MANAGING THE BUSINESS OF IDENTITY

SailPoint IdentityIQ Connector Factory

Version 6.4

User's Guide

Table of Contents

Chapter 1: Overview

This chapter presents the following topics:

About this book

This book contains information about the SailPoint Identity Connector Factory. It is intended for developers of customized Connectors for systems to be managed using IdentityIQ (IIQ).

This book has the following parts:

- **Getting Started**: contains installation and configuration information.
- **Developing IdentityIQ Direct Connectors**: contains instructions for creating and maintaining IdentityIQ Direct Connector

SailPoint IdentityIQ Connector Factory

The SailPoint IdentityIQ Connector Factory (referred to as the **Connector Factory**) is the primary development tool for SailPoint IdentityIQ Connectors. Working within the Connector Factory environment simplifies the process of designing and developing Connectors by breaking down the development process into clearly identifiable steps.

Key features of the Connector Factory are:

- **Guided Processes:** Connector Factory leads you step-by-step through the process of designing and developing IdentityIQ PE based Connectors (written in Java or Tcl or Perl) or IdentityIQ Direct Connectors required to integrate your Managed System into Service management.
- **Ease of Use:** A startup wizard guides you through the process of setting up a new project.
- **Choice of Development Language:** The Connector Factory allows you to develop function Connectors in one of three **programming languages: Java, Perl, or Tcl.**
- **Choice of Development Editor:** You can develop function Connectors using the built-in editor, or you can choose to employ any other text editor or development environment that you prefer.
- **Integrated Testing Environment:** You can test your new functions from within Connector Factory without having to compile and deploy new Connectors.
- **Ease of generation:** Connector Factory generated Connectors are totally ready for deployment.

  Note: IdentityIQ Connector Factory version 6.4 does not support multiple group schema feature.

Using SailPoint IdentityIQ Connector Factory

The following steps are involved in installing Connector Factory and using it to create, install, and configure a deployment Connector for a Managed System:

1. Install and configure Connector Factory.
   See "Chapter 3:Installing Connector Factory" on page 11 and "Chapter 4:Configuring Connector Factory" on page 21.
2. Define project details.

Use the Connector Factory New Project wizard to define project details, as well as Connector type and functionality, for a new project.

- For an IdentityIQ Direct Connector, see "Chapter 6: Wizard for IdentityIQ Direct Connectors"

3. Create project functionality.
   Use the Connector Factory Guided Development Environment to develop an IdentityIQ Connector to perform the functionality defined for the project.

   - For the Connector implementing IdentityIQ Direct Connector Interface , see "Chapter 8: IdentityIQ Direct Connector sample project"

   The end result of this process is a deployment Connector for deploying the new IdentityIQ Connector and (for Connectors only) for adding support for the new Managed System type in IdentityIQ.

Section 1: Getting Started

This section contains the information on the following:

- IdentityIQ architecture on page 9
- Installing Connector Factory on page 11
- Configuring Connector Factory on page 21
- Managing projects on page 27

Chapter 2: IdentityIQ architecture

This chapter discusses the following topics:

What is SailPoint IdentityIQ?

SailPoint IdentityIQ is a business-oriented identity governance solution that delivers risk-aware compliance management, adaptive role management, access request management, and identity intelligence. Some of the world's largest organizations are using IdentityIQ to improve security, minimize risk and streamline their compliance efforts.

IdentityIQ minimizes the access burden placed on IT staff by empowering end users across the organization to request and manage their own access.

Organizations strive for better visibility into potential risk factors across their business. With Identity Intelligence from IdentityIQ, organizations can transform technical identity data scattered across multiple enterprise systems into centralized, easily understood and business-relevant information. The visibility and insights offered by IdentityIQ through dashboards, risk metrics and reporting provide a clear understanding of identity and access information and help to proactively manage and focus compliance efforts strategically across even the most complex enterprise environments.

SailPoint IdentityIQ components

SailPoint IdentityIQ includes the core components illustrated in figure 1.

SailPoint IdentityIQ components

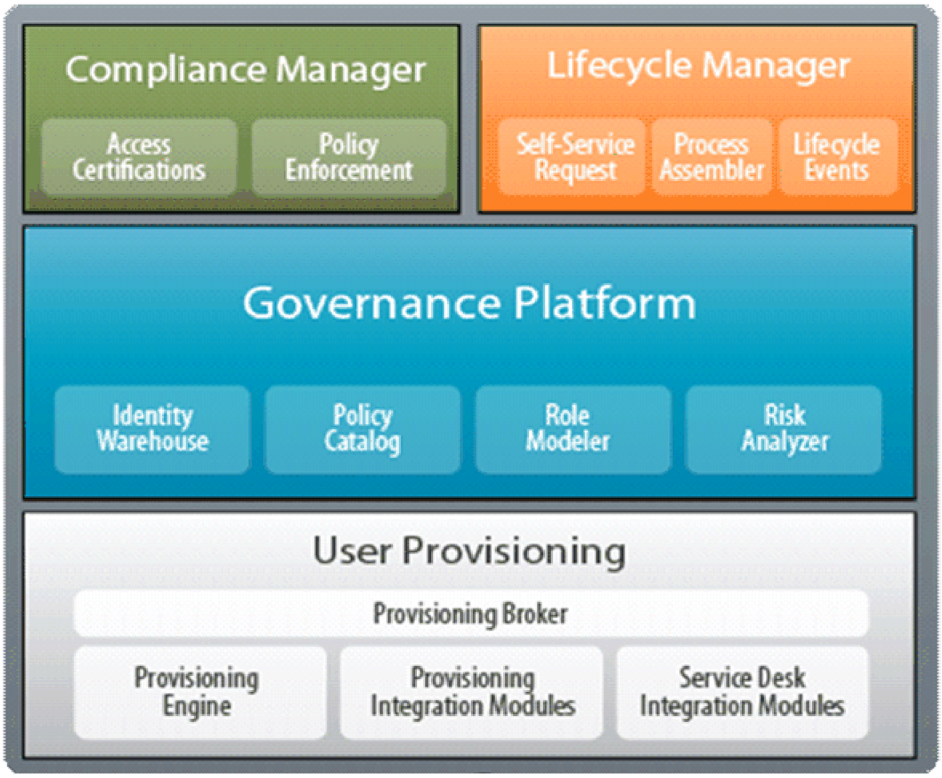


Figure 1—SailPoint IdentityIQ components

Together, these components provide a unique and comprehensive identity governance solution. The components are:

- **Compliance Manager**: streamlines complex compliance processes for greater efficiency, effectiveness and lower costs. Its integrated risk model provides the contextual framework that enables organizations to prioritize compliance activities and focus controls on the users, resources, and access privileges that represent the greatest potential risk to the business
- **Lifecycle Manager**: is a highly flexible solution for managing changes to user access and automating the initiation of provisioning activities within an enterprise. It maps directly to the lifecycle of a user within an organization (joining / moving / leaving) and the core identity business processes which support these lifecycle events (provision / change / deprovision)
- **Governance Platform**: to enhance compliance performance, improve security and reduce risk, Lifecycle Manager leverages the IdentityIQ Governance Platform to align provisioning requests with an organization's pre-established identity governance model. SailPoint's unique combination of roles, policy, and risk provides a strong framework for evaluating all requests for changes to access against predefined business policies.
- **Resource Connectivity**: The resource connectivity consists of the following modules:
  - IdentityIQ Provisioning Engine
  - Provisioning Integration Module
  - Service Desk Integration Module

Chapter 3: Installing Connector Factory

This chapter presents the following topics:

Operating system, software, and hardware requirements

IdentityIQ Connector Factory can be installed on the following Microsoft Windows operating systems:

- Microsoft Windows Server 2008 Standard Edition (32-bit and 64-bit)
- Microsoft Windows Server 2008 Enterprise Edition (32-bit and 64-bit)
- Microsoft Windows 8 PRO 1
- Microsoft Windows 7 Enterprise Edition SP1
- Microsoft Windows Vista Business Edition SP1 (32-bit and 64-bit)
- Microsoft Windows Vista Enterprise Edition SP1 (32-bit and 64-bit)
- Microsoft Windows Vista Ultimate Edition SP1 (32-bit and 64-bit)
- Microsoft Windows Server 2003 R2 Standard Edition SP2 (32-bit and 64-bit)
- Microsoft Windows Server 2003 R2 Enterprise Edition SP2 (32-bit and 64-bit)
- Microsoft Windows Server 2003 Standard Edition SP2 (32-bit and 64-bit)
- Microsoft Windows Server 2003 Enterprise Edition SP2 (32-bit and 64-bit)

Note: **IdentityIQ Connector Factory can be installed on a 64-bit operating system, but runs as a 32-bit application.**

The minimum hardware requirements for SailPoint IdentityIQ Connector Factory are:

- 1 GB memory
- 1 GB disk space
- 1 GHz processor

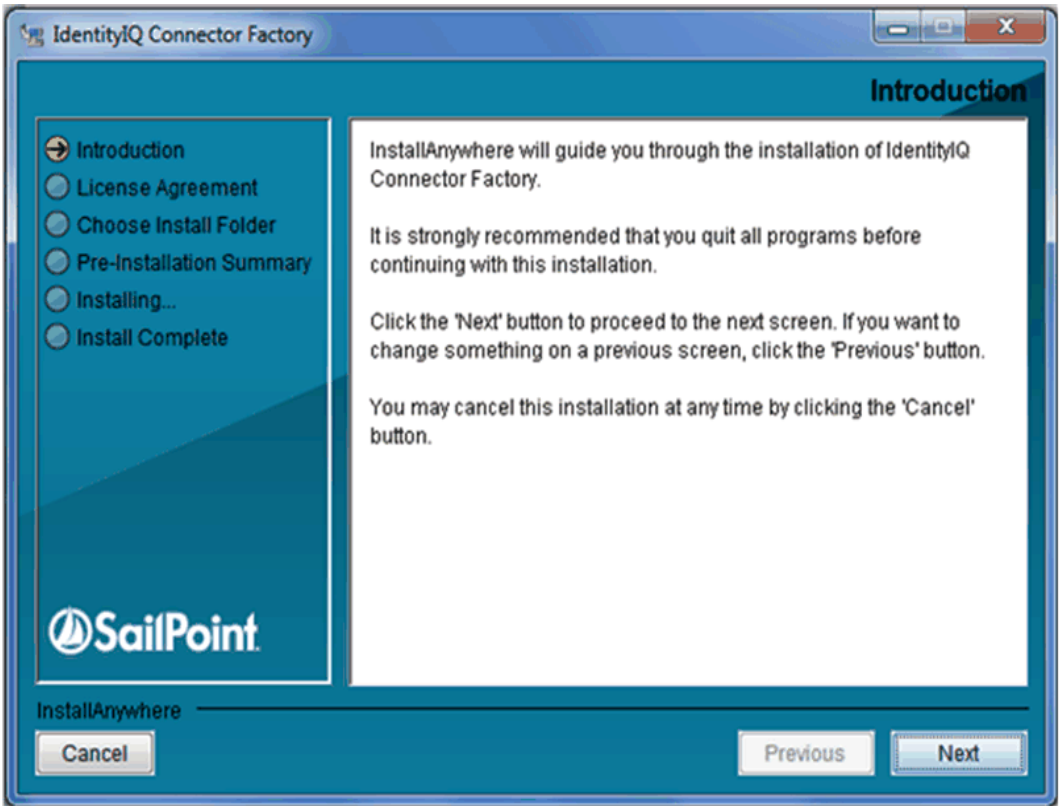
Installing Connector Factory

Follow this procedure to install a fresh installation of the Connector Factory. Currently no upgrade is provided to update an existing version of the Connector Factory.

1. Insert the Connector Factory installation CD into a CD drive.
2. From the Start menu, select **Run** and type the following command in the **Command Line** field, where *drive* is the drive in which you placed the installation CD:
   drive:**\Install\IdentityIQ Connector Factory Installer.exe**

   For example, **E:\Install\IdentityIQ Connector Factory Installer.exe**

   The InstallAnywhere wizard Welcome screen (Figure 2—Installation Wizard - Welcome screen) is displayed.



InstallAnywhere will guide you through the installation of IdentityIQ Connector Factory.

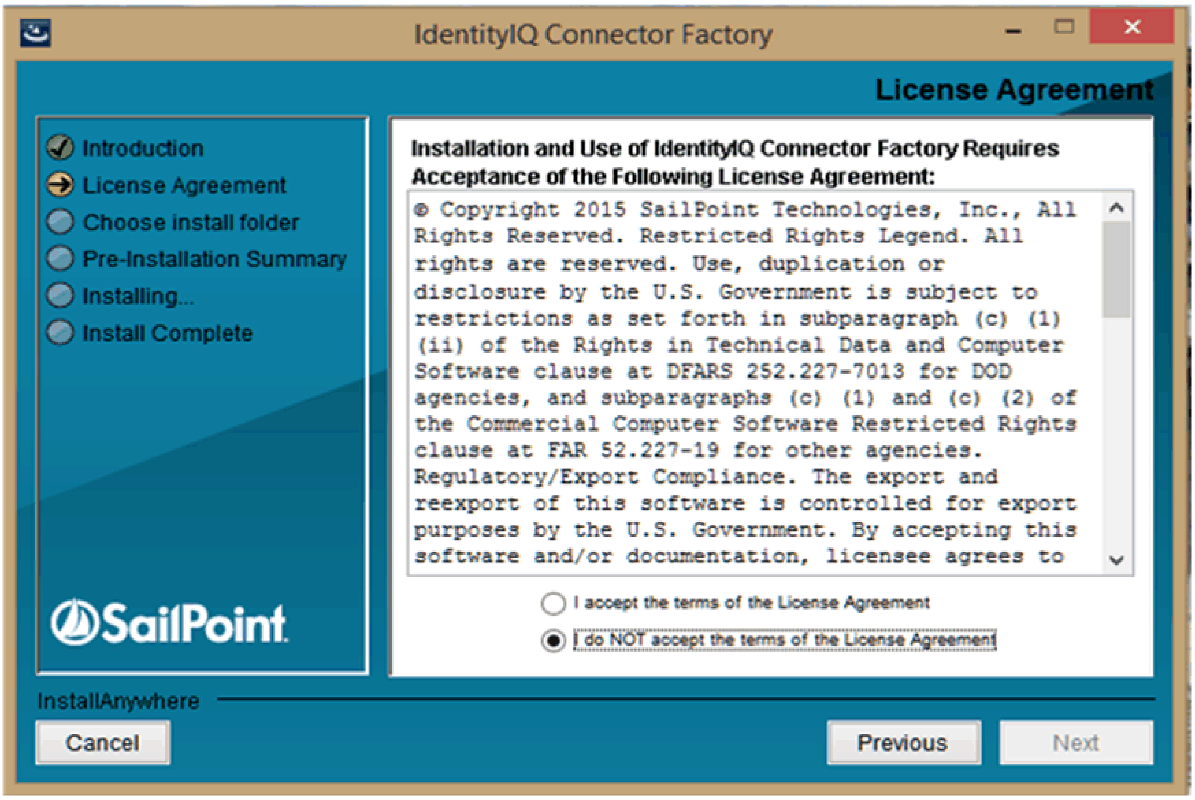
It is strongly recommended that you quit all programs before continuing with this installation.

Click the 'Next' button to proceed to the next screen. If you want to change something on a previous screen, click the 'Previous' button.

You may cancel this installation at any time by clicking the 'Cancel' button.

Figure 2—Installation Wizard - Welcome screen

3. Click **Next** to continue.
   The License Agreement screen (Figure 3— Installation wizard - License screen on page 13) is displayed.



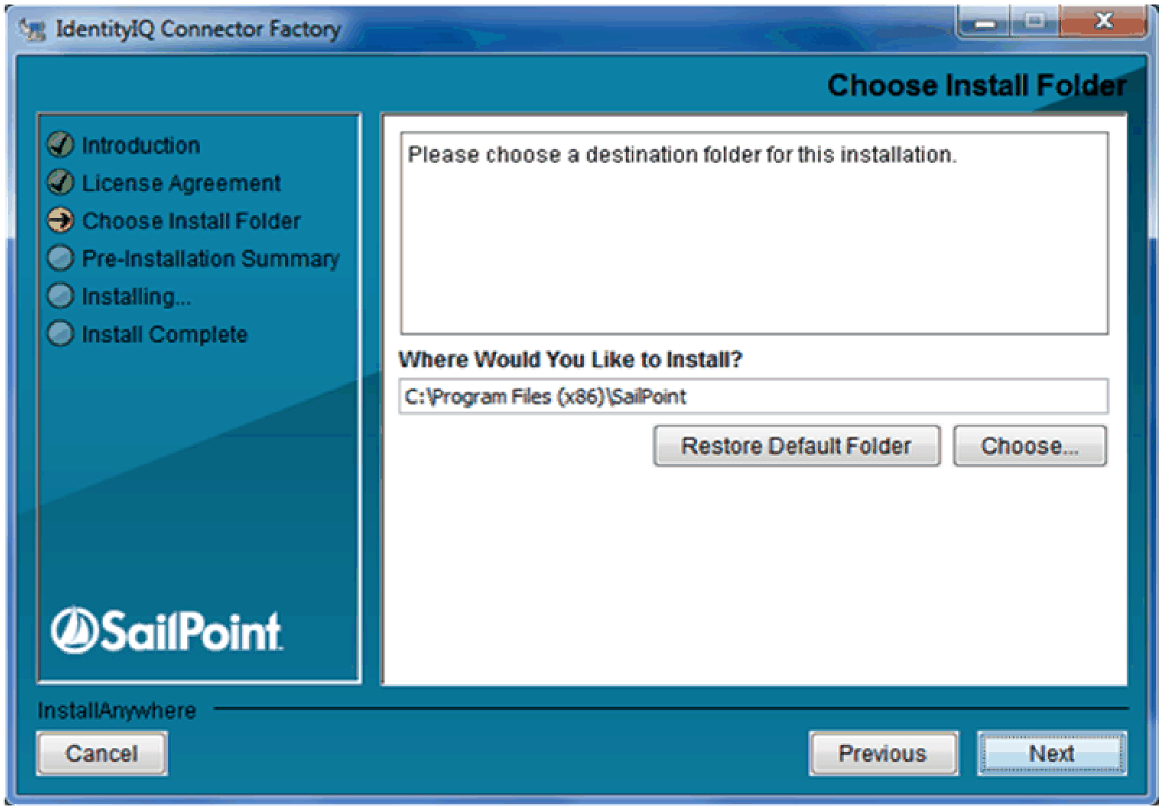
Installation and Use of IdentityIQ Connector Factory Requires Acceptance of the Following License Agreement:

```
© Copyright 2015 SailPoint Technologies, Inc., All
Rights Reserved. Restricted Rights Legend. All
rights are reserved. Use, duplication or
disclosure by the U.S. Government is subject to
restrictions as set forth in subparagraph (c) (1)
(ii) of the Rights in Technical Data and Computer
Software clause at DFARS 252.227-7013 for DOD
agencies, and subparagraphs (c) (1) and (c) (2) of
the Commercial Computer Software Restricted Rights
clause at FAR 52.227-19 for other agencies.
Regulatory/Export Compliance. The export and
reexport of this software is controlled for export
purposes by the U.S. Government. By accepting this
software and/or documentation, licensee agrees to
```

I accept the terms of the License Agreement [✓ I do NOT accept the terms of the License Agreement]

Figure 3—Installation wizard - License screen

4. Select the **I accept the terms of the License Agreement** and click **Next** to continue.
5. Select the Install Folder screen (Figure 4—Installation wizard - Select the Install Folder screen) is displayed.



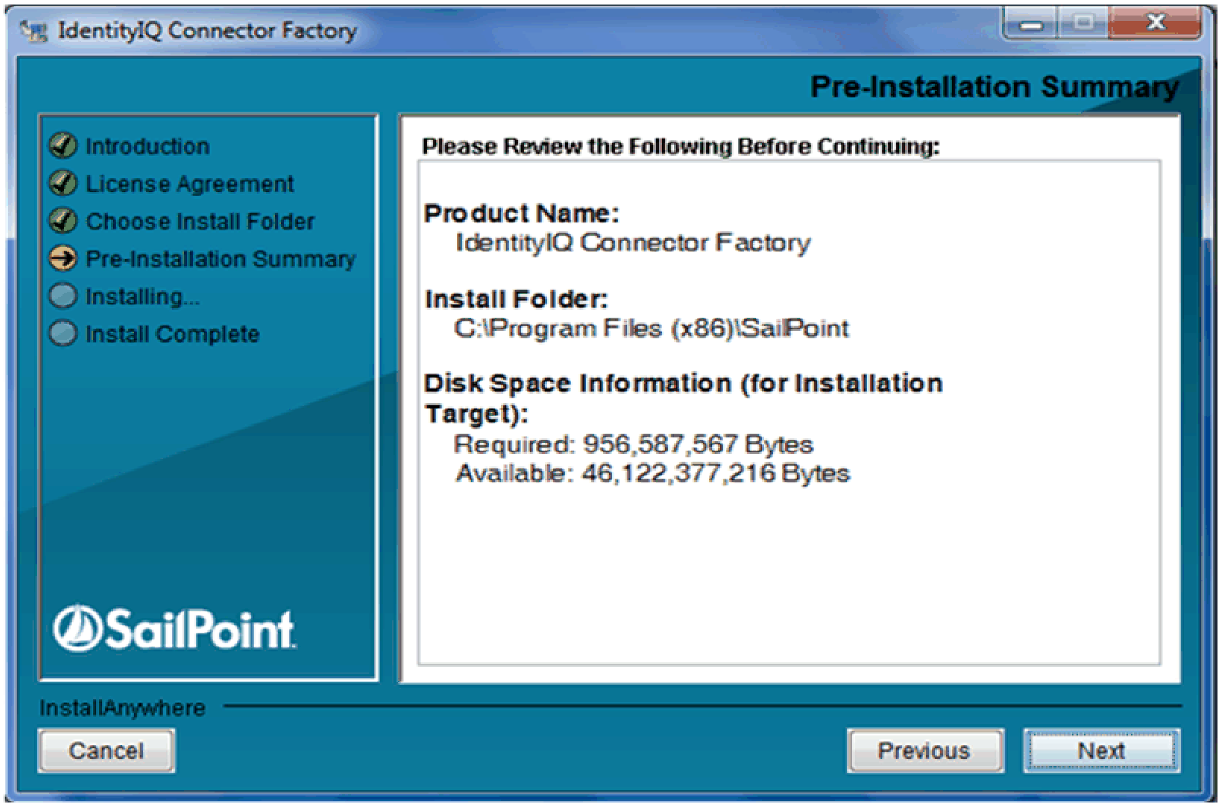
Please choose a destination folder for this installation.

Where Would You Like to Install?

C:\Program Files (x86)\SailPoint

Figure 4—Installation wizard - Select the Install Folder screen

6. Click **Next** to accept the default location for the installation, or click **Choose** to navigate to a new location for the installation.
   A Summary screen of the installation location and size is displayed.


IdentityIQ Connector Factory

Pre-Installation Summary

Please Review the Following Before Continuing:

Product Name:
IdentityIQ Connector Factory

Install Folder:
C:\Program Files (x86)\SailPoint

Disk Space Information (for Installation Target):
Required: 956,587,567 Bytes
Available: 46,122,377,216 Bytes

Figure 5—Installation wizard - Pre-Installation Summary screen

7. Click **Next** to start the installation.

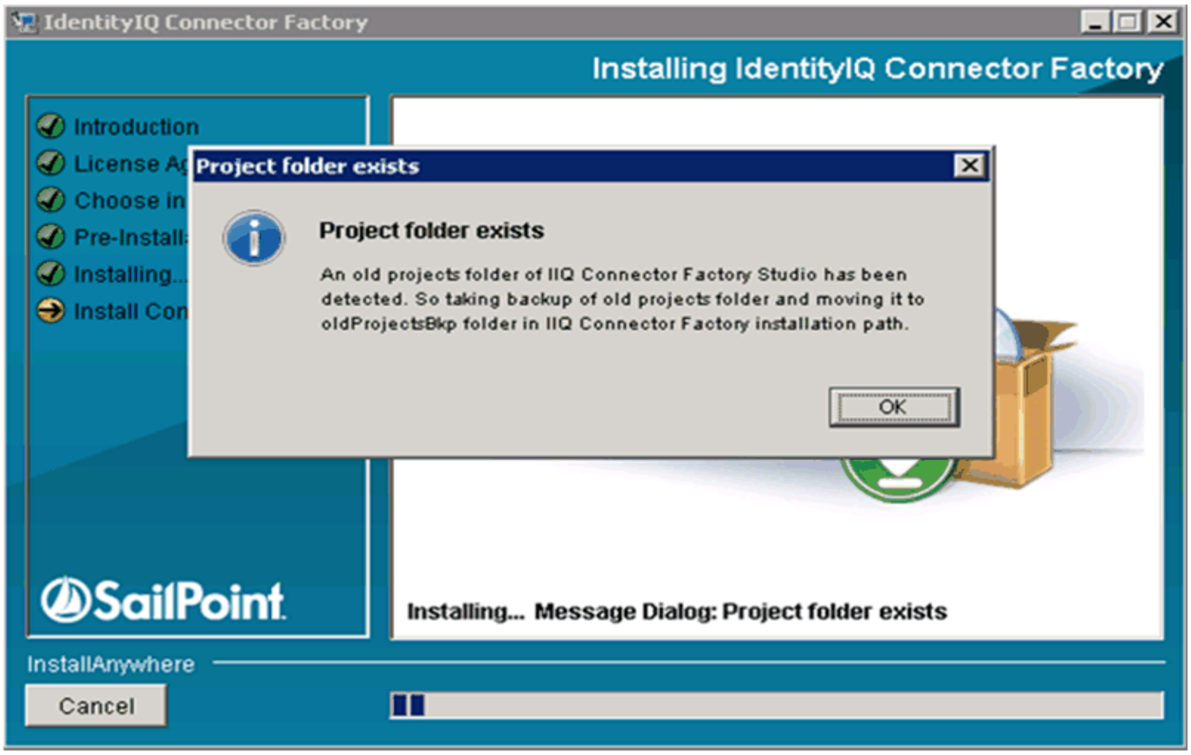
- If the installation path of the Connector Factory is the same as the earlier version of the Connector Factory that was uninstalled, and the Projects directory of the earlier version still exists, the Old Projects folder screen (Figure 6— Installation Wizard - Install complete screen on page 16) is displayed.

  Click **Next** to create a backup of the existing Projects folder and continue the installation. Existing projects will be copied to the **oldProjectsBkp** directory.

  Note: If the installation path of the new installation is not the same as the earlier version of the Connector Factory, import the Projects saved from an earlier installation. For more information on importing projects, see “Creating a new project” on page 27**.**

- If no Connector Factory installation exists, the installation screen (When the installation is completed, an Installation Complete screen is displayed informing you of the successful completion of the installation.) is displayed.

8. Click **Install**.
   The installation progress screen is displayed along with the Project folder exists pop up window as displayed below:

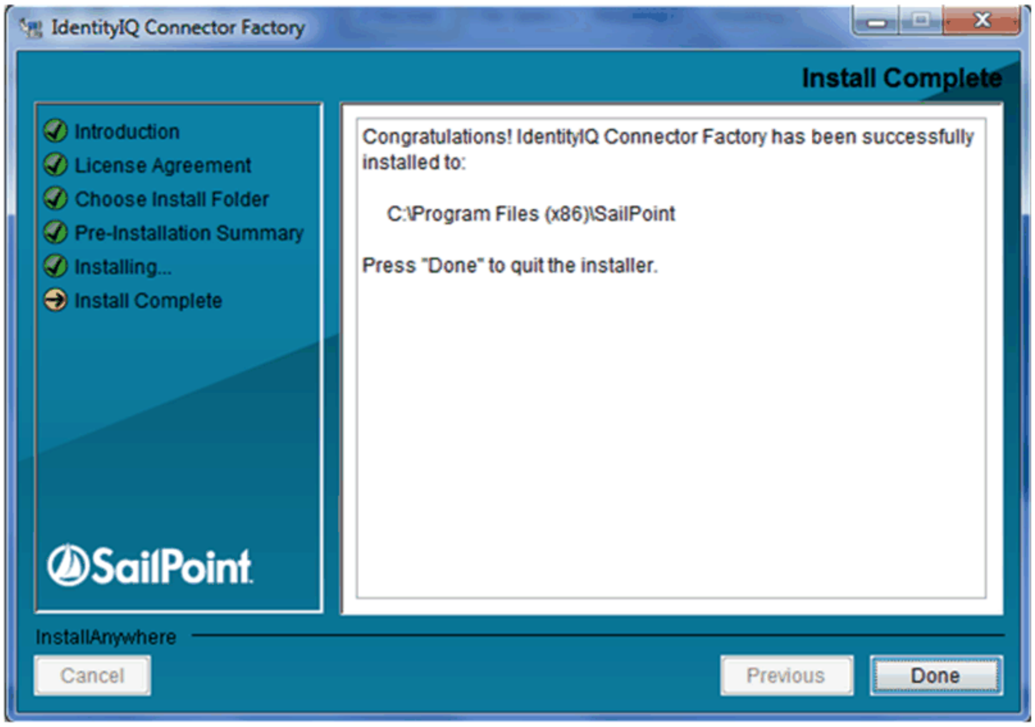


Project folder exists

An old projects folder of IIQ Connector Factory Studio has been detected. So taking backup of old projects folder and moving it to oldProjectsBkp folder in IIQ Connector Factory installation path.

Installing... Message Dialog: Project folder exists

When the installation is completed, an Installation Complete screen is displayed informing you of the successful completion of the installation.



Congratulations! IdentityIQ Connector Factory has been successfully installed to:

C:\Program Files (x86)\SailPoint

Press "Done" to quit the installer.

Figure 6—Installation Wizard - Install complete screen

9. Click **Done** to complete the installation.

In the installation directory a number of subdirectories and files are created for the Connector Factory components. You will use the following files and directories when using Connector Factory:

- The **IIQConnectorFactory.bat** file, located directly under the installation directory, activates Connector Factory.
- The **Generated Install** subdirectory will contain the output of any projects you generate using Connector Factory.
- The **cfg** directory contains the **Studio.properties** file. This file consists of the Connector Factory configuration parameters. One of these parameters is the path to an external text editor. You can modify this path, in order to configure any external text editor that you choose.
- JDBC drivers are provided to enable you to work with IdentityIQ PE based RDBMS Connectors:
  - jconn2.jar for working with Sybase
  - jtds-1.2.jar for working with MSSQL
  - ojdbc14.jar for working with Oracle®.

  These drivers are located in the ***connectorFactoryHome*\jars\jdbc_drivers** directory, where *connectorFactoryHome* is the installation directory.

Uninstalling Connector Factory

1. Close any Connector Factory applications that are running.

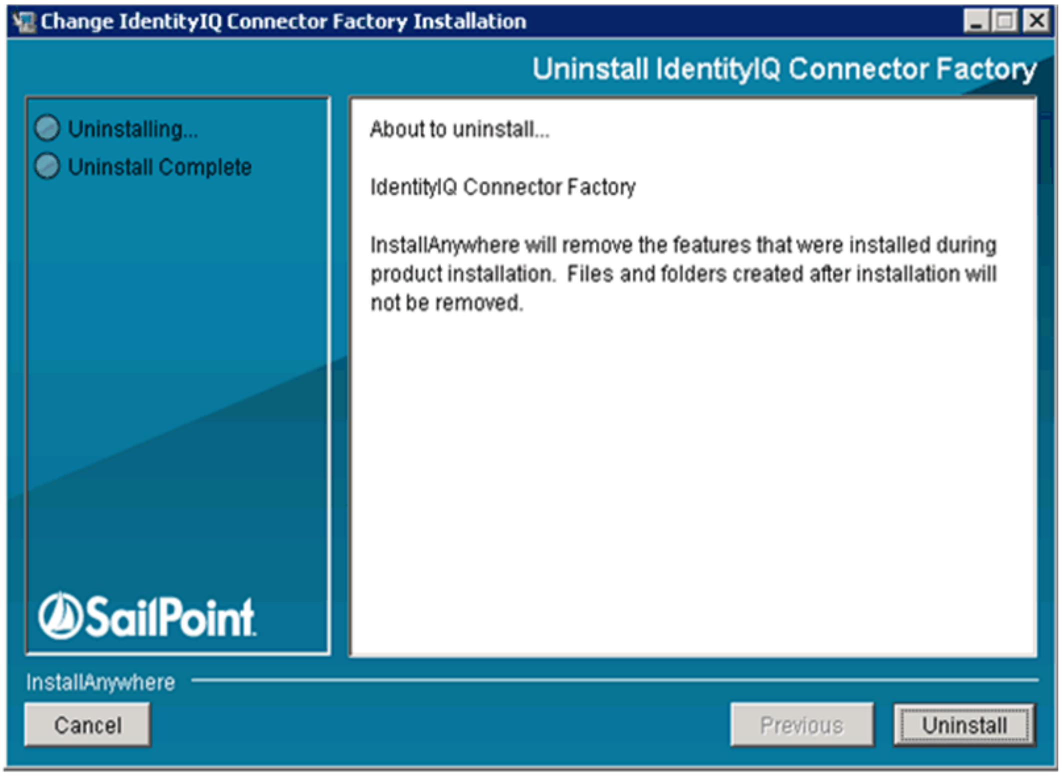
   Note: SailPoint recommends that you close all active Windows applications before starting the uninstall procedure.

2. Do one of the following:

- Select **Start => Programs => IdentityIQ Connector Factory=> Uninstaller**.
- From the Windows Control Panel, select the **Add or Remove Programs** option.

  In the Add or Remove Programs dialog, select IdentityIQ Connector Factory from the Currently installed programs list and click **Change/Remove**.

  The Uninstallation screen is displayed.


Change IdentityIQ Connector Factory Installation

Uninstall IdentityIQ Connector Factory

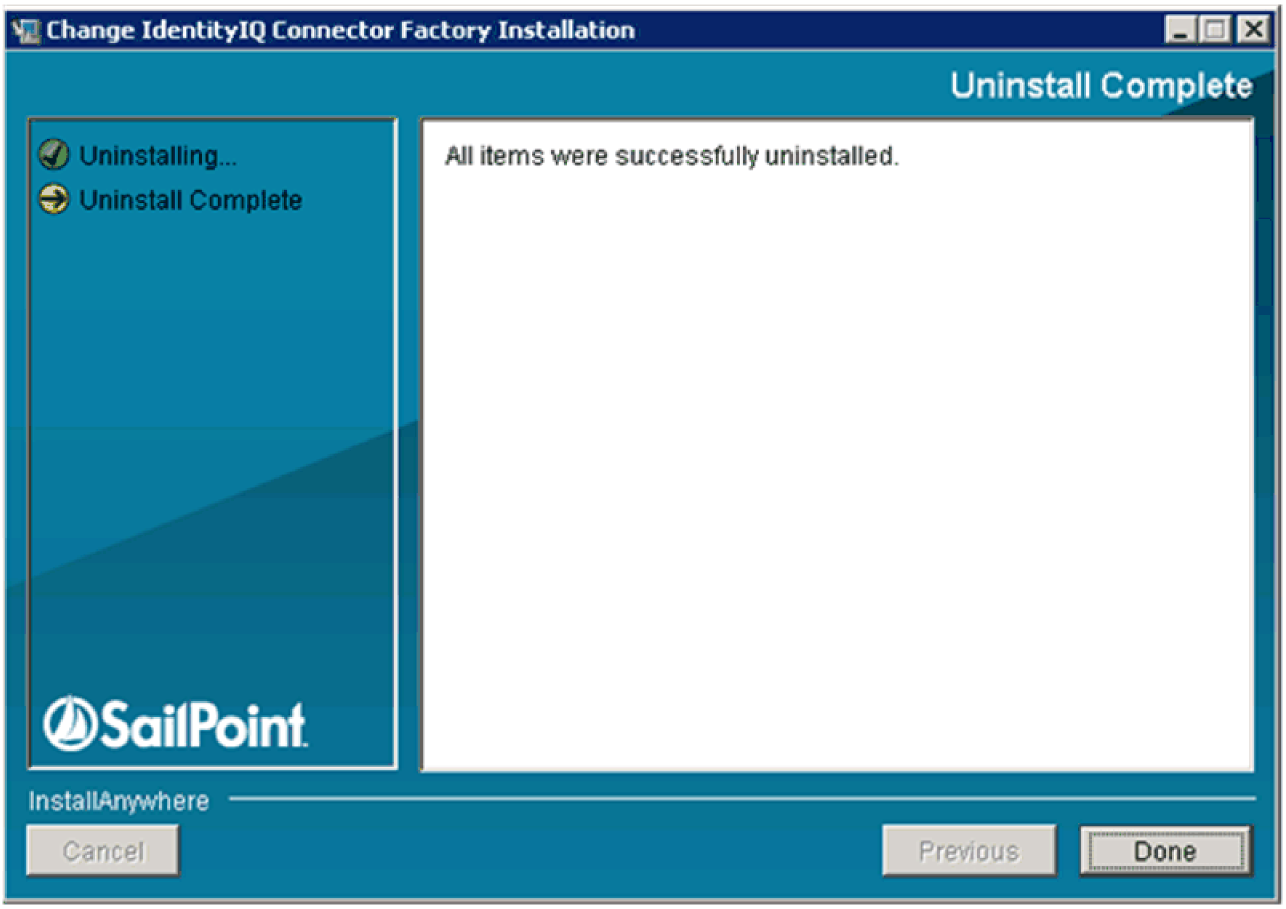
About to uninstall...

IdentityIQ Connector Factory

InstallAnywhere will remove the features that were installed during product installation. Files and folders created after installation will not be removed.

Figure 7—Uninstallation wizard - Uninstallation screen

3. Click **Uninstall** to continue.
   When the Connector Factory application is successfully removed from your computer, a confirmation screen (Figure 8—Uninstall complete screen) is displayed.



All items were successfully uninstalled.

Figure 8—Uninstall complete screen

Chapter 4: Configuring Connector Factory

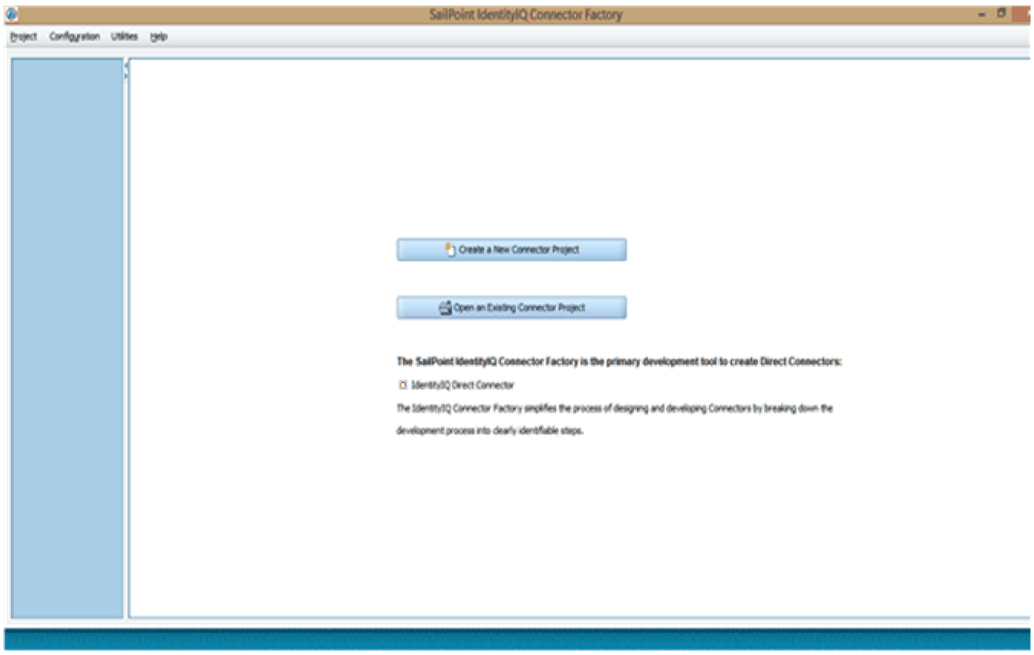
This chapter presents the following topics:

Accessing Connector Factory

You can access the Connector Factory using one of the following options:

- Click **Start => Programs => IdentityIQ Connector Factory => ConnectorFactory**
- In Windows Explorer, navigate to the directory where Connector Factory is installed, and double-click the **ConnectorFactory.bat** file.
- Click the Connector Factory desktop shortcut Icon.

The Connector Factory main window (Figure 9) is displayed.



SailPoint IdentityIQ Connector Factory
Project
Configuration
Utilities
Help
Create a New Connector Project
Open an Existing Connector Project

The SailPoint IdentityIQ Connector Factory is the primary development tool to create Direct Connectors:

- IdentityIQ Direct Connector

The IdentityIQ Connector Factory simplifies the process of designing and developing Connectors by breaking down the development process into clearly identifiable steps.

Figure 9—Connector Factory main window

Configuring an external editor

Connector Factory has a built-in text editor for editing function code. This editor offers convenient features, such as automatic insertion of Keywords and parameter names into the function code. However, in addition to the built-in editor, you may also use any other external text editor for writing your code. You will need to use the editor when you are developing an IdentityIQ PE based Connector.

To configure an external text editor, perform the following:

1. Using a text editor, open the following file, where *connectorFactoryHome* represents the directory in which Connector Factory was installed:
   ***connectorFactoryHome*/cfg/Studio.properties**

   Note: If Connector Factory is active, you can open the Studio.properties file by choosing Configuration => Studio Properties from the menu bar.
2. Search the file for the following entry:
   externaleditorpath=null
3. Change this entry to **externaleditorpath=*fullPathName***
   where *fullPathName* represents the full path and name of the external text editor that you want to use.

   **Note: You must use forward slashes (/) in the path.
   For example, externaleditorpath=D:/WinNT/notepad.exe**

   When using Notepad as an external editor, make sure that Word Wrap is not activated (Menu =>Format). If Word Wrap is selected, clear it.
4. Save the **Studio.properties** file.
5. After updating **Studio.properties**, restart Connector Factory.

When using Connector Factory to edit function code, use the **Open External Editor** option to open your function code in this external editor. For more information, see “Editing Function code” on page 96.

JDK configuration

Before developing an IdentityIQ PE based Connector in Java, you must first configure the JDK that will be used to compile and deploy the project. You can use an existing JDK profile, or you can add a new profile.

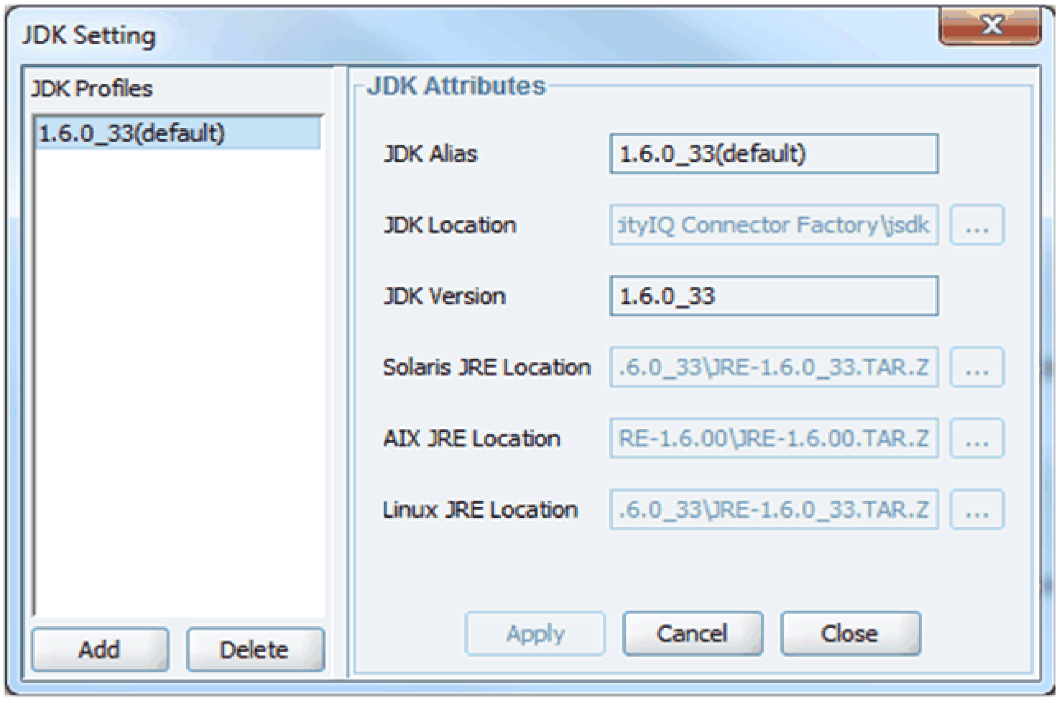
The Connector Factory installation contains JDK 1.6.0_33 This is the default configuration for the JDK configuration. However, you may choose to work with another JDK. Connector Factory supports JDK 1.4.2_11 and higher.

Note: Due to security issues, SailPoint strongly recommends that you use installed JDK version 1.6.0_33.

Configuring an existing JDK profile

To configure an existing JDK profile, perform the following:

1. Access the IdentityIQ Connector Factory as described in “Accessing Connector Factory” on page 21.
2. From the Connector Factory menu bar, select **Configuration => JDK Configuration**.
   The JDK Setting dialog (Figure 10— JDK Setting dialog on page 23) is displayed.



1.6.0_33(default)

JDK Alias: 1.6.0_33(default)

JDK Location: ityIQ Connector Factory\jsdk

JDK Version: 1.6.0_33

Solaris JRE Location: .6.0_33\JRE-1.6.0_33.TAR.Z

AIX JRE Location: RE-1.6.00\JRE-1.6.00.TAR.Z

Linux JRE Location: .6.0_33\JRE-1.6.0_33.TAR.Z

Figure 10—JDK Setting dialog

3. Select the required JDK profile from the left hand pane, update the parameters in the right pane, click **Apply** and **Close**.

Configuring a new JDK profile

To configure a new JDK profile, perform the following:

1. Access the IdentityIQ Connector Factory as described in "Accessing Connector Factory" on page 21.
2. From the Connector Factory menu bar, choose **Configuration => JDK Configuration**.
   The JDK Setting dialog is displayed.
3. Click **Add**.
4. In **JDK Alias**, type a unique name for the profile.
5. In **JDK Location**, enter the full path of the JDK home directory.
6. For projects that will be compiled with UNIX®; in **Unix JRE Location**, enter the full path to the zipped TAR file of the UNIX JRE. This TAR file will be enclosed in the JAR, extracted at install time, and used at runtime. The file name must follow the convention **JRE-*JREVersion*.TAR.Z**. For example, **JRE-1.5.0_11.TAR.Z**.

   The zipped tar file should be prepared as follows:
   a. Locate the JDK installation directory on the Solaris™ or AIX® system. For example,
      /usr/j2sdk_1.5.0_11/
   b. Enter the following command to access the JRE directory under the installation directory:
      cd /usr/j2sdk_1.5.0_11/jre
   c. Enter the following commands:
      tar -cvf ~/JRE-1.5.0_11.TAR *
      cd ~

You should now have the correct file. Copy it to the Connector Factory computer, and point to it as described in step 6.

7. Click **Apply**.
8. To add more profiles, repeat from step 3. to step 7.
9. Click **Close**.

Note: For custom JDK configuration the selected JDK version should be compatible with the corresponding Solaris or AIX JRE version.

Dynamic classpath

The start batch IIQConnectorFactory.bat dynamically adds all JAR files in the *connectorFactoryHome***\jars,** ***connectorFactoryHome*****\jars\user** and *connectorFactoryHome***\jars\jdbc_drivers** directories to the classpath.

This is done for RDBMS projects, as Connector Factory needs the JDBC driver in the classpath to connect to the Data Source. Connector Factory is supplied with JDBC drivers for all supported databases (MSSQL, Sybase and Oracle). To use different JDBC drivers from those supplied, you must put the JDBC jar files in the ***connectorFactoryHome*****\jars\jdbc_drivers** directory before starting Connector Factory.

In addition, the *connectorFactoryHome***\jars\user** directory is added to the JVM environment variable **java.library.path** to support JNI libraries. If you use third party libraries (JARS) which use JNI (DLL files) then these DLLs should be placed under the *connectorFactoryHome***\jars\user** directory so they will be in the JVM classpath when executing Connector Factory tests.

Increasing application memory

Due to extensive use of logs in the testing window, running many data intensive tests in the Connector Factory could cause the application to stop due to a lack of memory. Changes were implemented to avoid the use of excess memory and increase the process memory.

If you choose to continue running extensive tests with an extensive use of logs, you must manually increase process memory.

To increase process memory to 500MB, perform the following:

1. Open the following file in a text editor:
   \Program Files\SailPoint\IdentityIQ Connector Factory\ConnectorFactory.bat
2. Modify the following line::
   start jsdk\bin\javaw -Xmn50M -Xms100M -Xmx200M -classpath %THE_CP% -Djava.library.path=%JNI_DLL_PATHS% %MAIN_CLASS%

   to

   start jsdk\bin\javaw -Xmn100M -Xms250M -Xmx500M -classpath %THE_CP% -Djava.library.path=%JNI_DLL_PATHS% %MAIN_CLASS%

   The variables in the commands are as below:

-Xmn	size of the "young generation" memory space for short-lived objects
-Xms	initial java heap size
-Xmx	maximum java heap size

3. Save and close the file.

Chapter 5: Managing projects

This chapter presents the following topics:

Before starting a new project

Each IdentityIQ Direct Connector in Connector Factory represents a Managed System type in IdentityIQ.

Before starting a new Connector project, you should complete the following activities:

- Analyze your Managed System to determine the type of functionality you want to support for it in IdentityIQ.
- Connector Factory offers a selection of predefined Connector types, each containing the skeleton for a specific set of Connector functions. Determine which of these Connector types is appropriate for your objectives.

Information on each of these activities can be found in the *SailPoint IdentityIQ Connector Factory Developer's Guide*.

In addition, you may want to review the type of information that you are required to provide when running the New Project wizard to start a new project. For more information, see the "Chapter 8: IdentityIQ Direct Connector sample project" chapter for the IdentityIQ Direct Connectors.

Creating a new project

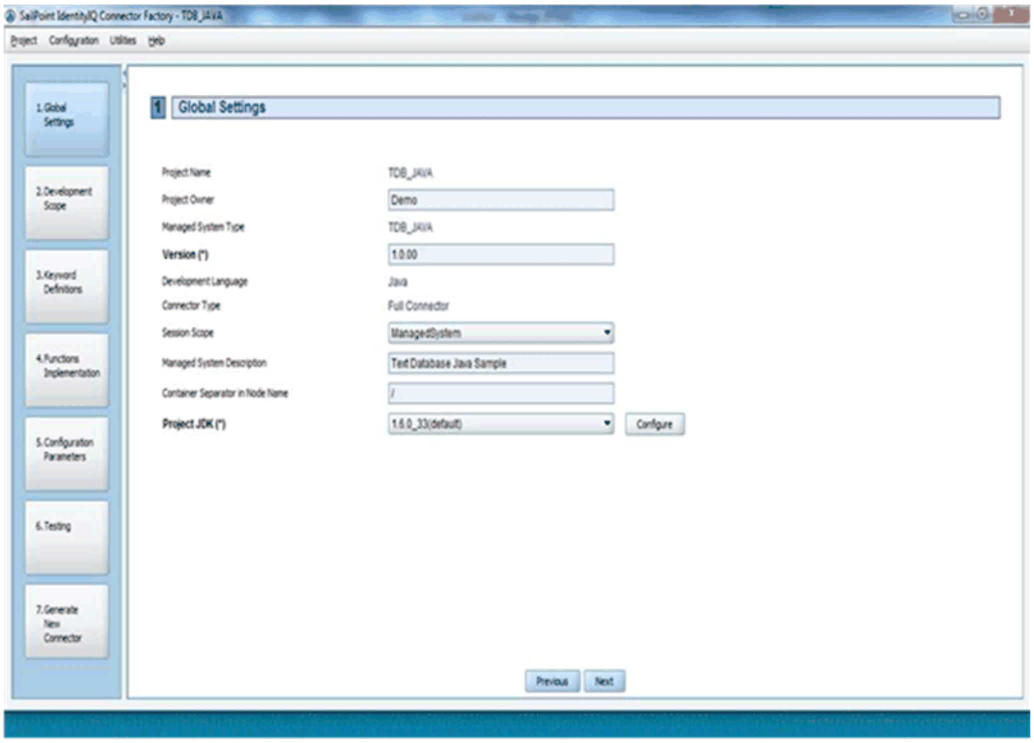
The first step in creating a customized Connector is to start a new project in Connector Factory. The New Project wizard guides you through the process of setting the project definitions. These definitions will determine the functionality of the project, and, therefore, the development requirements of your project.

To create a new project, perform the following:

1. From the Connector Factory menu bar, choose **Project =>New Project**
   OR
   Click **Create a New Connector Factory Project** in the main window
   OR
   Press **CTRL+N**. The New Project wizard starts.
2. Fill in the fields on each screen, and click **Next**.
3. When you have completed filling in the fields on the wizard, review them to make sure they actually fit your requirements, and click **Finish**.

   Note: After you click Finish in the last window of the wizard, it is not possible to change most of the project parameters.

   When you have completed the New Project wizard, the Global Settings window is displayed in the Guided Development Environment. The example shown in Figure 11 is for a IdentityIQ Direct Connector project.



Project Name	TDB_JAVA
Project Owner	Demo
Managed System Type	TDB_JAVA
Version (*)	1.0.00
Development Language	Java
Connector Type	Full Connector
Session Scope	ManagedSystem
Managed System Description	Text Database Java Sample
Container Separator in Node Name	/
Project JDK (*)	1.6.0_33(default)

Figure 11—Connector Factory Global Settings window

Each of the tasks to be performed in the process of defining the project appears on the task bar on the left side of the window.

For details on performing each task, refer to the“IdentityIQ Direct Connector sample project” on page 55.

Saving a project

Additions and changes that you perform while working on a project are saved to disk when you leave each step. Saving the current step changes without leaving the step can be done by using the **Save** menu.

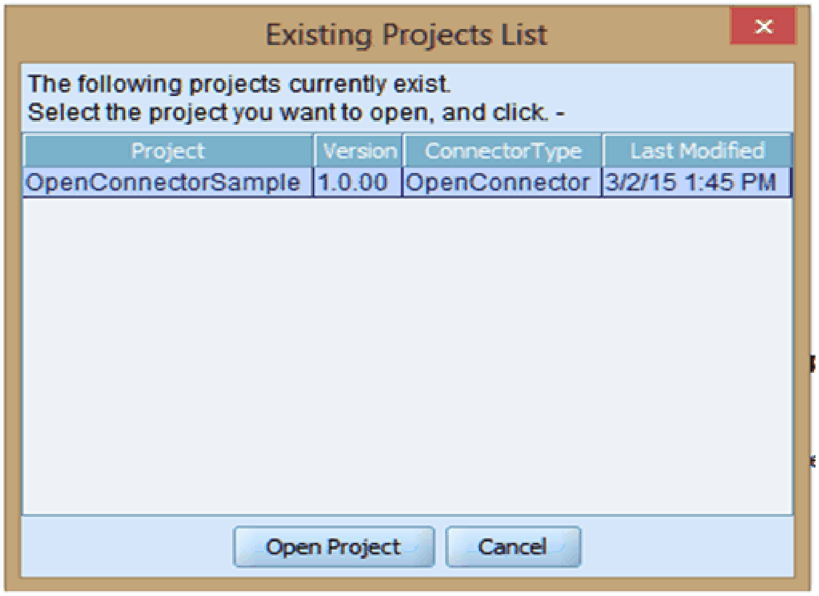
Opening an existing project

You can use the Guided Development Environment to modify an existing project.

To open an existing project, perform the following:

1. From the Connector Factory menu bar, choose **Project =>Open Project,** or click **Open an Existing Project** in the main window, or press **CTRL+O**.

   Note: If a different project was already open, it is automatically saved and closed
   The Existing Projects List dialog is displayed.


Existing Projects List

The following projects currently exist.
Select the project you want to open, and click. -

Project	Version	ConnectorType	Last Modified
OpenConnectorSample	1.0.00	OpenConnector	3/2/15 1:45 PM

Figure 12—Existing Project List dialog

2. Select an existing project, and click **Open Project**.
   The selected project is opened. If you have previously worked on this project in the Guided Development Environment, it opens to display the window in which you most recently viewed the project.
3. Use the Guided Development Environment to modify the project as necessary.

Deleting a project

1. From the Connector Factory menu bar, choose **Project =>Delete Project,** or press **CTRL+D**.
   The Existing Projects List dialog (Figure 12— Existing Project List dialog on page 29) is displayed.
2. Select the project that you want to delete, and click **Delete Project**.
3. Confirm that you want to delete the project.
   The project is deleted.

Closing a project

You can close an open project by choosing **Project => Close Project** from the menu, or pressing **CTRL+F4**.

Closing a project

Section 2: Developing IdentityIQ Direct Connectors

This section contains the information on the following:

- Wizard for IdentityIQ Direct Connectors on page 33
- Guided development environment for IdentityIQ Direct Connectors on page 39
- IdentityIQ Direct Connector sample project on page 55

Chapter 6: Wizard for IdentityIQ Direct Connectors

This chapter presents the following topics:

Overview

The IdentityIQ Connector Factory wizard enables you to quickly and easily define the following key elements in your new project:

- Project Details
  - The name, owner, and development language of the project
  - The name, version, and description of the Connector
- Project Type
  - IdentityIQ Direct Connector
- Connector Settings
  - **Connector Type Name**: The name of the Managed System type for which you are developing a Module. For example, Oracle®-9i or Active Directory.
  - **Connector Version Number**: The project version representing the milestone of the new Module you are developing. The project version format should be n.n.nn.
  - **Connector Description**: A summary description of the project including information such as the Managed System you are managing and the supported functionality scope.
  - **Class Name**: The name of the class that would implement the OpenConnector Interface.
  - **Form Name**: The name that would be used as a form name in the schema file.
  - **Feature String**: The value specified here can be used as a feature String in IdentityIQ schema file.

The Confirm Project Settings window is displayed. Review your choices, and click **Finish**. After you click **Finish**, the Guided Development Environment appears for your project, and you can no longer modify the project details that you have entered in the wizard.

General tips

Note the following when using the Connector Factory wizard:

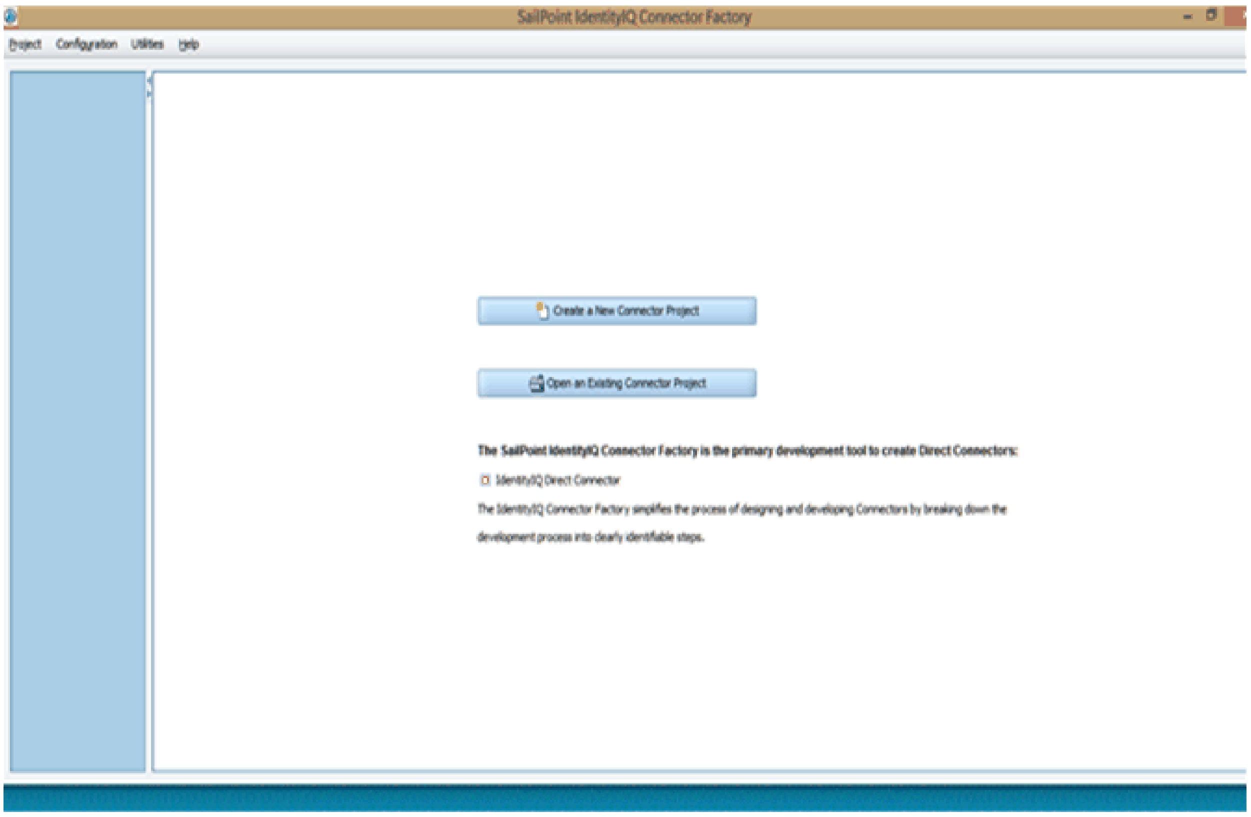
- Review the information in the wizard before you click **Finish** in its last window. After you click Finish you can no longer make modifications.
- Use the **Next** and **Previous** buttons to navigate back and forth between the wizard windows and change any of the information, as necessary.
- The status bar at the bottom of the window provides a visual indication of the progress of the New Project wizard as you proceed.
- The fields in the **New Project** wizard accept only characters that are valid for the field.
- Press ESC to close pop up dialogs.

Using the IdentityIQ Connector Factory Wizard for Connectors

This section describes how to use the IdentityIQ Connector Factory wizard to set the basic definitions for a new project to create an IdentityIQ PE based Connector.

To set definitions for a new project, perform the following:

1. IdentityIQ Connector Factory as described in "Accessing Connector Factory" on page 21.
   The Connector Factory window (Figure 13—Connector Factory) is displayed.


SailPoint IdentityIQ Connector Factory
Project
Configuration
Utilities
Help
Create a New Connector Project
Open an Existing Connector Project

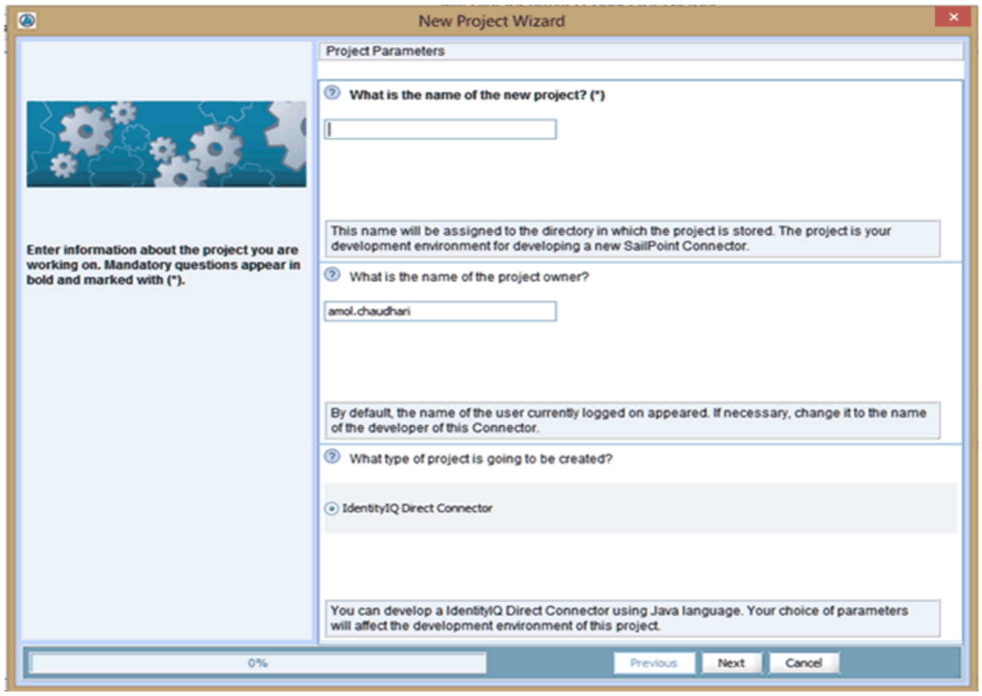
The SailPoint IdentityIQ Connector Factory is the primary development tool to create Direct Connectors:

- IdentityIQ Direct Connector

The IdentityIQ Connector Factory simplifies the process of designing and developing Connectors by breaking down the development process into clearly identifiable steps.

Figure 13—Connector Factory

2. From the menu bar choose **Project =>New Project**
   OR
   Click **Create a New Connector Factory Project** in the main window
   OR
   Press **CTRL+N**.
   The New Project Wizard (Figure 14)is displayed.



Enter information about the project you are working on. Mandatory questions appear in bold and marked with (*).

What is the name of the new project? (*)

This name will be assigned to the directory in which the project is stored. The project is your development environment for developing a new SailPoint Connector.

What is the name of the project owner?

amol.chaudhari

By default, the name of the user currently logged on appeared. If necessary, change it to the name of the developer of this Connector.

What type of project is going to be created?

[✓ IdentityIQ Direct Connector]

You can develop a IdentityIQ Direct Connector using Java language. Your choice of parameters will affect the development environment of this project.

0%

Figure 14—New Project Wizard

3. Specify the following details of the Connector development project:

   - **Project Name:** The project name. This name will be used as the name of the work directory for the Connector project.

   - **Project Owner:** The owner of the project. By default, the name of the user currently logged in appears as the owner, but you can modify it.

   - **Type of Project:** The type of project you wish to develop. To specify an IdentityIQ Direct Connector, select **IdentityIQ Direct Connector**.

   Click **Next**. The first **Connector Settings** window is displayed.

4. Specify the Connector settings:

   - **Connector Type Name:** The name of the Managed System type for which you are developing a Connector. For example, Oracle 9i or Active Directory.
     Note: The name that you specify is used as the Managed System type in IdentityIQ. The length of the name is limited to 12 characters.

   - **Connector Version Number:** The project version representing the milestone of the new Connector you are developing. The project version format should be **n.n.nn**.

   - **Project Description:** A summary description of the project including information such as the Managed System you are managing and the supported functionality scope.

   Click **Next.** The IdentityIQ Direct Connector's Parameter's window is displayed.

5. Specify the IdentityIQ Direct Connector Interface Parameter's:

   - **Class Name**: The name of the class that would implement the OpenConnector Interface.
   - **Form Name**: The name that would be used as a form name in the schema file.
   - **Feature String**: The value specified here can be used as a feature String in IdentityIQ schema file.

6. Click **Next**.
   The Confirm Project Settings window is displayed.
7. Review your choices, and click **Finish**.
   After you click **Finish**, the Guided Development Environment appears for your project, and you can no longer modify the project details that you have entered in the wizard.

Project directory structure

Each project has its own directory under ***connectorFactoryHome*\projects**. This directory contains the following sub directories:

- **third_party:** is the directory where all the libraries (jars, Perl/Tcl package) which are used for compilation (Java) or runtime are copied. All third party libraries under this directory will be copied to the JAR, and will be deployed and used in the installation environment.
- **work:** is a directory for your own requirements, and it will be copied directly to the JAR.
- **conf:** holds the configuration files generated by Connector Factory. The user can add configuration or script files under conf/win or conf/unix . The relevant file will be bundled into the JAR according to the target platform. This folder contains **OpenConnectorSample.xhtml** file which should be placed in identityiq application folder under **\build\define\applications**.
- **src:** directory is where all the source files for the Direct Connector Project are stored. If you wish to add your own package, you must store the source files in this directory.

Project directory structure

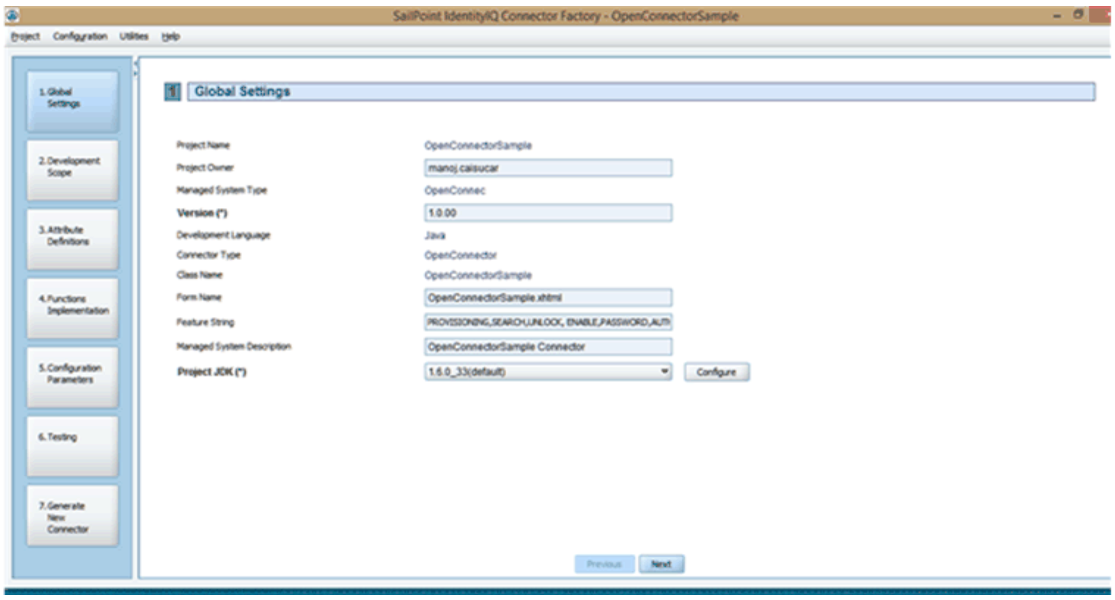
Chapter 7: Guided development environment for IdentityIQ Direct Connectors

This chapter presents the following topics:

Connector Factory window

Figure 15— Connector Factory - Global Settings window shows the Global Settings window of the Connector Factory, for IdentityIQ Direct Connector.

This is the first window displayed after you complete the New Project wizard. It is the starting point for developing your custom Connector.



Project Name	OpenConnectorSample
Project Owner	manoj.caisucar
Managed System Type	OpenConnec
Version (*)	1.0.00
Development Language	Java
Connector Type	OpenConnector
Class Name	OpenConnectorSample
Form Name	OpenConnectorSample.xhtml
Feature String	PROVISIONING,SEARCH,UNLOCK, ENABLE,PASSWORD,AUTI
Managed System Description	OpenConnectorSample Connector
Project JDK (*)	1.6.0_33(default)

Figure 15—Connector Factory - Global Settings window

The task bar

The task bar on the left side of the Connector Factory window provides access to each of the tasks to be performed when working on a project. The button for the task which is currently active appears bolded.

The tasks are presented in the logical sequence that they would typically be performed. However, at any point in the development process, you can move between tasks by clicking a task on the task bar or by clicking the **Previous** or **Next** buttons that appear at the bottom of the window.

Table 1— Guided development environment – Tasks for Connectors provides a brief description of each task. Each task is described in greater detail in this chapter.

Table 1—Guided development environment – Tasks for Connectors (Sheet 1 of 2)

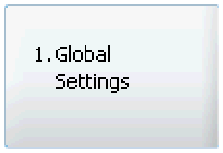
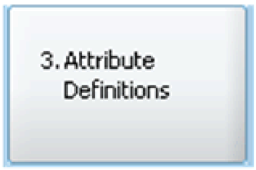
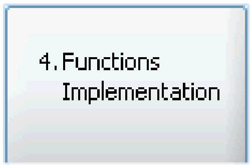
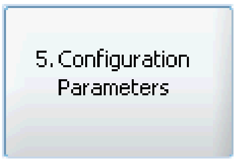
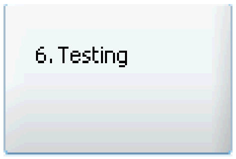
Task	Description
	Provides a display of project settings that were defined in the New Project wizard. Review this information. You can change fields which are open for update.
	Provides a (read-only) display of all entities and functions to be supported by the Connector created by Connector Factory, based on the information you provided in the New Project wizard. Review this information.
	Define attribute for each entity to be supported. Each Keyword represents an attribute of the entity. A corresponding field is created in the IdentityIQ database record for the entity.
	Write Connector functions using the provided built-in editor or use the external editor of your choosing.
	Provide default values for predefined parameters and custom parameters for the Managed System.

Table 1—Guided development environment – Tasks for Connectors (Sheet 2 of 2)

Task	Description
 6. Testing	Simulate your Connector without having to generate and deploy it.
7. Generate New Connector	Generate the deployment package. This package contains the installation files for the deployment of your new Connector on the Managed System and for importing the new Managed System type into IdentityIQ.

1 – Global Settings

The Global Settings window (Figure 15— Connector Factory - Global Settings window) displays properties for the current project, based on information you provided in the New Project wizard.

Review the information displayed before proceeding. If any of the read-only values are not correct, start a new project.

The Global Setting parameters are described in Table 2— Global setting parameters for Connectors.

Table 2—Global setting parameters for Connectors (Sheet 1 of 2)

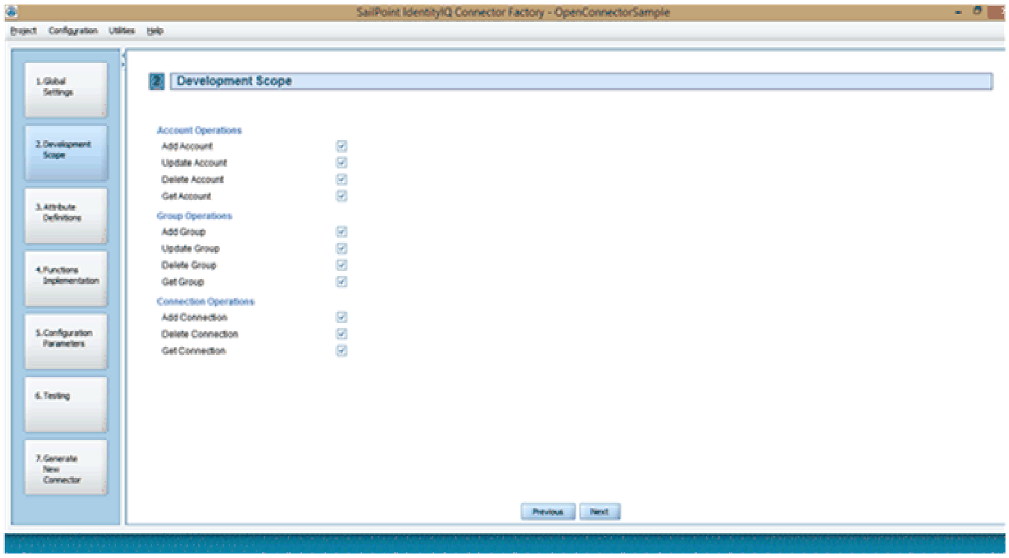
Parameter	Description
Project Name	The name assigned to the project. This is the name assigned to the directory in which the project is stored.
Project Owner	The person responsible for the project (descriptive information).
Managed System Type	The name to assign to this Managed System type in IdentityIQ.
Version	The version of the project in the correct format.
Development Language	The language to be used to develop the Connector functions. This determines the development environment and which function skeletons will be provided by Connector Factory.
Connector Type	Type of Connector to be developed. The type of Connector selected determines which Connector functions will be included in the Connector.
Class Name	The name of the class implementing the IdentityIQ Direct Connector.
Form Name	The name of the file that would be used in the schema files for filling the attributes information.
Feature String	The value specified here can be used as a feature string in IdentityIQ schema file.

Table 2—Global setting parameters for Connectors (Sheet 2 of 2)

Parameter	Description
Managed System Description	Description to assign to this Managed System type in IdentityIQ.
Project JDK	The version of JDK to be used with the Connector

2 – Development Scope

The Development Scope window indicates the Connector functions that you must develop in the current project, based on information you provided in the New Project wizard. For each function to be developed, the corresponding check box is selected. This information is read-only.


SailPoint IdentityIQ Connector Factory - OpenConnectorSample
Project Configuration Utilities Help
1.Global Settings
2.Development Scope
3.Attribute Definitions
4.Functions Implementation
5.Configuration Parameters
6.Testing
7.Generate New Connector
2 Development Scope
Previous
Next

Account Operations

Add Account	✓
Update Account	✓
Delete Account	✓
Get Account	✓

Group Operations

Add Group	✓
Update Group	✓
Delete Group	✓
Get Group	✓

Connection Operations

Add Connection	✓
Delete Connection	✓
Get Connection	✓

Figure 16—Development Scope Window for IdentityIQ Direct Connector

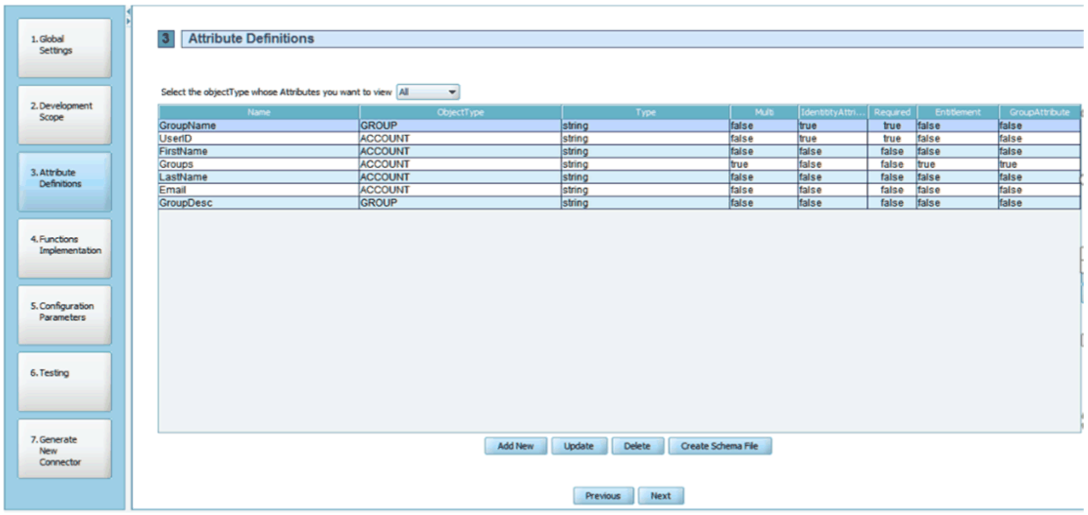
Review the information displayed before proceeding. If any of the values are not correct, start a new project.

For more information on Connector functions, see the *SailPoint IdentityIQ Connector Factory Developer's Guide*.

3 – Attribute Definition

This section describes the following:

- **Overview**: An Attribute in IdentityIQ Direct Connector defines the property of an entity (for example, **FirstName** in case of entity Account).



Select the objectType whose Attributes you want to view: All

Name	ObjectType	Type	Multi	IdentityAttri...	Required	Entitlement	GroupAttribute
GroupName	GROUP	string	false	true	true	false	false
UserID	ACCOUNT	string	false	true	true	false	false
FirstName	ACCOUNT	string	false	false	false	false	false
Groups	ACCOUNT	string	true	false	false	true	true
LastName	ACCOUNT	string	false	false	false	false	false
Email	ACCOUNT	string	false	false	false	false	false
GroupDesc	GROUP	string	false	false	false	false	false

Figure 17—Attributes Definition window

- **Using the Attribute Definitions window:** This section describes how to perform the following actions in the Attribute Definitions window.
  - Viewing the attributes
  - Creating a new attribute
  - Updating existing attributes
  - Deleting an existing attribute
  - Creating a schema file

Viewing the attributes

The Attribute Definitions window displays the attributes that would be used in the application deployed in the IdentityIQ environment.

The Attribute Definitions window displays the information in Table 3— Attribute Definition window parameters.

Table 3—Attribute Definition window parameters (Sheet 1 of 2)

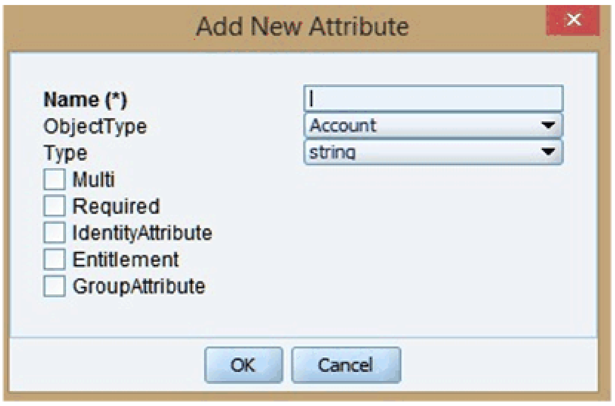
Field	Description
Attribute Name	The name of the attribute used in IdentityIQ. A mandatory field.
ObjectType	The type of object (Account, Group and Connection) to which the Keyword belongs.

Table 3—Attribute Definition window parameters (Sheet 2 of 2)

Field	Description
Type	The attribute data type. The following data types are supported: ■ string ■ int ■ char ■ boolean ■ date ■ secret
Multi	To indicate if the field is a list field. If this field is selected, you can have multiple values for the attributes.
Required	If checked is a required field.
IdentityAttribute	To indicate if the attribute is an identity attribute in IdentityIQ.
Entitlement	To indicate if the attribute is an entitlement attribute in IdentityIQ.
GroupAttribute	To indicate if the attribute is a group attribute in IdentityIQ.

Creating a new attribute

1. In the Attribute Definitions window, click **Add New.** The Add New Attribute dialog (Figure 18— Add New Attribute dialog) is displayed.


Add New Attribute

Name (*)
ObjectType: Account
Type: string
☐ Multi
☐ Required
☐ IdentityAttribute
☐ Entitlement
☐ GroupAttribute

Figure 18—Add New Attribute dialog

2. Enter values for the parameters. For more information, see Table 3— Attribute Definition window parameters on page 43.
3. Click **Add New**.
   The new attribute is added in the Attribute Definitions window.

   Note: **When a new attribute is added, a corresponding attribute is created.**
   The length of the keyword name, including the Managed System type prefix, should not exceed 26 characters.

Updating an existing attribute

1. In the Attribute Definitions window, perform one of the following:
   - Double-click the Attribute you want to update.
   - Select the Attribute you want to update, and click **Update.**

   The Update Attribute dialog is displayed.
2. Modify the attribute values as necessary.
3. Click **Update**. The Attribute is updated.

Deleting an existing attribute

1. In the Attribute Definitions window, select the Attribute you want to delete, and click **Delete Attribute.** A confirmation message appears.
2. Click **Yes**. The Attribute is deleted.

Creating a schema file

A file for importing Attributes is generated as part of the deployment package generated by IdentityIQ Connector Factory Studio. However, you have the option of generating the Attributes Schema file independent of the deployment package. For example, you can make minor changes and regenerate only the Schema file for a project which has already been deployed.

To generate the Schema file for Attributes, click **Create Schema File**. A **<ManagedSystem_type>_AppSchema.xml** file is created in the project directory.

4 - Functions Implementation

The Functions Implementation window is used to implement the Connector functions to be supported in the IdentityIQ Direct Connector.

This section describes the following:

- **Third Party configuration**: During the development of a new Connector Factory the developer may need to use the third party software. You have two options as to how to use the third party software:
  - **Third party directory**: Perform the following:
    a. Copy the required jars to the ***connectorFactoryHome*\projects\projectName\third_party** directory.
    b. If the jars use C libraries, copy them to the ***connectorFactoryHome*\projects\projectName\cmbin** directory. You may need to create the directory.

For more information on running the created program using the Testing window, see "Dynamic classpath" on page 24 section.

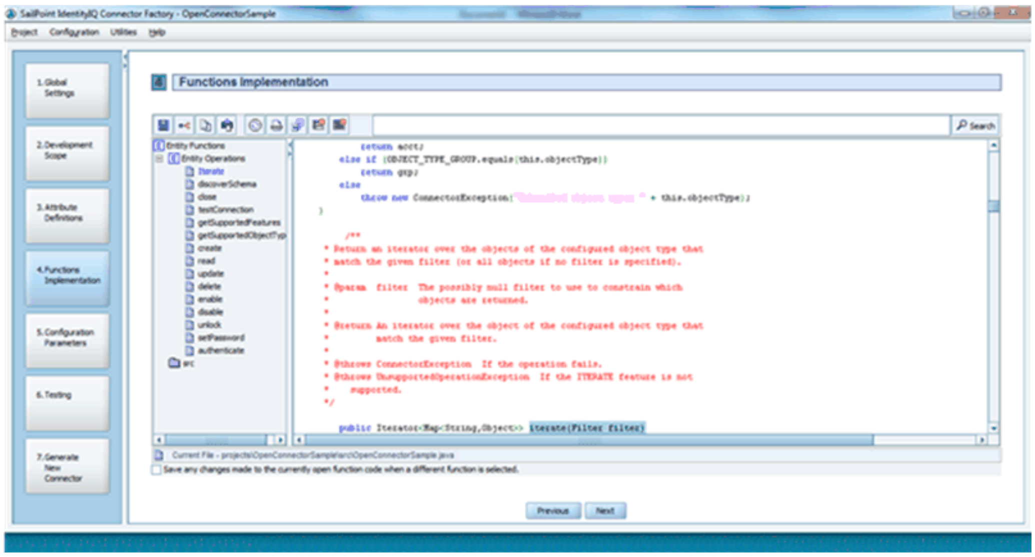
- **External library configuration**: There may be situations when you use third party libraries in the development environment, but you are unable to distribute them with the product (for example, because of legal limitations or version conflicts). In this case, these files should be declared in the **ExternalLibs.txt** file, which should be created in the ***connectorFactoryHome*\projects\projectName\location**.

  The **ExternalLibs.txt** file is used to add jars and directories to the Java CLASSPATH variable. Connector Factory uses these libraries during compilation and test step, but will not bundle them into the JAR file. A place holder Connector parameter will be defined for each third party after the Add Connector Factory process.

- **Overview**: When running the New Project wizard, you selected a Connector type for the project. These settings determine the Connector functions that you must implement for the project and the skeleton templates that Connector Factory provides for these functions.

  The loaded project file includes the Connector code infrastructure required for developing the new connector. This infrastructure includes skeleton function files with commented procedures.

  The Functions Implementation window (see Figure 19— Functions Implementation window) enables you to select each Connector function and edit the function, using the built-in editor or an external editor.



```
            return acct;
        else if (OBJECT_TYPE_GROUP.equals(this.objectType))
            return grp;
        else
            throw new ConnectorException("[illegible] " + this.objectType);
    }

        /**
     * Return an iterator over the objects of the configured object type that
     * match the given filter (or all objects if no filter is specified).
     *
     * @param  filter  The possibly null filter to use to constrain which
     *                 objects are returned.
     *
     * @return An iterator over the object of the configured object type that
     *         match the given filter.
     *
     * @throws ConnectorException  If the operation fails.
     * @throws UnsupportedOperationException  If the ITERATE feature is not
     *    supported.
     */

    public Iterator<Map<String,Object>> iterate(Filter filter)
```

Current File - projects\OpenConnectorSample\src\OpenConnectorSample.java

Save any changes made to the currently open function code when a different function is selected.

Figure 19—Functions Implementation window

- **Editing Function code**: When implementing Connector function files, you should refer extensively to the *SailPoint IdentityIQ Connector Factory Developer's Guide.*

  To edit a Connector function, perform the following:

  a. In the Function Editor, click the function in the list of function in the left pane of the editor window.

     The code for the function is displayed.

  b. Do one of the following:

     - **Edit the code using the built-in editor. Note that you can right-click in the built-in editor to copy and paste code.**

     - **Click Open External Editor to edit the code using the external editor you have configured.**

  Table 3— Function editing toolbar describes the function buttons available on the function editing toolbar.

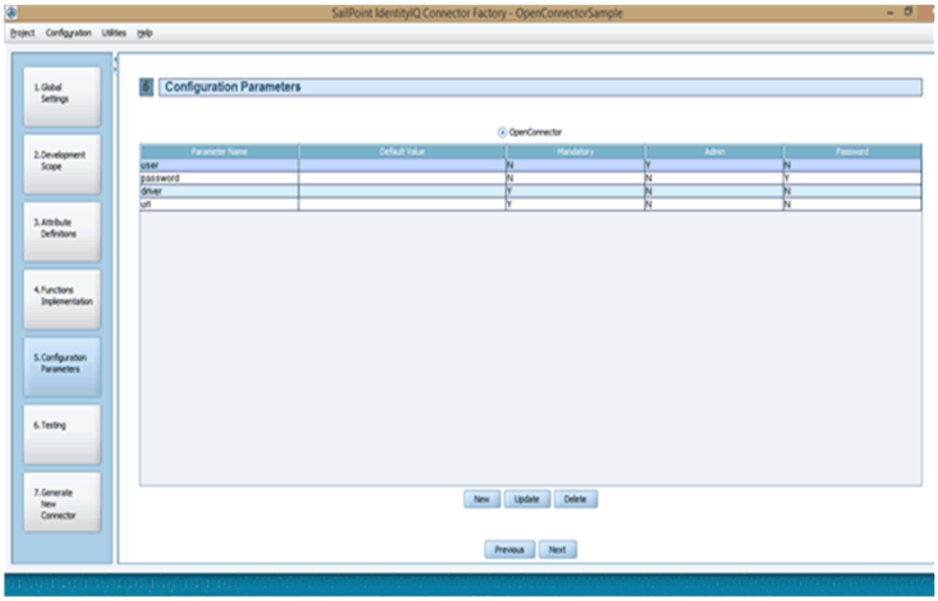
Table 3—Function editing toolbar

Function Button	Description
🖫	Saves updates to the current function file.
✂	Cuts selected text from the current function file to the clipboard.
⧉	Copies selected text from current function file to clipboard.
📋	Pastes clipboard text to selected destination.
!	This button is available only for projects whose development language is Java. Click to compile the code of the function currently open in the editor window. The Compile window displays messages regarding the progress of the compilation.
↗	Opens the selected function code in a predefined external editor. **Note:** The **Keywords List** and **Parameter List** buttons are only available when using the built-in editor. For information on how to define an external editor, see "Configuring an external editor" on page 21.
☰	Opens a dialog that lists the field names for all available Keywords. Select the desired field name, and click **Insert Attribute**. The selected field name is inserted at the current cursor position in the function code.
☰P	Opens a dialog that lists the Managed System parameters that you have defined in the Managed System Configuration window. Select the desired parameter, and click **Insert Parameter.** The selected parameter is inserted at the current cursor position in the function code.
Search	Searches for a specified text string. Type the text for which you want to search in the adjacent text box. Click again to jump to the next search result for the specified text string.
⟳	Re-reads the files in the work directory and updates the **Other Files** list.

- **Saving Function files**: All functions are contained in a single file. If you want the changes done to the currently open function code to be automatically saved when you click on a different function, select the Save any changes made to the currently open function code when a different function is selected check box. Otherwise, you will be prompted every time you switch between functions. You can also save changes by clicking the on the **Save** button.

5 - Configuration Parameters

The Configuration Parameters window (Figure 20— Configuration Parameters for Direct Connectors Window) is used to define parameters that would be used to talk to the Managed system.



Parameter Name	Default Value	Mandatory	Admin	Password
user		N	Y	N
password		N	N	Y
driver		Y	N	N
url		Y	N	N

Figure 20—Configuration Parameters for Direct Connectors Window

This section describes the following:

- **Overview**: By default the Direct Connector radio button is selected.
- **Using the Configuration Parameters window**: This section describes how to perform actions in the Configuration Parameters window.

  The parameters created would not be displayed in the schema file.

  - **Viewing Configuration Parameters**: The Configuration Parameters window displays the configuration parameters for your project. The parameters described in the following table (Table 4— Configuration parameters) are displayed:

Table 4—Configuration parameters

Parameters	Description
Parameter Name	Parameter name of the Direct Connector.
Mandatory	Define if the parameter is mandatory.
Default Value	This value is offered as the default response to the question.

  - **Working with configuration parameters**: This section describes the procedure for creating a new configuration parameter and updating and deleting an existing parameter.

    To create a new configuration parameter

    a. In the configuration parameter window, click **New**.

       The **Create New Parameter** dialog is displayed.

    b. Enter values for each of the fields. For more information, see Table 4— Configuration parameters.

    c. Click **Create New**. The new parameter is added in the configuration parameter window.

    To update an existing configuration parameter

    a. In the configuration parameter window (Figure 20— Configuration Parameters for Direct Connectors Window), double-click the parameter you want to update, or select it and click **Update**.

       The **Update Parameter** dialog is displayed.

b. Modify field values as necessary. For more information, see Table 4— Configuration parameters.
c. Click **Update**. The parameter is updated.

To delete an existing configuration parameter

a. In the configuration parameter window (Figure 20— Configuration Parameters for Direct Connectors Window), select the parameter you want to delete and click **Delete**. A confirmation message appears.
b. Click **Yes**. The parameter is deleted.

6 - Testing

The Testing window is used to simulate the work of Connector with following exceptions:

- In IdentityIQ Direct Connector interface module only Account, Group, and Connection entity are supported.
- Single administrator is enough to test the connector using testing framework. In testing framework the identity attribute would not be displayed as it is mapped to corresponding structured keyword of entity.

  For example, if an identity attribute USER_ID is defined for account using Attribute definition, then it would not be displayed for Account Entity operation. It would be internally mapped to XSA_ACCOUNT_NAME. Similarly the group identity attribute would be mapped to XSA_GROUP_NAME.

7- Generate New Connector

This section describes the following:

- **Overview**: The Generate New Connector window (Figure 21— Generate New Connector window) is used to generate the deployment package for IdentityIQ Direct Connector project which can be deployed in IdentityIQ environment. It can also be used to generate the deployment module for installing the Connector on the Managed System computer.

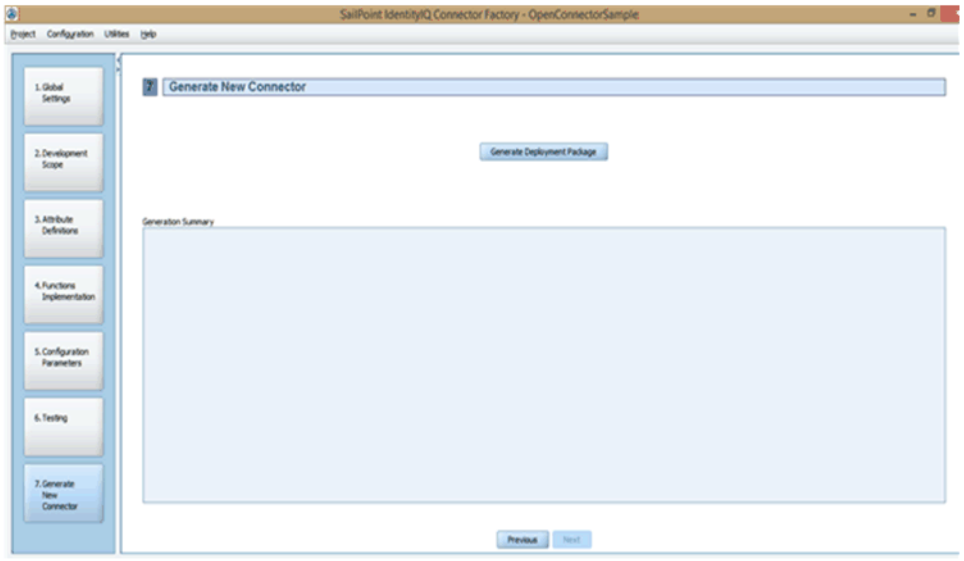


Figure 21—Generate New Connector window

- **Generate New Connector**: The Generate New Connector provides the following types of deployment packages:
  - **Deployment package for IdentityIQ**: We can generate the IdentityIQ Direct Connector package for IdentityIQ deployment by clicking on the Generate Deployment Package.

    A series of messages are displayed, describing the generation process. This would create the following files in the ***connectorFactoryHome*\Generated Install\ *projectName*\ folder** directory, where *projectName* is the name of the project specified during the project creation:

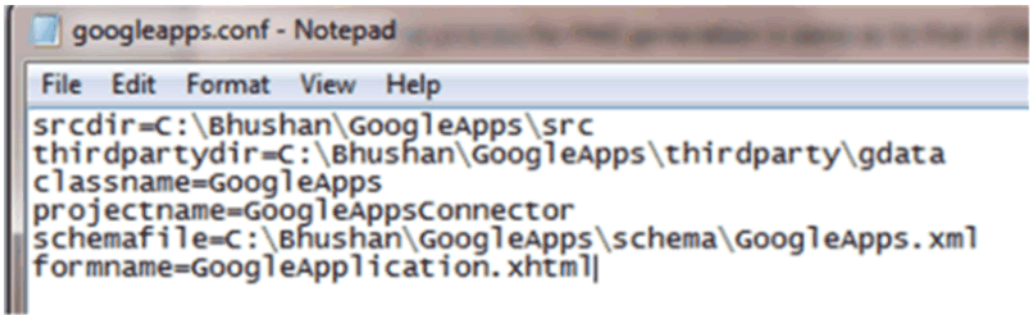
    - **moduleTypeName.jar**
    - **moduleTypeName_AppSchema.xml**
  - Deployment package for IdentityIQ: This option can be used to generate deployment package for Direct Connector and Connector Factory.

Importing an IdentityIQ Direct Connector Interface project

There may arise a scenario where a project implementing IdentityIQ Direct Connector Interface has been developed external to Connector Factory environment. Connector Factory provides an option to import such projects into Studio environment.

Perform the following steps to successfully import an existing project:

1. Create a file (**<projectname>.conf**) containing the following information:
   - **srcdir**: Directory of the project containing source code files.
   - **thirdpartydir**: Directory of the third party jar files referred by the project.
   - **projectname**: Name of the project that would be shown in Connector Factory.
   - **classname**: Name of the class that extends or implements the IdentityIQ Direct Connector interface.
   - **schemafile**: Location of the schema file used by the project.
   - **formname**: Name of the form that would be used in the IdentityIQ schema file.
     Note: Ensure that the variable names are same, else there are chances that the import might fail.



```
srcdir=C:\Bhushan\GoogleApps\src
thirdpartydir=C:\Bhushan\GoogleApps\thirdparty\gdata
classname=GoogleApps
projectname=GoogleAppsConnector
schemafile=C:\Bhushan\GoogleApps\schema\GoogleApps.xml
formname=GoogleApplication.xhtml
```

2. Open the Connector Factory Studio.
3. From the Connector Factory menu bar, select **Utilities => Import OpenConnector Project**.
   The Import Project dialog box (Figure 22— Import Project dialog) is displayed.

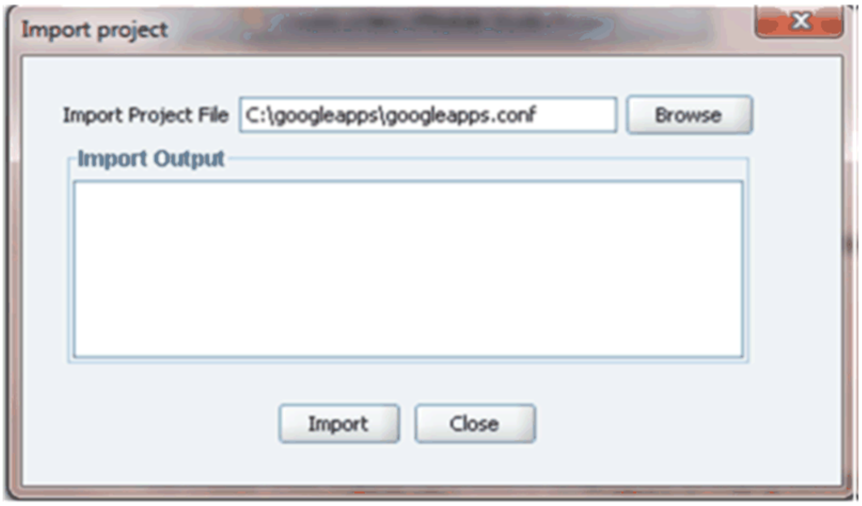


Figure 22—Import Project dialog

4. Browse to the location of the file which contains information about the project to be imported.
   For example, to import GoogleApps project browse to the **c:\GoogleApps\GoogleApps.conf** location.
5. Click **Import**.
   When the import process is finished, the project would be listed in the Connector Factory Studio. If there are any errors during the import, it would be displayed on the Import Project dialog box.

Implementing IdentityIQ Direct Connector Interface Module

The IdentityIQ Direct Connector interface is a generic connector interface that can be used to develop connectors for various managed systems (for example, GoogleApps, SalesForce) The interface can be implemented to retrieve, add, modify, or delete identity objects. User objects can be enabled, disabled, unlocked, change password, and assign group memberships.

IdentityIQ Direct Connector implementations can be used as plug-ins for IdentityIQ implementations using the same JAR file.

IdentityIQ Direct Connector Interface API overview

To create a custom connector using IdentityIQ Direct Connector Interface Module, the implementer must extend the AbstractConnector class which has implemented the Connector interface. In addition to this a set of methods must be overridden to provide the required functionality.

When the custom connector class is created using the Connector Factory, the skeleton class automatically incorporates the empty methods for performing necessary provisioning. The connector developer must incorporate the connector logic across the various methods exposed by the interface. Additional methods may be introduced in the connector class if required.

Following are the object types supported by the IdentityIQ Direct Connector Interface:

- **account**: Used to manage users
- **group**: Used to manage groups

Attribute mappings

The IdentityIQ Direct Connector Interface can manage identity objects such as accounts, groups, and group memberships. Group membership information is captured in account object type. The IdentityIQ Direct Connector Interface being generic in nature, requires attributes of interest to be mapped against the account and group object type. Connector Factory enables mapping these attributes as required by IdentityIQ.

The following describes the attributes in respect to IdentityIQ:

- **The Identity attribute**: Each object type must have an attribute mapped as the identity attribute. This attribute is a unique identifier of the object type and represents the primary key of the object type on the managed system.
- **Entitlement attribute**: To manage group memberships, the user object must incorporate the information of all the groups to which the user is a member. This attribute is in the form of multivalued attribute holding group ids of all the groups to which the user object is a member and is mapped as the entitlement attribute. The attribute may or may not be part of the actual user object on the managed system.

  For instance, an Active Directory user object has the 'member-of' attribute that contains the group membership details of the user object on the managed system. For managed systems whose user object does not incorporate the group membership details, a custom multi-value attribute must be mapped in the Application schema and specified as the entitlement attribute for the user object. When a user object with no group membership details is retrieved, the implementer must also retrieve group membership information of this user and populate this information in the entitlement attribute before the user object is sent to IdentityIQ.

- **Regular attributes**: Any attribute of an identity object may be mapped in the application schema to properly manage the IdentityIQ solutions unless as specified in the “Managing account/group objects” section below.

Managing account/group objects

- **Retrieving account objects**: When an account object is retrieved, the identifier attribute is used to hold the user id and values of the mapped attributes that must be populated in the form of a map and sent to IdentityIQ. Values of only mapped attributes can be seen in Identity IQ. For disabled or locked users, the implementer should properly populate the value of **Connector.ATT_DISABLED** and **Connector.ATT_LOCKED** respectively.
- **Creating/updating a user/group object type**: Whenever a user account is created or updated, a list of attribute-value pairs are sent from Identity IQ. It is the developers responsibility to map the value of a specific attribute to the respective attribute on the managed system.

  For instance, the password would be contained in the **Connector.ATT_PASSWORD** attribute which must be mapped to the corresponding managed system user password attribute in the managed system.

  Note: You can also have the name of the mapped attribute same as that of the name of the attribute in the managed system.
- **Disabling or Enabling user object**: To disable or enable a managed system user or to view its status, the implementer must have a boolean value for the **Connector.ATT_DISABLED** attribute. This attribute is a place holder attribute that enables IdentityIQ to determine that the user object is in enabled or disabled state. If the **Connector.ATT_DISABLED** attribute is not populated with proper value, then the IdentityIQ cannot determine the enabled/disabled state of the user object. If you additionally map the managed system attribute (holding enabled/disabled value) of the user object, then it would duplicate the value of the enabled/disabled state through the following methods:
  - once through the **Connector.ATT_DISABLED** attribute
  - once through the value of the mapped attribute itself
- **Managing passwords**: IdentityIQ refers to the **Connector.ATT_Password** attribute that holds the value of the user password. The implementer must map the user password attribute on the managed system to the **Connector.ATT_Password** attribute. There is no need to additionally map the user password attribute.
- **Unlocking users**: IdentityIQ refers to the **Connector.ATT_LOCKED** attribute that holds the locked status of the user. This attribute takes Boolean values indicating whether the user is in the locked state or not. The implementer must set an appropriate value to indicate the locked state of the user.

Summary of IdentityIQ Direct Connector API

This section provides a brief description of the APIs. The custom connector class is named as SampleConnector.

GET/SET Transactions: All transactions performed from IdentityIQ result in invoking a specific method of the SampleConnector object. The transaction could be one of the following type:

- **GET**: retrieving one or more objects from the managed system
- **SET**: creating a new object, updating, or deleting an already existing object on the managed system

This results in an invocation of a specific method of the SampleConnector. This corresponds to the following methods for the respective type of transactions:

- **GET**: read() and iterate() methods
- **SET**: Create(), Update(), Delete(), Disable(), Enable(), or SetPassword()

All these methods are implicitly invoked by either IdentityIQ. Accordingly, the connector developer must write the necessary logic complying to the action specified by the method.

Guidelines for developing SampleConnector from Connector Factory

- **SampleConnector object instantiation**: Each transaction invoked by IdentityIQ is in context of a instance of SampleConnector object. The object instantiation is the responsibility of IdentityIQ. Once the transaction is completed, the implementer must take care of freeing resources related to the transaction irrespective of whether the call is from a single SampleConnector object for all transactions or single SampleConnector object per transaction.
- **Session handling**: Connection resources opened to perform a transaction should be closed after the transaction is over. Alternatively, implementor could keep connection object alive for multiple transactions. In this case, implementor must take care whether the connection is in a valid state. If not, recreate the connection object.

  - **Returning objects**: In GET type of transactions, if the object to be retrieved exists on the managed system, implementer must return a map of the object. However, if the object or set of objects to be retrieved do not exist on the managed system, the implementer must return null. For any other managed system related failures during a GET, should result in throwing an exception.

  - **Returning Result**: For the SET type of transactions (Create, Update, or Delete methods), it is the developers responsibility to return an openconnector interface result as follows:

    - **object populated with the status of the transaction**

    - **in case of a failed transaction the result must include a failure message**

    If the Result object is not populated, IdentityIQ cannot identify the status of the transaction.
- **XSA keyword mappings**: The keywords have been implicitly mapped with IdentityIQ Direct Connector Interface keywords. Following table depicts the mapping:

XSA Keyword	IdentityIQ Direct Connector Interface mapping
XSA_ACCOUNT_NAME	Identity attribute mapped for account object type. For creating Identity attribute, see Attribute mappings on page 52.
XSA_GROUP_NAME	Identity attribute mapped for group object type. For creating Identity attribute, see Attribute mappings on page 52.
XSA_PASSWORD	Connector.ATT_PASSWORD
XSA_REVOKE_STATUS	Connector.ATT_REVOKED
XSA_LOCK_STATUS	Connector.ATT_LOCKED

Chapter 8: IdentityIQ Direct Connector sample project

This chapter presents the following topics

You can use IdentityIQ Direct Connector Sample Project to learn the functionality related to IdentityIQ Direct Connector project option of Connector Factory. Each project in IdentityIQ Connector Factory has all the definitions needed to run it in the testing step or to generate the project, create the JAR file, schema XML file and deploy it in the IdentityIQ environment.

The **IdentityIQ Direct ConnectorSample** project can work in Windows and UNIX® environments.

Note: In this chapter, OpenConnectorSample project is referred as IdentityIQ Direct Connector Sample project.

Database environment setup

The project requires any standard database with an user with privileges who can run SELECT, CREATE, or DELETE queries on the tables.

This project comes bundled with a sample SQL file which needs to be imported on the MySQL server.

Note: User can implement any kind of database. An example of SQL schema file is shipped with the IdentityIQ Direct Connector project folder.

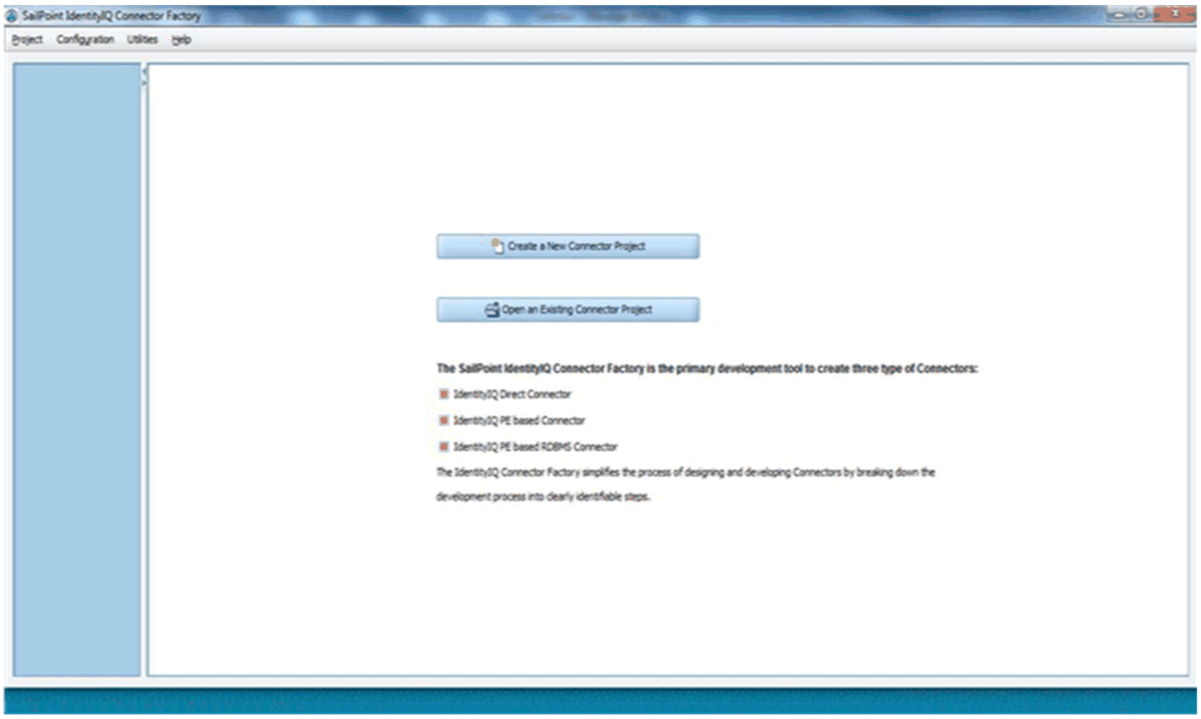
To implement the database, perform the following:

1. The **create_openconnsample.sql** file which should be imported in SQL environment is located at ***connectorFactoryHome*\projects\IdentityIQ Direct ConnectorSample\conf\** directory.
2. Import the **create_openconnsample.sql** file and execute it.
   The database called **openconntest** with the userids and its respective fields are displayed.

Generation of jars, xml, and testing the sample project

1. You can access the Connector Factory using one of the following options:
   - Click **Start** => **All Programs** => **IdentityIQ Connector Factory** => **IIQConnectorFactory.bat**.
   - In Windows Explorer, navigate to the directory where Connector Factory is installed, and double-click the **IIQConnectorFactory.bat** file.
   - Click the Connector Factory desktop shortcut Icon.

   The following window is displayed:

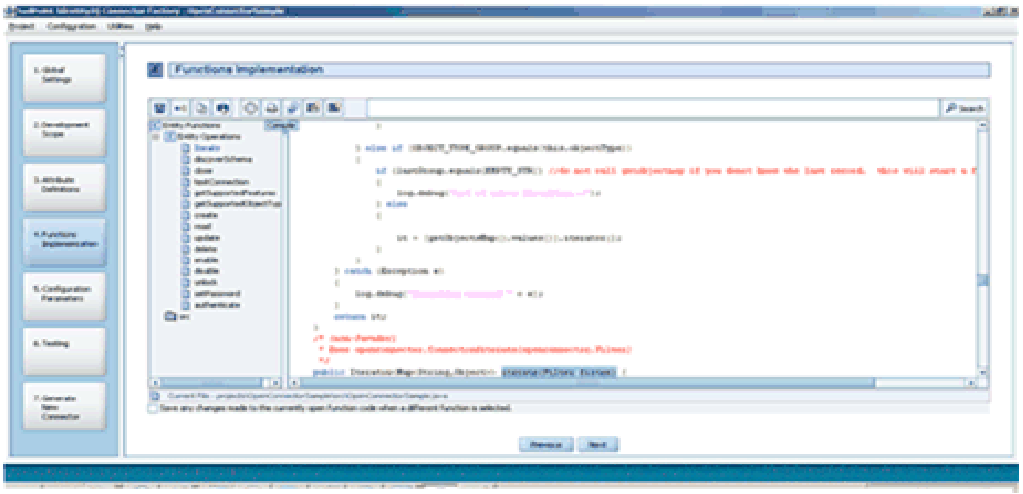

SailPoint IdentityIQ Connector Factory
Project Configuration Utilities Help
Create a New Connector Project
Open an Existing Connector Project

The SailPoint IdentityIQ Connector Factory is the primary development tool to create three type of Connectors:

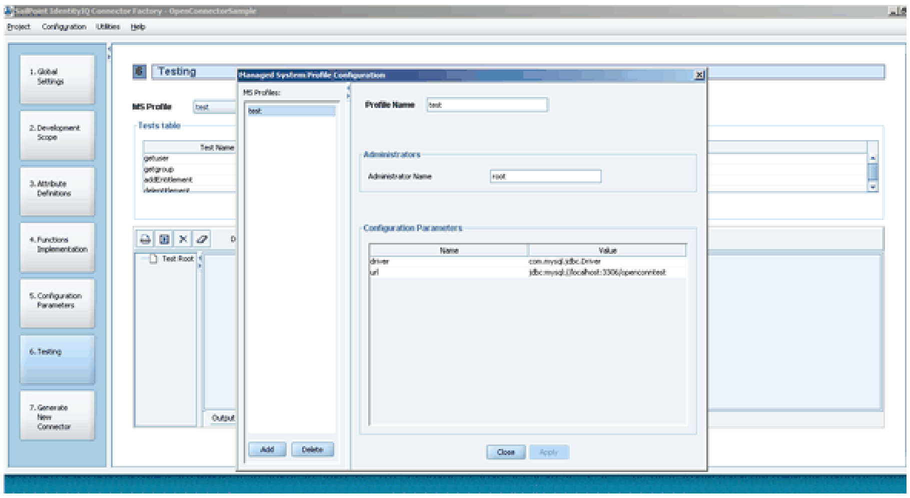
- IdentityIQ Direct Connector
- IdentityIQ PE based Connector
- IdentityIQ PE based RDBMS Connector

The IdentityIQ Connector Factory simplifies the process of designing and developing Connectors by breaking down the development process into clearly identifiable steps.

2. Click **Open an Existing Connector Project** button.
   The existing Project List is displayed.
3. Select the **IdentityIQ Direct ConnectorSample** project and click open Project.
   The Global setting attributes step is displayed.
4. Compilation and the code changes as well as the implementation of function is done at STEP # 4 "Functions Implementation" as shown below:



5. Test Framework provide test functionality for **getuser**, **getgroup**, **addEntitlement** and **delEntitiement**.
   Set the url and database name by clicking "Manage MS Profile".
6. To check the test result select any function from **getuser**, **getgroup**, **addEntitlement** and **deletEntitiement.**


SailPoint IdentityIQ Connector Factory - OpenConnectorSample
Project
Configuration
Utilities
Help
1. Global Settings
2. Development Scope
3. Attribute Definitions
4. Functions Implementation
5. Configuration Parameters
6. Testing
7. Generate New Connector
Testing
Managed System Profile Configuration
MS Profiles:
Add
Delete
Close
Apply

MS Profile: test

Tests table

Test Name
getuser
getgroup
addEntitlement
deleteEntitlement

Profile Name: test

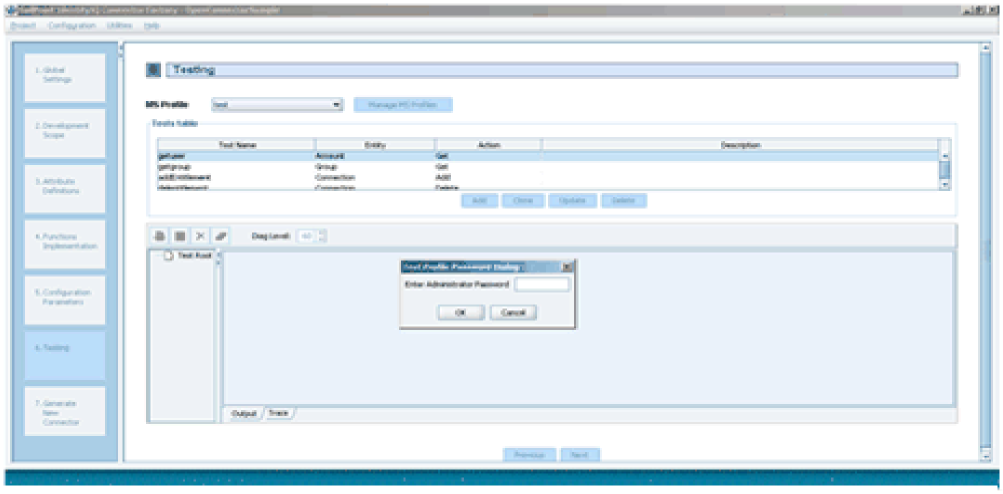
Administrators

Administrator Name: root

Configuration Parameters

Name	Value
driver	com.mysql.jdbc.Driver
url	jdbc:mysql://localhost:3306/openconntest

7. The Test Profile Password Dialog is displayed in the following figure:

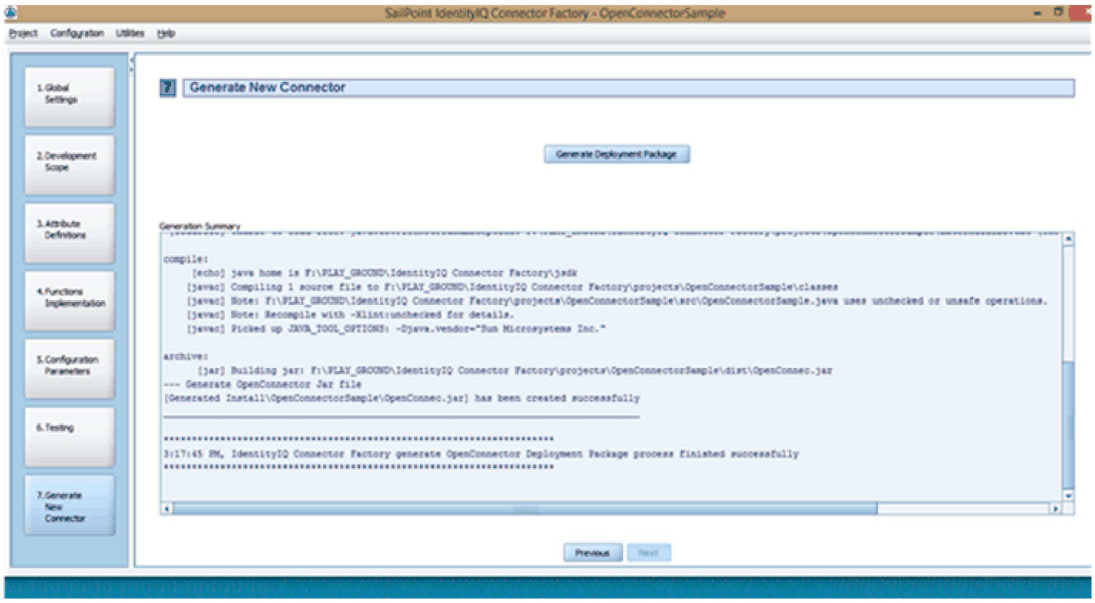


MS Profile: test

Tests table

Test Name	Entity	Action	Description
getuser	Account	Get	
getgroup	Group	Get	
addEntitlement	Connection	Add	
deleteEntitlement	Connection	Delete	

8. Click on STEP # 7 “Generate New Connector”.


SailPoint IdentityIQ Connector Factory - OpenConnectorSample
Project Configuration Utilities Help
1.Global Settings
2.Development Scope
3.Attribute Definitions
4.Functions Implementation
5.Configuration Parameters
6.Testing
7.Generate New Connector
7 Generate New Connector
Generate Deployment Package
Generation Summary
Previous
Next

```
compile:
     [echo] java home is F:\PLAY_GROUND\IdentityIQ Connector Factory\jsdk
    [javac] Compiling 1 source file to F:\PLAY_GROUND\IdentityIQ Connector Factory\projects\OpenConnectorSample\classes
    [javac] Note: F:\PLAY_GROUND\IdentityIQ Connector Factory\projects\OpenConnectorSample\src\OpenConnectorSample.java uses unchecked or unsafe operations.
    [javac] Note: Recompile with -Xlint:unchecked for details.
    [javac] Picked up JAVA_TOOL_OPTIONS: -Djava.vendor="Sun Microsystems Inc."

archive:
      [jar] Building jar: F:\PLAY_GROUND\IdentityIQ Connector Factory\projects\OpenConnectorSample\dist\OpenConnec.jar
--- Generate OpenConnector Jar file
[Generated Install\OpenConnectorSample\OpenConnec.jar] has been created successfully

***********************************************************************
3:17:45 PM, IdentityIQ Connector Factory generate OpenConnector Deployment Package process finished successfully
***********************************************************************
```

9. Click “Generate Deployment Package”.

Note: The Deployment Package will be generated in the Install directory.

Implementation of jars and xml file in IdentityIQ environment

1. After successful generation of connector Deployment Package. IdentityIQ Connector Studio will create the **Generated Install** folder in the *connectorFactoryHome* directory.
2. Browse to the following location to find the **Generated Install** folder:
   ***connectorFactoryHome*\Generated Install\IdentityIQ Direct ConnectorSample**

   The **Generated Install** folder consist of the following files:

   - OpenConnec.jar
   - OpenConnec_AppSchema.xml

   Copy the **OpenConnec_AppSchema.xml** file at **WEB-INF/config/connector/** directory.

   Create an xml file by copying the following contents and import it from IdentityIQ Console. Name the xml file as **sample.xml** and save it in a temporary directory (for example, D:/temp).

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE sailpoint PUBLIC "sailpoint.dtd" "sailpoint.dtd">
<sailpoint>
 <ImportAction name="mergeConnectorRegistry">
  <Configuration>
   <Attributes>
    <Map>
     <entry key='fileList'>
      <value>
       <List>
        <String>WEB-INF/config/connector/OpenConnec_AppSchema.xml</String>
```

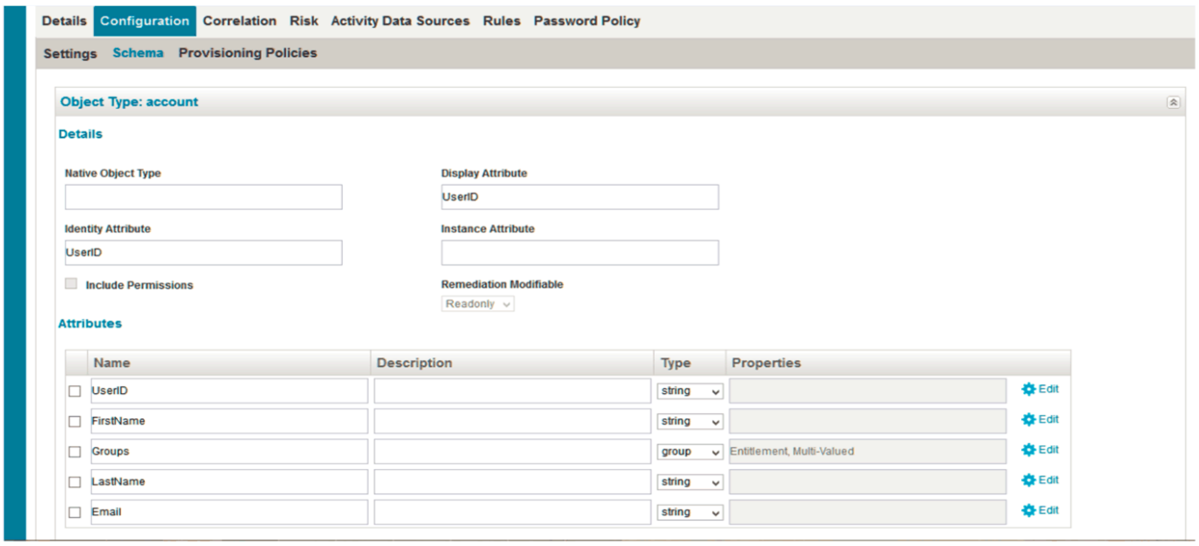
```
            </List>
           </value>
          </entry>
         </Map>
        </Attributes>
       </Configuration>
      </ImportAction>
     </sailpoint>
```

Note: The xml format of previous existing connectors which were developed in Connector Factory version 6.0 must be replaced with the standards according to the provided sample application "OpenConnec_AppSchema.xml" schema xml file.

Note: Similarly the xhtml format of application.xhtml must be according to the standards provided in "OpenConnectorSample.xhtml" file of sample project.

For more information on developing the compatible application for IdentityIQ version 6.4, see Appendix , "Developing the compatible application for IdentityIQ".

3. Navigate to apache tomcat folder where the IdentityIQ folder structure is present.
4. Navigate to the **WEB-INF\bin** folder as follows:
   C:\Program Files (x86)\apache\apache\webapps\identityiq\WEB-INF\lib
5. Import the **sample.xml** file from the temporary directory (D:/temp) using IdentityIQ Console as follows:
   C:\Program Files (x86)\apache\apache\webapps\identityiq\WEB-INF\lib> iiq console
   >import D:/temp/sample.xml
6. Copy **OpenConnectorSample.xhtml** from ***connectorFactoryInstallDir*\projects\OpenConnectorSample\conf** to ***identityIQInstallDir*\define\applications**.
7. Open the IdentityIQ browser and click on application. Under the define tab, click on new application.
8. Enter all the required details according to application name.
9. The **OpenConnec** option will be automatically generated in Application type. Select the Application type as **OpenConnec**.
10. Save the application.
    If the application is opened again the schema of that application for account and group is displayed as follows:



Object Type: account

Details

Native Object Type:

Display Attribute: UserID

Identity Attribute: UserID

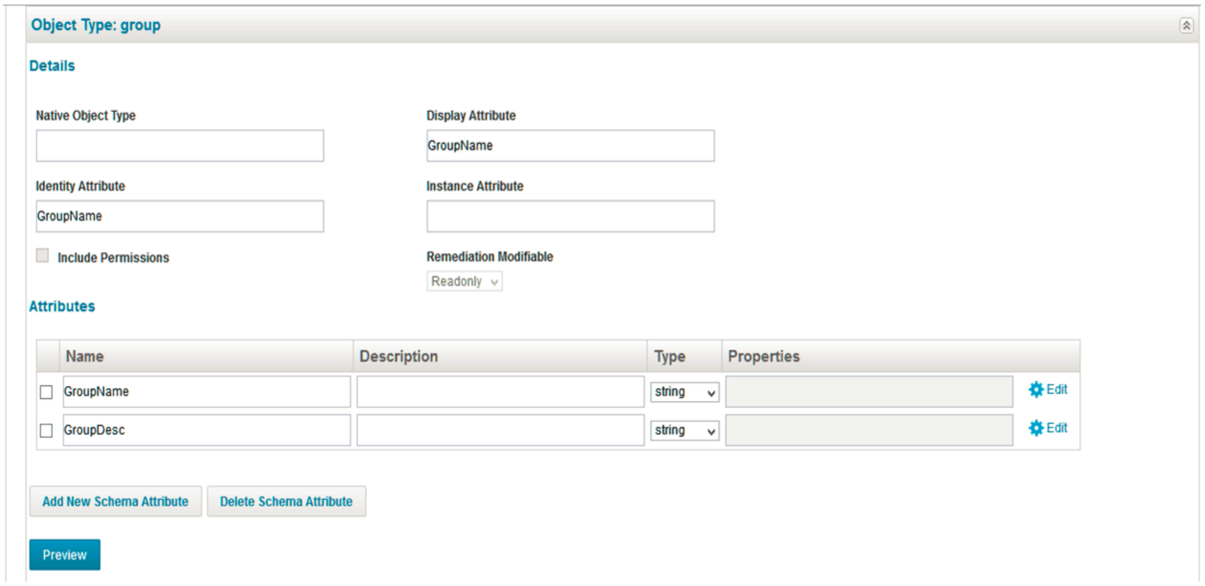
Instance Attribute:

Include Permissions

Remediation Modifiable: Readonly

Attributes

Name	Description	Type	Properties
UserID		string	
FirstName		string	
Groups		group	Entitlement, Multi-Valued
LastName		string	
Email		string	


Object Type: group

Details

Native Object Type	Display Attribute
	GroupName
Identity Attribute	Instance Attribute
GroupName	
☐ Include Permissions	Remediation Modifiable: Readonly

Attributes

	Name	Description	Type	Properties	
☐	GroupName		string		Edit
☐	GroupDesc		string		Edit

11. Start **Account Aggregation** and **Account Group Aggregation**.
    All the identities which are in the created database on MySQL server will be displayed by clicking **Group** option under **define** tab. The groups can been seen by clicking **identity** option under **define** tab.

Appendix A: Troubleshooting

This appendix describes the problems that may be encountered while implementing the Connector Factory and the resolutions for these problems.

1 - Issue with the integrated test setup of connector factory for provisioning of groups.

The following error message is displayed if the user encounters issue with the integrated test setup of connector factory for provisioning of groups:

```
Transaction failed at ConnectorInit with RC=ESA_FATAL
```

Resolution: Implement the logic in the connector's skeleton code and test it through the IdentityIQ console or UI by creating the group provisioning policy.

Glossary

D

Deployment Package (Installation Package)

Set of files generated by SailPoint IdentityIQ Connector Factory. These files are used to install your customized SailPoint IdentityIQ Connector Factory on the computer where your Managed System is located and to import the new Managed System type into IdentityIQ.

E

IdentityIQ (IIQ):

The central management component of SailPoint, SailPoint's solution to the problem of enterprise security administration.

G

Guided Development Environment (GDE)

Part of the Connector Factory user interface. Enables developing the functionality defined in the New Project wizard and generating an a deployment package for the Connector.

S

Service Provisioning Markup Language (SPML

An XML-based framework for exchanging user, resource, and service provisioning information between organizations. The framework is expected to establish an open, standard protocol for the integration and inter-operability of service provisioning requests.

X

IdentityIQ Connector Factory

User interface that includes guided tasks for developing the new Connectors.

Index