operations to Stored Procedures (see page 56)
Stored Procedure and Column Considerations (see page 59)

Operation Bindings – Operations Editor

This dialog displays a summary of the operation bindings for the currently selected operation. Operations are performed in the order displayed in the list. You can use this dialog to bind a script or stored procedure to an operation and change the order in which the operating bindings are executed.

This dialog contains the following fields that are not self-explanatory:

Available object classes

Displays the available mapped object classes.

Filtered object classes

Lets you filter the opbindings that are displayed in the Before, Instead of and After lists. Operation bindings for the classes displayed in this list are displayed in these lists.

The selections you make in this list are also displayed in all Filtered object classes lists in all Opbinding Operation editors. Also any changes to you make to this list in other Operation binding editors are also displayed in this list.

Before

Displays the operation binding that the connector performs before the specified operation. The connector executes the operation binding in the order displayed in the list.

Instead Of

Displays the operation binding that the connector performs instead of the specified operation. The connector executes the operation binding in the order displayed in the list.

After

Displays the operation binding that the connector performs after the specified operation. The connector executes the operation binding in the order displayed in the list.

Create

Displays the Create Operation Bindings dialog which lets you specify operation binding information.

Delete

Deletes the currently selected operation binding.

Move Up

Changes the order in which the selected operation binding is performed.

Move Down

Changes the order in which the selected operation binding is performed.

Preferences Dialog

You can use the Connector Xpress Preferences Dialog to change various aspects of the way Connector Xpress looks and behaves, such as its look and feel, search limits, and command parameters.

This dialog contains the following fields:

Search Size Limit

Specifies the maximum number of search results the Provisioning Servers tree attempts to retrieve for a particular node. Connector Xpress uses this value to prevent a large drain on system resources, for example, if many policies exist.

Show Extended Metadata

Displays an extended set metadata on several dialogs that contains all additional metadata that is relevant to this data model item.

Command to launch web browser

Specifies the command used to launch the web browser.

Default: (Windows) "rundll32" "url.dll,FileProtocolHandler"

Default: (Solaris)"/usr/sfw/bin/mozilla"

More Information:

Set Preferences (see page 22)

Provisioning Server Details Dialog

The Provisioning Server Details dialog lets you configure a connection to a Provisioning Server to which you can deploy a connector.

This dialog contains the following fields:

Host Name

Defines the name of the host where Provisioning Server runs.

User Domain

Defines the domain containing the user whose credentials the connector uses to log on to the Provisioning Server.

User Name

Defines the username that Connector Xpress uses to log on to the server.

Use TLS

Specifies that a secure TLS connection to the server is established. This means that the connector does not transmit the password in clear text. If not selected, the connector transmits the password in clear text.

Key

Defines the name of the configuration option that the connector uses to connect to the Provisioning Server.

Value

Defines the value of the configuration option.

Key column

Displays the full list of configuration options that the connector uses to connect to the Provisioning Server.

Value column

Displays the corresponding values for the configuration options displayed in the Key column.

More Information:

Add and Configure a Provisioning Server (see page 73)
Edit Provisioning Server Details (see page 74)

Provisioning Server Password Required Dialog

The Provisioning Server Password Required dialog lets you specify the password for the selected Provisioning Server.

This dialog contains the following field:

Password

Specifies the password required to access the selected Provisioning Server.

More Information:

Deploy the Connector (see page 34)

Script Editor Dialog

This dialog lets you specify the parameters for a script style operation binding.

This dialog contains the following fields:

Available object classes

Displays the available mapped object classes.

Added object classes

Lets you specify the object classes you wan to add to this script operation binding.

Execute this script

Specifies the script you want the connector to execute before, after instead of, or after a selected operation.

Description

Describes your operation binding.

Abort operation if script fails

Specifies how errors are handled. For example, if the timing is set to Before or After, and this check box is selected, exceptions are thrown and the operation aborts if any errors occur. If this check box is cleared, the connector logs the error but the operation continues if an error occurs.

If the timing is set to Instead Of, and an error occurs, the connector always throws exceptions and aborts the operation when executing the operation binding.

Lookup Level

Specifies whether the attributes or the deleted object are cached for the opbinding to function and controls the lookup of all an object's values. Caching is typically done preemptively for Post delete operations.

The default for scripts is Full. For methods, the framework verifies all the parameters mapped for the Post delete stored procedure and selects Full or Exists depending whether non-contextual attributes are present or not, respectively.

Note: The LookUpLevel attribute is important to the handling of Post delete stored procedure operation bindings, as attribute values may need to be cached before the target object is deleted.

Full

Specifies that all attribute values are cached. This is the default for scripts.

Exists

Specifies that the Java Connector Server checks that the object being deleted exists. If the object does not exist, the Java Connector Server throws an LdapNameNotFoundException.

None

Specifies that the method or scripting payload is responsible for determining whether the target object exists. If you select Full or Exits, the framework determines whether the target object exists automatically. This means that the Java Connector Server passes the call on, and that the script or stored procedure you call should determine if the object exists.

Execute Directly

Specifies that the connector performs the operation binding directly, rather than generating text that the ScriptStyleOpProcessor runs later.

Search Asynchronously

If selected specifies that the execution of the payload occurs asynchronously and a NamingEnumeration is used to return results in a streaming fashion for one-level and subtree searches.

Note: Connector Xpress disdplays this field when you select the Search operation from the Available Operations dialog in the Create Operation dilaog.

Execute a function in a global script

Specifies that a function in a global script is executed as the script for this operation binding.

Global Script

Specifies the global script from which the function is selected.

New

Displays the Edit Script dialog which lets you create a new global script.

Function Name

Defines the function that is executed in the selected global script.

Execute an individual script

Specifies the individual script that is executed for this operation binding.

Edit Script

Displays the Edit Script dialog which lets you add or modify the individual script.

Script Name Dialog

This dialog lets you modify the parameters for a global script.

This dialog contains the following fields:

Script Language

Specifies the available scripting languages that Java Connector Server framework currently supports.

Execute Directly

Specifies that the connector performs the operation binding directly, rather than generating text that the ScriptStyleOpProcessor runs later.

More information:

Operation Bindings (see page 54)
Scripts (see page 55)
Bind Operations to Scripts (see page 57)

Scripts Dialog

This dialog lets you specify the parameters for a global script.

This dialog contains the following fields:

Add New Global Script

Creates a script and adds a new global script node under the Scripts node.

Delete Global Script

Deletes the selected global script and removes the script under the Scripts node.

Select Data Source for new project Dialog

The Select Data Source for new project dialog lets you select a previously configured data source for a new project, add or edit an existing data source, or remove a data source.

This dialog contains the following fields:

Name

Displays the name of a configured data source you can use to create a project.

Type

Displays the type of configured sources you can use to create a project.

Add

Displays the Source Types dialog where you can choose from available source types and then configure your new data source.

Edit

Displays the Edit Source dialog where you can edit the connection details for the specified data source.

Remove

Removes the data source from the list box.

More information:

Create a Project (see page 26)
Set Up Data Sources (see page 14)

Select Template Dialog

This dialog lets you select a template to use as a starting point for mapping common endpoint schemas. Starting a project with a template is useful for LDAP, where standards such as RFCs define widely used schemas, and where purely custom schemas are less common. Templates also include specialized JavaScript mark-up that CA Identity Manager uses to render account management screens.

You can click each template to see a brief description.

Open in Wizard Mode

Starts a project using the wizard, and steps you through the basic process of mapping an account class, or an account class and group class, depending on the template you selected.

More information:

Templates (see page 24)

Source Types Dialog

The Source Types dialog lets you select an available source type for the data source.

This dialog contains the following fields:

None

Lets you create attributes and edit generic metadata that is not intended for deployment without specifying a data source.

Available Source Types

Displays the available source types you can use as the basis for your connector definition.

JDBC

Lets you map to database tables and stored procedures.

JNDI

Lets you map to object classes and attributes contained in a directory server.

More Information:

Create a Project (see page 26)
Set Up Data Sources (see page 14)

Wizard Summary Dialog

This dialog displays a summary of mappings you have made.

More information:

Create a Project Using the Wizard (see page 28)