Oracle® Identity Governance

Configuring the Database Application Tables Application

12c (12.2.1.3.0)
F13723-08
June 2022

ORACLE

Oracle Identity Governance Configuring the Database Application Tables Application, 12c (12.2.1.3.0)

F13723-08

Primary Author: Gowri.G.R

Contributors: Bhargav Janapati, Swati Negi

Contents

List of Figures

List of Tables

Preface

This guide describes the connector that is used to integrate Oracle Identity Governance with database tables that store user data.

Audience

This guide is intended for resource administrators and target system integration teams.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Related Documents

For information about installing and using Oracle Identity Governance 12.2.1.3.0, visit the following Oracle Help Center page:

https://docs.oracle.com/en/middleware/idm/identity-governance/12.2.1.3/index.html

For information about installing and using Oracle Identity Manager 11.1.2.3, visit the following Oracle Help Center page:

http://docs.oracle.com/cd/E52734_01/index.html

For information about Oracle Identity Governance Connectors 12.2.1.3.0 documentation, visit the following Oracle Help Center page:

https://docs.oracle.com/en/middleware/idm/identity-governance-connectors/12.2.1.3/index.html

For information about Oracle Identity Manager Connectors 11.1.1 documentation, visit the following Oracle Help Center page:

http://docs.oracle.com/cd/E22999_01/index.htm

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

What's New in This Guide

These are the updates made to the software and documentation for release 12.2.1.3.0 of the Database Application Tables connector.

The updates provided in this chapter are divided into the following categories:

- Software Updates

  This section describes updates made to the connector software.

- Documentation-Specific Updates

  This section describes major changes made to the connector documentation. These changes are not related to software updates.

Software Updates

These are the updates made to the connector software.

Software Updates in Release 12.2.1.3.0

The following is the software update in release 12.2.1.3.0:

Support for Onboarding Applications Using the Connector

From this release onward, the connector bundle includes application onboarding templates required for performing connector operations on the Oracle Database, MySQL, Microsoft SQL, and IBM DB2 targets. This helps in quicker onboarding of the applications for these targets into Oracle Identity Governance by using an intuitive UI.

Documentation-Specific Updates

These are the updates made to the connector documentation.

Documentation-Specific Updates in Release 12.2.1.3.0

The following documentation-specific update has been made in revision "07" of this guide:

The "Target systems" row of Table 1-1 has been updated to include support for Oracle Database 19c.

The following documentation-specific update has been made in revision "06" of this guide:

The "Target systems" row of Table 1-1 has been updated to include Microsoft SQL Server 2016.

The following documentation-specific update has been made in revision "05" of this guide:

The "Target systems" row of Table 1-1 has been updated.

The following documentation-specific update has been made in revision "04" of this guide:

The jdbcUrlTemplate parameter of Basic Configuration Parameters has been updated to include new sample values for Oracle Database with SID and Oracle Database with Service Name.

The following documentation-specific updates have been made in revision "03" of this guide:

- The "Oracle Identity Governance or Oracle Identity Manager" row of Table 1-1 has been updated to include support for Oracle Identity Governance 12*c* (12.2.1.4.0).
- The "Target systems" row of Table 1-1 has been updated to include support for SAP HANA DB version 2.0 SP 01 or SP 04.
- The "ngdbc" row has been added to Table 1-1.
- Usage Recommendation has been updated to include information about using Oracle Identity Governance versions 12.2.1.3.A and 12.2.1.3.0 of the connector.
- Configuring the Datasource and JNDI Properties for SAP HANA DB has been added.
- Step 1.e of Creating an Application By Using the Connector has been updated to include information about adding attributes and providing mappings for Oracle Identity Governance 12*c* versions 12.2.1.3.0 and 12.2.1.4.0.
- Entries for parameters jdbcDriver and jdbcUrlTemplate have been added to Basic Configuration Parameters.
- Entry for parameter sapHanaDb has been added to Advanced Settings Parameters.
- Information regarding SAP HANA DB has been added to a "Note" in Creating an Application By Using the Connector.

The following documentation-specific updates have been made in revision "02" of this guide:

- The "Target systems" row of Table 1-1 has been updated to include Microsoft SQL Server 2017.
- Configuring the Datasource and JNDI Properties has been updated to include information about performing the procedure only when the connector uses datasource configuration to connect to the target system.

The following documentation-specific update has been made in revision "01" of this guide:

This is the first release of the Oracle Identity Governance Connector for Database Application Tables. Therefore, there are no documentation-specific updates in this release.

1

About the Connector

This chapter introduces the Database Application Tables connector.
This chapter discusses the following topics:

1.1 Introduction to the Database Application Tables Connector

Oracle Identity Governance (OIG) platform automates access rights management, security, and provisioning of IT resources. Oracle Identity Governance connects users to resources, and revokes and restricts unauthorized access to protect sensitive corporate information. Oracle Identity Governance connectors are used to integrate Oracle Identity Governance with external and identity-aware applications such as PeopleSoft and MySQL.

In an enterprise setup, many applications in your organization may use relational database tables as a repository for user data. This guide describes the procedure to dynamically generate the connector based on the underlying schema of the database table user store, and to install and use this connector for managing user lifecycle and entitlements from Oracle Identity Governance. After you integrate the tables with Oracle Identity Governance by using the connector, you can use them either as a managed (target) resource or as an authoritative (trusted) source of user data for Oracle Identity Governance.

The connector that you generate is known as a **Database Application Tables connector** (DBAT connector). The following sample scenario describes the requirement that can be addressed by a DBAT connector:

Example Inc. has some database-driven custom applications. These applications do not have any APIs for identity administration. The company wants to manage the lifecycle of users in these custom applications by using a centralized identity management system such as OIM.

The DBAT connector is one of the solutions to this business problem. Example Inc. can use this connector to enable the exchange of user data between the database and Oracle Identity Governance.

From Oracle Identity Governance release 12.2.1.3.0 onward, connector deployment can also be handled using the application onboarding capability of Oracle Identity Self Service. This capability lets business users to onboard applications with minimum details and effort. The connector installation package includes a collection of predefined templates (XML files) that contain all the information required for provisioning and reconciling data from a given

application or target system. These templates also include basic connectivity and configuration details specific to your target system. The connector uses information from these predefined templates allowing you to onboard your applications quickly and easily using only a single and simplified UI.

Application onboarding is the process of registering or associating an application with Oracle Identity Governance and making that application available for provisioning and reconciliation of user information.

> **Note:**
>
> In this release, the DBAT connector can be deployed either by using application onboarding or the Connector Installer. In this guide, the connector that is deployed using the Applications option on the Manage tab of Identity Self Service is referred to as an **AOB application**. The connector that is deployed using the Manage Connector option in Oracle Identity System Administration is referred to as a **CI-based connector** (Connector Installer-based connector).

> **Note:**
>
> In this guide:
>
> - The database tables and their relation tables that store user data are collectively referred to as the **target system.**
> - The computer on which the database is installed is referred to as the **target system host computer.**
> - *RELEASE_NUMBER* has been used as a placeholder for the current release number of the connector. Therefore, replace all instances of *RELEASE_NUMBER* with the release number of the connector. For example, 12.2.1.3.0.

1.2 Understanding Target System Discovery in the DBAT Connector

Target systems are identity-aware applications such as databases, Microsoft Active Directory, Siebel and so on that can be managed by Oracle Identity Governance connectors.

In general, there are two broad categories of target systems for which Oracle Identity Governance connectors exist:

- **Predefined target systems:** These are target systems that have a static schema and the connector is aware of this schema. This means that connectors for such target systems are shipped with preconfigured metadata or connector artifacts such as IT resource definition, process forms, resource objects, and so on.
- **Discovered target systems:** These are target systems for which the schema is not known in advance. For example, a flat file does not have a fixed schema. Each target system can have a totally different schema. The connector is not initially

aware of the schema that it is supposed to integrate with and the attributes available.

The DBAT connector is a connector for a discovered target system.

Connectors for discovered target systems are not shipped with any artifacts. They are shipped only with a set of deployment utilities that help in discovering the schema and then generating the artifacts.

Discovery is the process of identifying the underlying schema of your database. You can discover the schema of your database by configuring a groovy file and running the DBAT Generator. This is discussed later in the guide.

1.3 Certified Components

Table 1-1 lists the certified components for this connector.

Table 1-1 Certified Components

Component	Requirement for AOB Application	Requirement for CI-Based Connector
Oracle Identity Governance or Oracle Identity Manager	You can use one of the following releases of Oracle Identity Governance: • Oracle Identity Governance 12*c* (12.2.1.4.0) • Oracle Identity Governance 12*c* (12.2.1.3.0)	You can use one of the following releases of Oracle Identity Governance or Oracle Identity Manager: • Oracle Identity Governance 12*c* (12.2.1.4.0) • Oracle Identity Governance 12*c* (12.2.1.3.0) • Oracle Identity Manager 11*g* Release 2 PS3 (11.1.2.3.0)
Target systems	The target system can be database tables from any one of the following RDBMSs: • IBM DB2 Version 11.*x* • Microsoft SQL Server 2016, 2017 • MySQL 5.*x* • Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 • Oracle Database 19c or 18c or 12c as a single database, pluggable database (PDB), or Oracle RAC implementation * Oracle Database 10g and 11g as either a single database or Oracle RAC implementation • SAP HANA DB version 2.0 SP 01 or SP 04	The target system can be database tables from any one of the following RDBMSs: • IBM DB2 Version 11.*x* • Microsoft SQL Server 2016, 2017 • MySQL 5.*x* • Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 • Oracle Database 19c or 18c or 12c as a single database, pluggable database (PDB), or Oracle RAC implementation * Oracle Database 10g and 11g as either a single database or Oracle RAC implementation • SAP HANA DB version 2.0 SP 01 or SP 04

Table 1-1 (Cont.) Certified Components

Component	Requirement for AOB Application	Requirement for CI-Based Connector
JDBC drivers	Depending on the target system that you use, download one of the following sets of JDBC drivers from the Vendor's Web site: **For IBM DB2:** • For all platforms: db2jcc • For IBM DB2 with the autoincrement option set on the primary key column: db2jcc4 **For Microsoft SQL Server:** • For Microsoft SQL Server 2014: sqljdbc4 version 4.0 **For MySQL:** mysql-connector-java-5.1.12-bin **For Oracle Database or Oracle RAC:** • For JDK 1.6: ojdbc6	Depending on the target system that you use, download one of the following sets of JDBC drivers from the Vendor's Web site: **For IBM DB2:** • For all platforms: db2jcc • For IBM DB2 with the autoincrement option set on the primary key column: db2jcc4 **For Microsoft SQL Server:** • For Microsoft SQL Server 2014: sqljdbc4 version 4.0 **For MySQL:** mysql-connector-java-5.1.12-bin **For Oracle Database or Oracle RAC:** • For JDK 1.6: ojdbc6
ngdbc	**For SAP HANA DB:** Download SAP HANA Database JDBC Driver jar, for example, `ngdbc-2.4.64.jar` from SAP Development tools for SAP HANA	**For SAP HANA DB:** Download SAP HANA Database JDBC Driver jar, for example, `ngdbc-2.4.64.jar` from SAP Development tools for SAP HANA
Connector Server	11.1.2.1.0 or later	11.1.2.1.0 or later
Connector Server JDK	JDK 1.6 or later	JDK 1.6 or later
Format in which user data is stored in the target system	You can use a Database Application Tables connector only if user data is stored in the target system in any one of the following formats: • All user data is in a single table or view. • User data is spread across one parent table and one or more child tables. This target system can be configured only as a target resource, and not as a trusted source. • All user data is in a single updatable view (that is based on one or more tables). • User data is spread across one updatable view (that is based on one or more tables) and one or more child views (that are based on one or more tables). This type of target system can be configured only as a target resource, and not as a trusted source with this connector. In other words, a trusted source cannot store child data.	You can use a Database Application Tables connector only if user data is stored in the target system in any one of the following formats: • All user data is in a single table or view. • User data is spread across one parent table and one or more child tables. This target system can be configured only as a target resource, and not as a trusted source. • All user data is in a single updatable view (that is based on one or more tables). • User data is spread across one updatable view (that is based on one or more tables) and one or more child views (that are based on one or more tables). This type of target system can be configured only as a target resource, and not as a trusted source with this connector. In other words, a trusted source cannot store child data.

Table 1-1 (Cont.) Certified Components

Component	Requirement for AOB Application	Requirement for CI-Based Connector
Other requirements of the target system	The target system must meet the following requirement: If parent and child tables are not joined by a foreign key (for example, if you are using views), then the names of the foreign key columns in both tables must be the same.	The target system must meet the following requirement: If parent and child tables are not joined by a foreign key (for example, if you are using views), then the names of the foreign key columns in both tables must be the same.

1.4 Usage Recommendation

These are the recommendations for the Database Application Tables connector version that you can deploy and use depending on the Oracle Identity Governance or Oracle Identity Manager version that you are using.

- If you are using Oracle Identity Governance release 12*c* (12.2.1.4.0) or 12*c* (12.2.1.3.0) or later, SAP HANA DB version 2.0 SP 01 or SP 02 or SP 03, then use the 12.2.1.3.A (p30197332_122130_Generic.zip) version of this connector.
- If you are using Oracle Identity Governance release 12*c* (12.2.1.4.0) or 12*c* (12.2.1.3.0) or later, SAP HANA DB version 2.0 SP 04 , then use the 12.2.1.3.0 version of this connector.
- If you are using Oracle Identity Governance release 12*c* (12.2.1.3.0) or later, then use the latest 12.2.1.x version of this connector. Deploy the connector using the Applications option on the Manage tab of Identity Self Service.
- If you are using any of the Oracle Identity Manager releases listed in the 'Requirement for CI-Based Connector' column in Table 1-1, then use the latest 11.1.x version of this connector.
- If you want to use the 12.1.x version of this connector, then you can install and use it only in the CI-based mode. If you want to use the AOB application, then you must upgrade to Oracle Identity Governance release 12c (12.2.1.3.0) or later.

1.5 Certified Languages

The connector will support the languages that are supported by Oracle Identity Manager. Resource bundles are not part of the connector installation media as the resource bundle entries vary depending on the target system being used.

1.6 Supported Data Types

The data types supported for reconciliation and provisioning operations are listed in the following section:

> **Note:**
>
> Complex data types, such as RAW, Binary File, CLOB, and BLOB, are not supported. Any data type that is not supported *and* is not a complex data type is treated as a String data type.

For IBM DB2 Database:

- SMALLINT
- BIGINT
- INTEGER
- REAL
- FLOAT
- DOUBLE
- DECIMAL
- CHARACTER
- VARCHAR
- DATE
- TIMESTAMP

For Microsoft SQL Server:

- CHAR
- VARCHAR
- SMALLINT
- INT
- BIGINT
- DECIMAL
- NUMERIC
- NVARCHAR
- FLOAT
- REAL
- SMALLDATETIME
- DATETIME

For MySQL:

- BOOL
- SMALLINT
- MEDIUMINT
- INT
- BIGINT

- FLOAT
- DOUBLE
- DECIMAL
- CHAR
- VARCHAR
- TINYTEXT
- DATE
- DATETIME
- TIMESTAMP

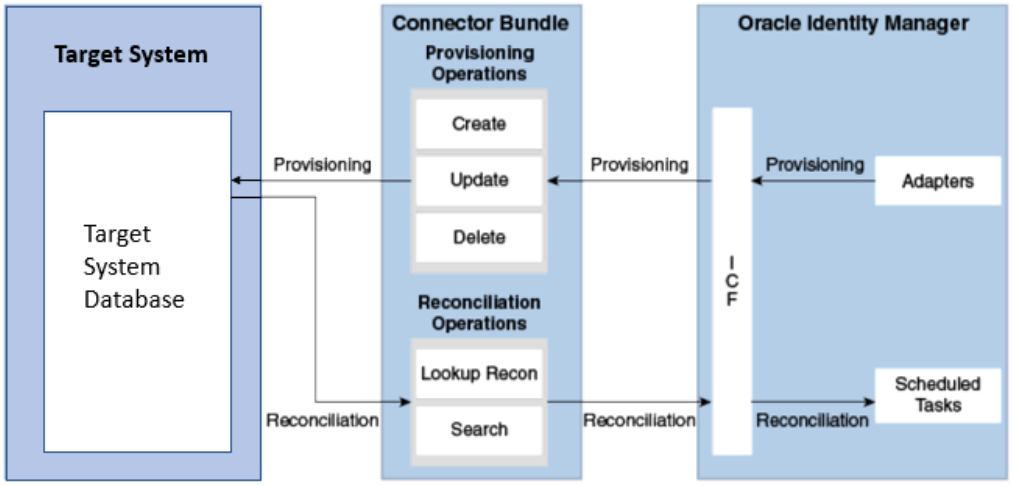
For Oracle Database:

- VARCHAR2
- CHAR
- NUMBER
- NUMERIC
- INTEGER
- INT
- SMALLINT
- DOUBLE
- FLOAT
- DECIMAL
- DEC
- REAL
- DATE
- TIMESTAMP

1.7 Connector Architecture

Figure 1-1 shows the architecture of the connector.

Figure 1-1 Connector Architecture



The Database Application Tables connector is implemented by using the Identity Connector Framework (ICF). The ICF is a component that provides basic reconciliation and provisioning operations that are common to all Oracle Identity Governance connectors. In addition, ICF provides common features that developers would otherwise need to implement on their own, such as connection pooling, buffering, time outs, and filtering. The ICF is shipped along with Oracle Identity Governance.

The DBAT connector can be configured to run in one of the following modes:

- Identity reconciliation

  In the identity reconciliation mode, the target system is used as the trusted source and users are directly created and modified on it directly outside Oracle Identity Governance.

  During reconciliation, a scheduled job establishes a connection with the target system and sends reconciliation criteria to the APIs. The APIs extract user records that match the reconciliation criteria and hand them over to the scheduled task, which brings the records to Oracle Identity Governance. The next step depends on the mode of connector configuration.

  Each record fetched from the target system is compared with existing OIM Users. If a match is found, then the update made to the record on the target system is copied to the OIM User attributes. If no match is found, then the target system record is used to create an OIM User.

  > **Note:**
  >
  > Trusted reconciliation does not support multivalued attributes, for example, child table entries.

- Account Management

  In the account management mode, the target system is used as a target resource. The connector enables the target resource reconciliation and provisioning

operations. Through provisioning operations performed on Oracle Identity Governance, user accounts are created and updated on the target system for OIM Users. During reconciliation from the target resource, the Database Application Tables connector fetches into Oracle Identity Governance data about user accounts that are created or modified on the target system. This data is used to add or modify resources allocated to OIM Users.

During provisioning operations, adapters carry provisioning data submitted through the process form to the target system. APIs on the target system accept provisioning data from the adapters, carry out the required operation on the target system, and return the response from the target system to the adapters. The adapters return the response to Oracle Identity Governance.

During reconciliation, a scheduled task calls the connector bundle which gets the data from target and handles it and returns to OIM and associates it based on Recon rule.

Note:

- It is recommended that you do not configure the target system as both an authoritative (trusted) source and a managed (target) resource.
- See Installing Connectors in *Oracle Fusion Middleware Administering Oracle Identity Governance* for detailed information about connector deployment configurations.

1.8 Supported Connector Features Matrix

Provides the list of features supported by the AOB application and CI-based connector.

Table 1-2 Supported Connector Features Matrix

Feature	AOB Application	CI-Based Connector	Supported Target Systems
Full reconciliation	Yes	Yes	All
Incremental reconciliation	Yes	Yes	All
Limited (filtered) reconciliation	Yes	Yes	All
Both target resource and trusted source reconciliation	Yes	Yes	All
Reconciliation of deleted user records	Yes	Yes	All
Transformation and validation of account data	Yes	Yes	All
Adding user-defined fields for reconciliation and provisioning	Yes	Yes	All

Table 1-2 (Cont.) Supported Connector Features Matrix

Feature	AOB Application	CI-Based Connector	Supported Target Systems
Configuring the connector for stored procedures	Yes	Yes	All

1.9 Features of the Connector

The following are features of the connector:

- Full and Incremental Reconciliation
- Limited (Filtered) Reconciliation
- Support for Both Target Resource and Trusted Source Reconciliation
- Support for Reconciliation of Deleted User Records
- Transformation and Validation of Account Data
- Support for Adding User-Defined Fields for Reconciliation and Provisioning
- Support for Configuring the Connector for Stored Procedures

1.9.1 Full and Incremental Reconciliation

After you create the connector, you can perform full reconciliation to bring all existing user data from the target system to Oracle Identity Governance. After the first full reconciliation run, you can configure your connector for incremental reconciliation. In incremental reconciliation, only records that are added or modified after the last reconciliation run are fetched into Oracle Identity Governance.

See Performing Full Reconciliation and Incremental Reconciliation for more information on full and incremental reconciliation.

1.9.2 Limited (Filtered) Reconciliation

To limit or filter the records that are fetched into Oracle Identity Governance during a reconciliation run, you add conditions in the Filter attribute of the scheduled job or in the customizedQuery parameter of the IT resource.

See Performing Limited Reconciliation for more information.

1.9.3 Support for Both Target Resource and Trusted Source Reconciliation

You can use the connector to configure your target system as either a target resource or trusted source of Oracle Identity Governance.

See Reconciliation Scheduled Jobs for more information.

1.9.4 Support for Reconciliation of Deleted User Records

Apart from the scheduled jobs for user records reconciliation, there are independent scheduled jobs for reconciliation of deleted user records. In target resource mode, if a record is deleted on the target system, then the corresponding Database Application Tables resource is revoked from the OIM User. In trusted source mode, if a record is deleted on the target system, then the corresponding OIM User is deleted.

See Scheduled Jobs for Reconciliation of Deleted Users Records for more information about the scheduled jobs used for reconciling deleted user records.

1.9.5 Transformation and Validation of Account Data

You can configure validation of account data that is brought into or sent from Oracle Identity Governance during reconciliation and provisioning. In addition, you can configure transformation of account data that is brought into Oracle Identity Governance during reconciliation. For more information, see Configuring Transformation and Validation of Data.

1.9.6 Support for Adding User-Defined Fields for Reconciliation and Provisioning

You can create mappings for OIM User fields that are not included in the list of default mappings. These fields can be either a part of the standard set of OIM User fields provided on the target system or user-defined fields that you add to Oracle Identity Governance.

1.9.7 Support for Configuring the Connector for Stored Procedures

The connector runs default SQL queries and statements when you use it to perform reconciliation and provisioning operations. The connector supports calling custom stored procedures to perform connector operations. Instead of these default SQL queries and statements, you can configure the connector to call a script written in the Groovy scripting language, which runs the custom stored procedures.

See Configuring the Connector for Stored Procedures and Groovy Scripts for more information.

2

Creating an Application By Using the Database Application Tables Connector

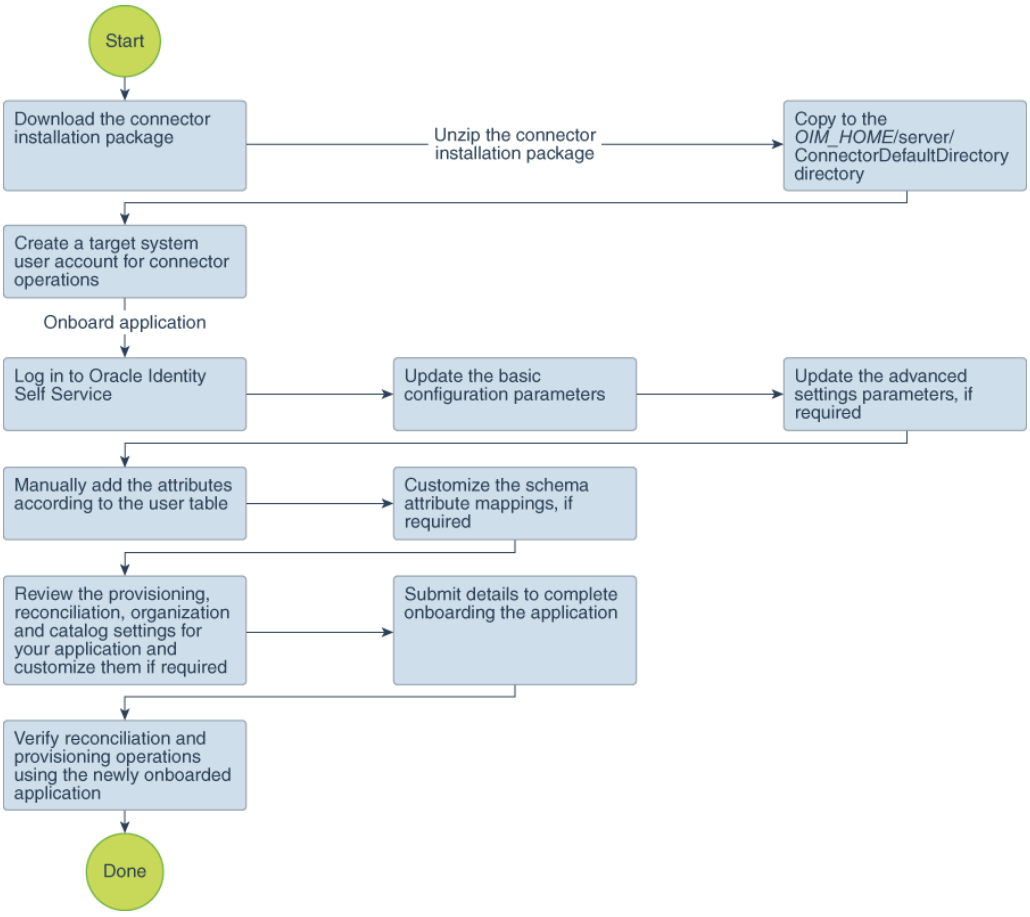
Learn about onboarding applications using the connector and the prerequisites for doing so.

2.1 Process Flow for Creating an Application By Using the Connector

From Oracle Identity Governance release 12.2.1.3.0 onward, connector deployment is handled using the application onboarding capability of Identity Self Service.

Figure 2-1 is a flowchart depicting high-level steps for creating an application in Oracle Identity Governance by using the connector installation package.

Figure 2-1 Overall Flow of the Process for Creating an Application By Using the Connector



2.2 Prerequisites for Creating an Application By Using the Connector

Learn about the tasks that you must complete before you create the application.

- Downloading the Connector Installation Package
- **Creating a Target System User Account for Database Application Tables Connector Operations**

2.2.1 Downloading the Connector Installation Package

You can obtain the installation package for your connector on the Oracle Technology Network (OTN) website.

To download the connector installation package:

1. Navigate to the OTN website at http://www.oracle.com/technetwork/middleware/id-mgmt/downloads/connectors-101674.html.
2. Click **OTN License Agreement** and read the license agreement.

3. Select the **Accept License Agreement** option.

   You must accept the license agreement before you can download the installation package.

4. Download and save the installation package to any directory on the computer hosting Oracle Identity Governance.

5. Extract the contents of the installation package to any directory on the computer hosting Oracle Identity Governance. This creates a directory named *CONNECTOR_NAME-RELEASE_NUMBER.*

6. Copy the *CONNECTOR_NAME-RELEASE_NUMBER* directory to the *OIG_HOME*/server/ConnectorDefaultDirectory directory.

2.2.2 Creating a Target System User Account for Database Application Tables Connector Operations

Oracle Identity Governance uses a target system user account to provision to and reconcile data from the target system.

For all target systems certified for this connector, the following are the minimum rights to be assigned to the target system user account:

- **For reconciliation:** The user account must have permissions to run SELECT statements on the tables that must be managed by this connector.
- **For provisioning:** The user account must have permissions to perform select, insert, update, and delete operations on the tables to be managed by this connector.
- If you are configuring the connector to use custom stored procedures to perform connector operations, then the user account must have execute permissions on the relevant stored procedures. See the target system documentation for the procedure to create a target system user account with the preceding permissions required for performing connector operations.

2.3 Creating an Application By Using the Connector

You can onboard an application into Oracle Identity Governance from the connector package by creating a Target application or Authoritative application. To do so, you must log in to Identity Self Service and then choose the **Applications** box on the **Manage** tab.

The following is the high-level procedure to create an application by using the connector:

> **Note:**
>
> For detailed information on each of the steps in this procedure, see Creating Applications of *Oracle Fusion Middleware Performing Self Service Tasks with Oracle Identity Governance*.

1. Create an application in Identity Self Service. The high-level steps are as follows:

   a. Log in to Identity Self Service either by using the **System Administration** account or an account with the **ApplicationInstanceAdministrator** admin role.

   b. Ensure that the **Connector Package** option is selected when creating an application.

c. Update the basic configuration parameters to include connectivity-related information.

d. If required, update the advanced setting parameters to update configuration entries related to connector operations.

e. Add the attributes and provide the mappings:

- If you are using Oracle Identity Governance 12*c* (12.2.1.3.0), then add the attributes and provide the mappings.
- If you are using Oracle Identity Governance 12*c* (12.2.1.4.0), click **Discover**. All attributes are automatically fetched from the database and by default, the Provision field and the Reconciliation field are marked as `true`.

f. Review the provisioning, reconciliation, organization, and catalog settings for your application and customize them if required. For example, you can customize the default correlation rules for your application if required.

g. Review the details of the application and click **Finish** to submit the application details.

The application is created in Oracle Identity Governance.

h. When you are prompted whether you want to create a default request form, click **Yes** or **No**.

If you click **Yes**, then the default form is automatically created and is attached with the newly created application. The default form is created with the same name as the application. The default form cannot be modified later. Therefore, if you want to customize it, click **No** to manually create a new form and attach it with your application.

2. Verify reconciliation and provisioning operations on the newly created application.

> **Note:**
>
> - For Application on Boarding: Export the HANA Database JDBC Driver, for example, `ngdbc-2.4.64.jar` to *OIM_SEVER_CLASSPATH*.
> - For Connector Installation: To run the DBAT Generator, copy the HANA Database JDBC Driver, for example, `ngdbc-2.4.64.jar` to the *dbat-generator-RELEASE_NUMBER*/lib/ directory.

Note:

For Connector installation, under the **Configuration** section, update DBATConfiguration.groovy file with below parameters:

- JDBC driver class name
  Sample value for SAP HANA DB: `'jdbcDriver': 'com.sap.db.jdbc.Driver',`
- JDBC URL template of the target database
  Sample value for SAP HANA DB: `'jdbc:sap://acmedb.com:30015',`

Parameter	Type	Mandatory	Required for JDBC Driver Configuration?	Required for DataSource Configuration?	Default Value	Description
sapHanaDb	Boolean	Yes	No	No	NA	This property suggests sapHanaDb parameter support, if using for sapHanaDb only. Sample value: `True`

For more information on connector installation, see Installing the Connector of *Oracle Identity Manager Connector Guide for Database Application Tables*.

3

Configuring the Database Application Tables Connector

While creating an application, you must configure connection-related parameters that the connector uses to connect Oracle Identity Governance with your target system and perform connector operations. In addition, you can view and edit attribute mappings between the process form fields in Oracle Identity Governance and target system columns, predefined correlation rules, situations and responses, and reconciliation jobs.

3.1 Basic Configuration Parameters

These are the connection-related parameters that Oracle Identity Governance requires to connect to the target. These parameters are common for both target applications and authoritative applications.

Table 3-1 Parameters in the Basic Configuration Section

Parameter	Type	Mandatory?	Required for JDBC Driver Configuration ?	Required for DataSource Configuration ?	Default Value	Description
host	String	Yes	Yes, when %h	No	NA	Host name or IP address of the computer hosting the target system. Sample value: *HOST_IP_ADDRESS*
port	String	Yes	Yes, when %p	No	NA	Enter the number of the port at which the target system database is listening. Sample value: *PORT_NUMBER*

Table 3-1 (Cont.) Parameters in the Basic Configuration Section

Parameter	Type	Mandatory?	Required for JDBC Driver Configuration ?	Required for DataSource Configuration ?	Default Value	Description
database	String	Yes	Yes, when %d	No	NA	Name of the target database. Sample value: *DB_NAME*
jdbcDriver	String	No	Yes	No	NA	JDBC driver class name. Sample value for Oracle database: `oracle.jdbc.driver.OracleDriver` Sample value for MySQL: `com.mysql.jdbc.Driver` Sample value for MS SQL: `com.microsoft.sqlserver.jdbc.SQLServerDriver` Sample value for DB2: `com.ibm.db2.jcc.DB2Driver`

Table 3-1 (Cont.) Parameters in the Basic Configuration Section

Parameter	Type	Mandatory?	Required for JDBC Driver Configuration ?	Required for DataSource Configuration ?	Default Value	Description
jdbcUrlTemplate	String	No	Yes(%h, %p, and, %d)	Yes Provide the value as NA when DataSource is configured.	NA	JDBC URL template of the target database. The value that you specify depends on the database product that you are using. Sample value for Oracle database with SID: `jdbc:oracle:thin:@mydb.com:PORT:oim` Sample value for Oracle database with Service Name: `jdbc:oracle:thin:@mydb.com:PORT/oim` Sample value for MySQL: `jdbc:mysql://mydb.com:PORT/mysql` Sample value for MS SQL: `jdbc:sqlserver://mydb.com:PORT;Database=acmedb` Sample value for DB2: `jdbc:db2://mydb.com:PORT/mydb`

Table 3-1 (Cont.) Parameters in the Basic Configuration Section

Parameter	Type	Mandatory?	Required for JDBC Driver Configuration ?	Required for DataSource Configuration ?	Default Value	Description
user	String	Yes	Yes	Yes for Oracle Database No for other databases	NA	User ID of the database user account that Oracle Identity Governance uses to connect to the target system. Sample value: *DB_USERNAME*
password	String	Yes	Yes	No	NA	Password of the database user account that Oracle Identity Governance uses to connect to the target system. Sample value: *DB_PASSWORD*
table	String	Yes	Yes	Yes	NA	Name of the parent table or view that contains user records. Sample value: *DB_TABLE_NAME*
keyColumn	String	Yes	Yes	Yes	NA	Name of the column that uniquely identifies each row in the parent table. Sample value: *PRIMARY_KEY_OF_DB_PARENT_TABLE*

Table 3-1 (Cont.) Parameters in the Basic Configuration Section

Parameter	Type	Mandatory?	Required for JDBC Driver Configuration ?	Required for DataSource Configuration ?	Default Value	Description
passwordColumn	String	No	No	No	NA	Name of the column in the parent table that holds the passwords of the target system records. This is an optional parameter. **Note:** The value for this parameter is the same as the value specified for the passwordColumn property in the Config entry. You cannot change the value in the IT resource. Sample value: *PASSWORD*

Table 3-1 (Cont.) Parameters in the Basic Configuration Section

Parameter	Type	Mandatory?	Required for JDBC Driver Configuration ?	Required for DataSource Configuration ?	Default Value	Description
statusColumn	Boolean	No	No	No	NA	Name of the column in the target system that holds the status of a user record. You must specify a value for this attribute only if both the following conditions are true: • You want to perform the enable user account or disable user account provisioning operations. • There exists a column in the target system that holds the status of a user record. Sample value: `ACTIVE`
enableValue	String	No	No	No	NA	Value used on the target system that depicts that a user record is in the enabled status. Sample value: `enable`

Table 3-1 (Cont.) Parameters in the Basic Configuration Section

Parameter	Type	Mandatory?	Required for JDBC Driver Configuration ?	Required for DataSource Configuration ?	Default Value	Description
disableValue	String	No	No	No	NA	Value used on the target system that depicts that a user record is in the disabled status. Sample value: `disable`
relationTables	String	No	No	No	NA	A comma-separated list of child table names when user data is spread across parent and child tables. Sample value: *`CHILD_DB_TABLE_NAME`*
Connector Server Name	String	No	No	No	NA	Name of the connector server IT resource. Sample value: *`CONNECTOR_SERVER_NAME`*
validConnectionQuery	String	No	No	No	NA	If no value is specified for this property, then the connection is validated by switching the auto commit mode. For example, you might have the following query, which might be more efficient for some databases: `SELECT 1 FROM DUMMY`

Table 3-1 (Cont.) Parameters in the Basic Configuration Section

Parameter	Type	Mandatory?	Required for JDBC Driver Configuration ?	Required for DataSource Configuration ?	Default Value	Description
changeLogColumn	String	No	No	No	NA	Name of the column where the last update-related, non-decreasing, value is stored. Can be a number or a timestamp. The data type of this column can be any of the data types supported by the target system. However, if you are using Oracle Database, then data types such as BLOB, CLOB, and LONG are not supported. See Supported Data Types for information about data types supported for your target system. The values in this column are used during incremental reconciliation to determine the newest or youngest record reconciled from the target system. **Note:** You must specify a value for this property if you want to

Table 3-1 (Cont.) Parameters in the Basic Configuration Section

Parameter	Type	Mandatory?	Required for JDBC Driver Configuration ?	Required for DataSource Configuration ?	Default Value	Description
						perform incremental reconciliation.
customizedQuery	String	No	No	No	NA	A WHERE clause in a SQL query specifying the subset of newly added or modified records that you want to reconcile. The WHERE clause can contain relations to other tables or views.
allNative	Boolean	No	No	No	false	If value of this property is `false`, then attribute data is converted to Strings by using the JDBC driver. Set the value of this property to `true` to use the appropriate JDBC types and to force the connector to perform the conversion. The new Date format and Timestamps format invalidate this setup.

Table 3-1 (Cont.) Parameters in the Basic Configuration Section

Parameter	Type	Mandatory?	Required for JDBC Driver Configuration ?	Required for DataSource Configuration ?	Default Value	Description
dateformat	String	No	No	No	dd/MM/yyyy	Allows the user to format how date data is converted to strings. • If you want to handle date data as a date editor, then do not enter any value for this parameter. • If you want to handle date data as text, then you must enter the date format. Specifying a value for this parameter invalidates the allNative parameter.
timestampFormat	String	No	No	No	dd/MM/yyyy HH:mm:ss:SSS	Allows the user to format how timestamp data is converted to strings. Specifying this property invalidates the nativeTimestamps and allNative properties.

Table 3-1 (Cont.) Parameters in the Basic Configuration Section

Parameter	Type	Mandatory?	Required for JDBC Driver Configuration ?	Required for DataSource Configuration ?	Default Value	Description
nativeTimestamps	Boolean	No	No	No	false	If the value of this property is set to `false`, then timestamp data is read as Strings, which can cause a loss of time in milliseconds. If the value of this property is set to `true`, then timestamp data is retrieved as java.sql.Timestamp type, and then the connector performs the conversion.
enableEmptyString	Boolean	No	No	No	false	Set to `true` if you want to enable support for writing an empty string instead of a NULL value. Set to `false` if empty strings must be written as NULL values. **Note:** This property can be applied only to mandatory String attributes

Table 3-1 (Cont.) Parameters in the Basic Configuration Section

Parameter	Type	Mandatory?	Required for JDBC Driver Configuration ?	Required for DataSource Configuration ?	Default Value	Description
quoting	String	No	No	No	None	Column quoting property (such as `None`, `Single`, `Double`, `Back`, or `Brackets`) that best fits your target system database. Column names are displayed between single quotes, double quotes, back quotes, or brackets in the generated SQL when accessing the database.
jdbcDriver	String	No	Yes	No	NA	JDBC driver class name. Sample value for SAP HANA DB: `'jdbcDriver': 'com.sap.db.jdbc.Driver',`
jdbcUrlTemplate	String	No	Yes	No	NA	JDBC URL template of the target database. Sample value for SAP HANA DB: `'jdbc:sap://acmedb.com:30015',`

Table 3-1 (Cont.) Parameters in the Basic Configuration Section

Parameter	Type	Mandatory?	Required for JDBC Driver Configuration ?	Required for DataSource Configuration ?	Default Value	Description
rethrowAllSQL Exceptions	Boolean	No	No	No	false	Set to `false` if SQL exceptions with a zero (0x00) error code must be considered a success. In other words, SQL exceptions with the zero error code are caught and suppressed by the SQL statement. Otherwise, set to `true`.

3.2 Advanced Settings Parameters

These are the configuration-related entries that the connector uses during reconciliation and provisioning operations.

> **Note:**
>
> Unless specified, the parameters in the table are applicable to both target and authoritative applications.

Table 3-2 Advanced Setting Parameters

Parameter	Mandatory?	Required for JDBC Driver Configuration?	Required for DataSource Configuration?	Default Value	Description
Connector Name	Yes	Yes	Yes	`org.identityconnectors.databasetable.DatabaseTableConnector`	This parameter holds the name of the connector class.
Connector Bundle	Yes	Yes	Yes	`org.identityconnectors.databasetable`	This parameter holds the name of the connector bundle package.

Table 3-2 (Cont.) Advanced Setting Parameters

Parameter	Mandatory?	Required for JDBC Driver Configuration?	Required for DataSource Configuration?	Default Value	Description
Connector Version	Yes	Yes	Yes	12.3.0	This parameter holds the version of the connector bundle class.
Pool Max Idle	No	No	No	10	Maximum number of idle objects in a pool.
Pool Max Size	No	No	No	10	Maximum number of connections that the pool can create.
Pool Max Wait	No	No	No	150000	Maximum time, in milliseconds, the pool must wait for a free object to make itself available to be consumed for an operation.
Pool Min Evict Idle Time	No	No	No	120000	Minimum time, in milliseconds, the connector must wait before evicting an idle object.
Pool Min Idle	No	No	No	1	Minimum number of idle objects in a pool.
datasource	No	No	Yes	NA	Data source name for the data source naming properties. Sample value: jdbc/ operationsDB

Table 3-2 (Cont.) Advanced Setting Parameters

Parameter	Mandatory?	Required for JDBC Driver Configuration?	Required for DataSource Configuration?	Default Value	Description
jndiProperties	No	No	Yes	NA	Properties used to establish a connection with the target system by using JDBC drivers, enable additional connection properties, or look up a DataSource using JNDI. Sample value: `"java.naming.factory.initial=weblogic.jndi.WLInitialContextFactory","java.naming.provider.url=t3://example.com:15000","java.naming.security.principal=weblogic","java.naming.security.credentials=`*`WEBLOGIC_PASSWORD`*`"`

Table 3-2 (Cont.) Advanced Setting Parameters

Parameter	Mandatory?	Required for JDBC Driver Configuration?	Required for DataSource Configuration?	Default Value	Description
createScript	No	No	No	None	This property is present only in the section for target resource configuration. Specify a value for this property only if you want to configure the connector to use custom stored procedures or SQL statements rather than default SQL statements for performing provisioning operations. Enter the Groovy script or the file URL of the Groovy script created for the create user account provisioning operation. When this script is called, the parent form data is added. You must enter the file URL in the following format: `file:///URL` Sample value: `file:///home/jdoe/dbat/scripts/create_user.groovy`

Table 3-2 (Cont.) Advanced Setting Parameters

Parameter	Mandatory?	Required for JDBC Driver Configuration?	Required for DataSource Configuration?	Default Value	Description
updateSript	No	No	No	None	This property is present only in the section for target resource configuration. Specify a value for this property only if you want to configure the connector to use custom stored procedures or SQL statements rather than default SQL statements for performing provisioning operations. Enter the Groovy script or the file URL of the Groovy script created for the update user account provisioning operation. This script is called when you update the parent form, or enable or disable the user account. You must enter the file URL in the following format: `file:///URL` Sample value: `file:///home/jdoe/dbat/scripts/update_user.groovy`

Table 3-2 (Cont.) Advanced Setting Parameters

Parameter	Mandatory?	Required for JDBC Driver Configuration?	Required for DataSource Configuration?	Default Value	Description
deleteScript	No	No	No	None	This property is present only in the section for target resource configuration. Specify a value for this property only if you want to configure the connector to use custom stored procedures or SQL statements rather than default SQL statements for performing provisioning operations. Enter the Groovy script or the file URL of the groovy script created for the delete user account provisioning operation. This script is called when you remove or delete an account without child data. You must enter the file URL in the following format: `file:///URL` Sample value: `file:///home/jdoe/dbat/scripts/delete_user.groovy`

Table 3-2 (Cont.) Advanced Setting Parameters

Parameter	Mandatory?	Required for JDBC Driver Configuration?	Required for DataSource Configuration?	Default Value	Description
executeQueryScript	No	No	No	None	Specify a value for this property only if you want to configure the connector to use custom stored procedures or SQL queries rather than default SQL queries to perform reconciliation. Enter the Groovy script or the file URL of the Groovy script created for reconciliation. The connector delegates the reconciliation operation to the Groovy script, which is responsible for passing the information (connector object) to the callback handler. This script is called while performing an account search (operations such as full and filtered reconciliation). You must enter the file URL in the following format: `file:///URL` Sample value: `file:///home/jdoe/dbat/scripts/recon_user.groovy`

Table 3-2 (Cont.) Advanced Setting Parameters

Parameter	Mandatory?	Required for JDBC Driver Configuration?	Required for DataSource Configuration?	Default Value	Description
lookupScript	No	No	No	None	This property is present only in the section for target resource configuration. Specify a value for this property only if you want to configure the connector to use custom stored procedures or SQL queries rather than default SQL queries to perform lookup field synchronization. Enter the Groovy script or the file URL of the Groovy script created for lookup field synchronization. You must enter the file URL in the following format: `file:///URL` Sample value: `file:///home/jdoe/dbat/scripts/lookup_field_sync.groovy`

Table 3-2 (Cont.) Advanced Setting Parameters

Parameter	Mandatory?	Required for JDBC Driver Configuration?	Required for DataSource Configuration?	Default Value	Description
syncScript	No	No	No	None	Specify a value for this property only if you want to configure the connector to use custom stored procedures or SQL queries rather than default SQL queries to perform incremental reconciliation. Enter the Groovy script or the file URL of the Groovy script created for incremental reconciliation. You must enter the file URL in the following format: `file:///URL` Sample value: `file:///home/jdoe/dbat/scripts/increm_recon_user.groovy`

Table 3-2 (Cont.) Advanced Setting Parameters

Parameter	Mandatory?	Required for JDBC Driver Configuration?	Required for DataSource Configuration?	Default Value	Description

addMultiValuedAttributeScript	No	No	No	None	This property is present only in the section for target resource configuration.
					Specify a value for this property only if you want to configure the connector to use custom stored procedures or SQL statements rather than default SQL statements for performing provisioning operations.
					Enter the Groovy script or the file URL of the Groovy script created for the add multivalued attribute provisioning operation. This script is called when you add multivalued child attributes.
					You must enter the file URL in the following format:
					`file:///URL`
					Sample value:
					`file:///home/jdoe/dbat/scripts/add_mulval_attr.groovy`

Table 3-2 (Cont.) Advanced Setting Parameters

Parameter	Mandatory?	Required for JDBC Driver Configuration?	Required for DataSource Configuration?	Default Value	Description
removeMultiValuedAttributeScript	No	No	No	None	This property is present only in the section for target resource configuration. Specify a value for this property only if you want to configure the connector to use custom stored procedures or SQL statements rather than default SQL statements for performing provisioning operations. Enter the Groovy script or the file URL of the Groovy script created for lookup field synchronization. This script is called while removing multivalued child attributes. You must enter the file URL in the following format: `file:///URL` Sample value: `file:///home/jdoe/dbat/scripts/remove_mulval_attr.groovy`

Table 3-2 (Cont.) Advanced Setting Parameters

Parameter	Mandatory?	Required for JDBC Driver Configuration?	Required for DataSource Configuration?	Default Value	Description
sapHanaDb	Yes	No	No	NA	This property suggests sapHanaDb parameter support, if using for sapHanaDb only. Sample value: `True`

3.3 Attribute Mappings for an Oracle Database Target Application

The Schema page for a target application displays the default schema (provided by the connector) that maps Oracle Identity Governance attributes to target system columns. The connector uses these mappings during reconciliation and provisioning operations.

Table 3-3 lists the user-specific attribute mappings between the process form fields in Oracle Identity Governance and Oracle Database columns. The table also lists whether a specific attribute is used during provisioning or reconciliation and whether it is a matching key field for fetching records during reconciliation. By default, there are two schema attributes.

For the DBAT connector, you must manually add the attributes based on the database table in the target application. See Creating a Target Application in *Performing Self Service Tasks with Oracle Identity Governance* for information about adding attribute mappings.

Table 3-3 Default Attribute Mappings for Oracle DB User Account

Display Name	Target Attribute	Data Type	Mandatory Provisioning Property?	Provision Field?	Recon Field?	Key Field?	Case Insensitive?
Unique Id	_UID_	String	No	No	Yes	yes	No
Password	_PASSWORD_	String	No	Yes	No	No	No

3.4 Rules, Situations, and Responses

Learn about the predefined rules, responses, and situations for target and authoritative applications.

The connector uses these rules and responses for performing reconciliation.

- Rules, Situations, and Responses for a Target Application

- Rules, Situations, and Responses for an Authoritative Application

3.4.1 Rules, Situations, and Responses for a Target Application

The connector uses predefined rules, responses, and situations for a target application for performing reconciliation.

- Predefined Identity Correlation Rules for a Target Application
- Predefined Situations and Responses for a Target Application

3.4.1.1 Predefined Identity Correlation Rules for a Target Application

By default, the Database Application Tables connector provides a simple correlation rule when you create a target application. The connector uses this correlation rule to compare the entries in Oracle Identity Governance repository and the target system repository, determines the difference between the two repositories, and applies the latest changes to Oracle Identity Governance.

Table 3-4 lists the default simple correlation rule for Database Application Tables connector. If required, you can edit the default correlation rule or add new rules. You can also create complex correlation rules. For more information about adding or editing simple or complex correlation rules, see Creating a Target Application in *Oracle Fusion Middleware Performing Self Service Tasks with Oracle Identity Governance*.

Table 3-4 Predefined Identity Correlation Rule for a Database Application Tables Target Application

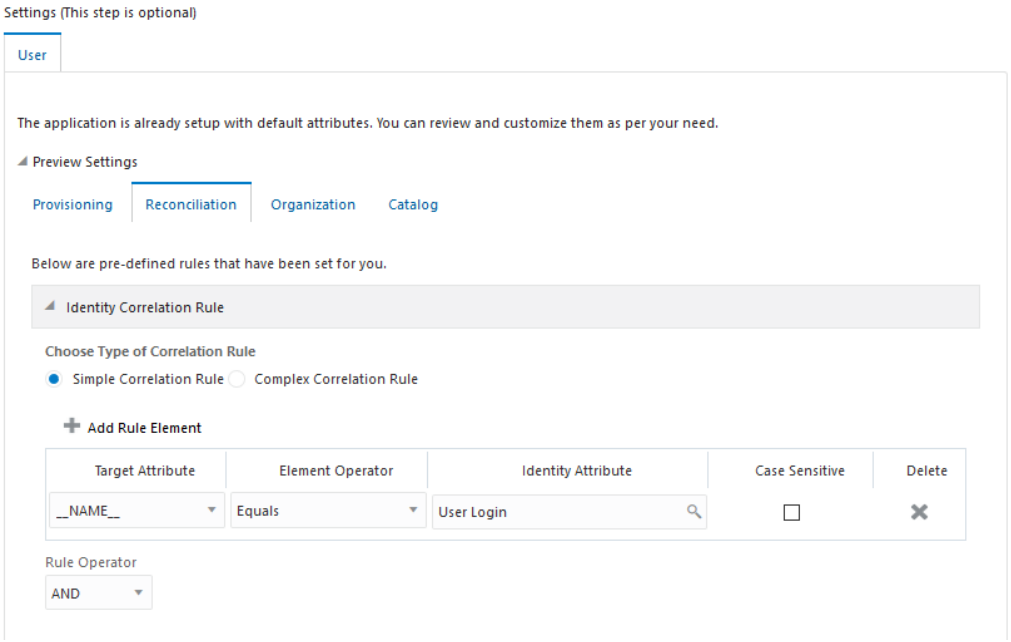
Target Attribute	Element Operator	Identity Attribute	Case Sensitive?
NAME	Equals	User Login	No

In this identity rule:

- __NAME__ is a single-valued attribute on the target system that identifies the user account.
- User Login is the field on the OIG User form.

Figure 3-1 shows the simple correlation rule for a Database Application Tables target application.

Figure 3-1 Simple Correlation Rule for a Database Application Tables Target Application



3.4.1.2 Predefined Situations and Responses for a Target Application

The Database Application Tables connector provides a default set of situations and responses when you create a target application. These situations and responses specify the action that Oracle Identity Governance must take based on the result of a reconciliation event.

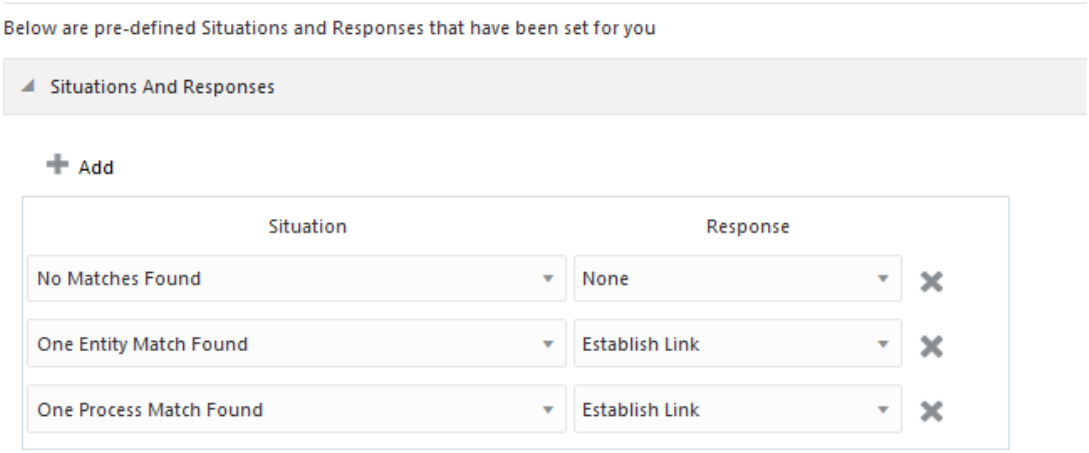
Table 3-5 lists the default situations and responses for a Database Application Tables target application. If required, you can edit these default situations and responses or add new ones. For more information about adding or editing situations and responses, see Creating a Target Application in *Oracle Fusion Middleware Performing Self Service Tasks with Oracle Identity Governance*.

Table 3-5 Predefined Situations and Responses for a Database Application Tables Target Application

Situation	Response
No Matches Found	Assign to Administrator With Least Load
One Entity Match Found	Establish Link
One Process Match Found	Establish Link

Figure 3-2 shows the situations and responses for Database Application Tables that the connector provides by default.

Figure 3-2 Predefined Situations and Responses for a Database Application Tables Target Application



3.4.2 Rules, Situations, and Responses for an Authoritative Application

Learn about the predefined rules, responses, and situations for an authoritative application.

The connector uses these rules and responses for performing reconciliation.

- Predefined Identity Correlation Rules for an Authoritative Application
- Predefined Situations and Responses for an Authoritative Application

3.4.2.1 Predefined Identity Correlation Rules for an Authoritative Application

By default, the Database Application Tables connector provides a simple correlation rule when you create an authoritative application. The connector uses this correlation rule to compare the entries in Oracle Identity Governance repository and the authoritative application repository, determines the difference between the two repositories, and applies the latest changes to Oracle Identity Governance.

Table 3-6 lists the default simple correlation rule for Database Application Tables connector. If required, you can edit the default correlation rule or add new rules. You can also create complex correlation rules. For more information about adding or editing simple or complex correlation rules, see Creating a Target Application in *Oracle Fusion Middleware Performing Self Service Tasks with Oracle Identity Governance*.

Table 3-6 Predefined Identity Correlation Rule for a Database Application Tables Authoritative Application

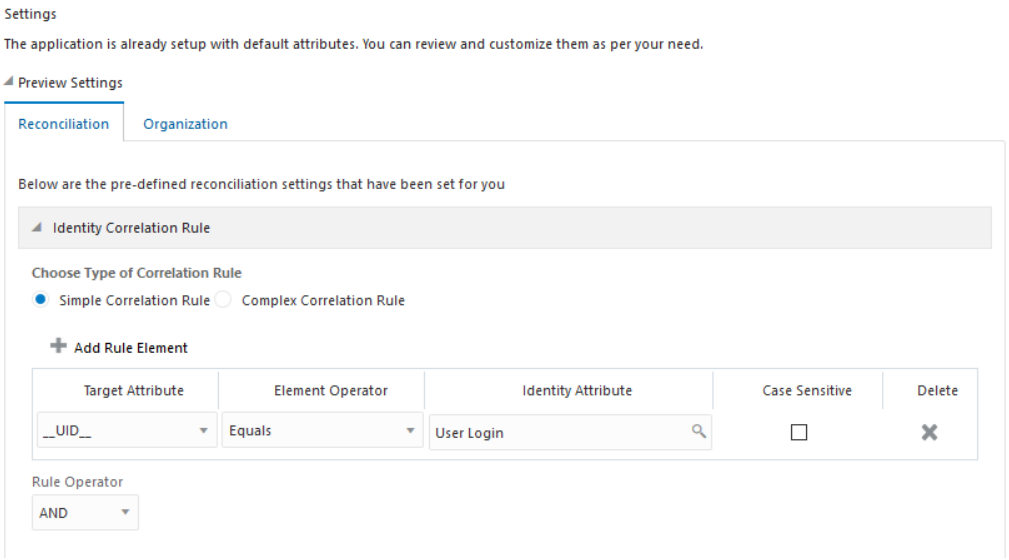
Authoritative Attribute	Element Operator	Identity Attribute	Case Sensitive?
UID	Equals	User Login	No

In this identity rule:

- __UID__ is an attribute on the target system that uniquely identifies the user account.
- User Login is the field on the OIG User form.

Figure 3-3 shows the simple correlation rule for a Database Application Tables authoritative application.

Figure 3-3 Simple Correlation Rule for a Database Application Tables Authoritative application



3.4.2.2 Predefined Situations and Responses for an Authoritative Application

The Database Application Tables connector provides a default set of situations and responses when you create an Authoritative application. These situations and responses specify the action that Oracle Identity Governance must take based on the result of a reconciliation event.

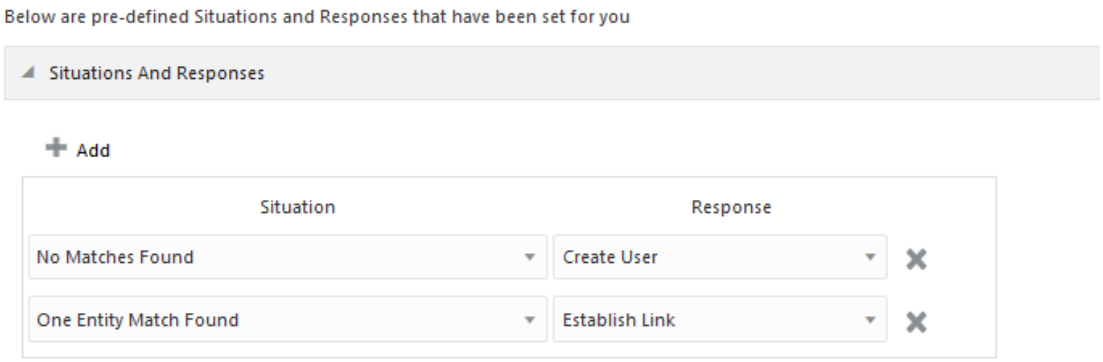
Table 3-7 lists the default situations and responses for a Database Application Tables authoritative application. If required, you can edit these default situations and responses or add new ones. For more information about adding or editing situations and responses, see Creating a Target Application in *Oracle Fusion Middleware Performing Self Service Tasks with Oracle Identity Governance*.

Table 3-7 Predefined Situations and Responses for a Database Application Tables Authoritative Application

Situation	Response
No Matches Found	Create User
One Entity Match Found	Establish Link

Figure 3-4 shows the situations and responses for Database Application Tables that the connector provides by default.

Figure 3-4 Predefined Situations and Responses for a Database Application Tables Authoritative Application



3.5 Reconciliation Scheduled Jobs

When you run the Connector Installer, scheduled jobs are automatically created in Oracle Identity Governance.

This section discusses the following topics:

- Scheduled Job for Lookup Field Synchronization
- Attributes of the Scheduled Jobs

3.5.1 Scheduled Job for Lookup Field Synchronization

The *RESOURCE* Lookup Reconciliation scheduled job is used for lookup field synchronization. You must specify values for the attributes of this scheduled job.

Table 3-8 describes the attributes of the *RESOURCE* Lookup Reconciliation scheduled job.

> **Note:**
>
> - Attribute values are predefined in the connector XML file that you import. Specify values only for those attributes that you want to change.
> - Values (either default or user-defined) must be assigned to all the attributes. If even a single attribute value were left empty, then reconciliation would not be performed.

Table 3-8 Attributes of the RESOURCE Lookup Reconciliation Scheduled Job

Attribute	Description
Code Key Attribute	Enter the name of the attribute that is used to populate the Code Key column of the lookup definition (specified as the value of the Lookup Name attribute). The value must be in the following format: • When scripts are not being used: *TABLE_NAME.COLUMN_NAME* Sample value: `ROLES.ROLE_ID` • When scripts are being used, it would be according to the script mentioned in groovy file. Sample value: `Code Key Attribute-roleId` Where, roleId is the columns in the table on which lookup is being run.
Decode Attribute	Enter the name of the attribute that is used to populate the Decode column of the lookup definition (specified as the value of the Lookup Name attribute). The value must be in the following format: • When scripts are not being used: *TABLE_NAME.COLUMN_NAME* Sample value: `ROLES.ROLE_NAME` • When scripts are being used, it would be according to the script mentioned in groovy file. Sample value: `Decode Attribute-roleName` Where, roleName is the columns in the table on which lookup is being run.
IT Resource Name	Enter the name of the IT resource for the target system installation from which you want to reconcile records. Default value: `DBAT Lookup`
Lookup Name	Enter the name of the lookup definition in Oracle Identity Governance that must be populated with values fetched from the target system. Default value: `Lookup.DBAT.Example` **Note:** Before you perform lookup field synchronization, the lookup definition name that you specify must exist in Oracle Identity Governance.
Object Type	Enter the type of object you want to reconcile. Default value: `Other` **Note:** For lookup field synchronization, the object type must be any object other than "User."

3.5.2 Attributes of the Scheduled Jobs

This section discusses the attributes of the following scheduled jobs:

- Scheduled Jobs for Reconciliation of User Records
- Scheduled Jobs for Reconciliation of Deleted Users Records
- Scheduled Jobs for Incremental Reconciliation

3.5.2.1 Scheduled Jobs for Reconciliation of User Records

After you create the connector, the scheduled task for user data reconciliation is automatically created in Oracle Identity Governance. A scheduled job, which is an

instance of this scheduled task is used to reconcile user data from the target system. The following scheduled jobs are used for user data reconciliation:

- *RESOURCE* Target Resource User Reconciliation

  This scheduled job is used to reconcile user data in the target resource (account management) mode of the connector.

- *RESOURCE* Trusted Resource User Reconciliation

  This scheduled job is used to reconcile user data in the trusted source (identity management) mode of the connector.

You must specify values for the attributes of the user reconciliation scheduled jobs. Table 3-9 describes the attributes of both scheduled jobs.

Table 3-9 Attributes of the User Reconciliation Scheduled Jobs

Attribute	Description
Filter	Enter the search filter for fetching records from the target system during a reconciliation run. See Performing Limited Reconciliation for more information.
ITResource Name	Enter the name of the IT resource for the target system installation from which you want to reconcile user records. Sample value: `DBAT`
Object Type	Enter the type of object you want to reconcile. Sample value: `User` **Note:** User is the only object that is supported. Therefore, do not change the value of the attribute.
Resource Object Name	Enter the name of the resource object that is used for reconciliation. Sample value: `DBAT User`
Scheduled Task Name	Name of the scheduled task that is used for reconciliation. The default value of this attribute in the *RESOURCE* Target Resource User Reconciliation scheduled job is RESOURCE `Target Resource User Reconciliation.` The default value of this attribute in the *RESOURCE* Trusted User Reconciliation scheduled job is *`RESOURCE`*`Trusted Resource User Reconciliation.`

3.5.2.2 Scheduled Jobs for Reconciliation of Deleted Users Records

After you create the connector, the scheduled task for reconciling data about deleted user records is automatically created in Oracle Identity Governance. A scheduled job, which is an instance of this scheduled task is used to reconcile user data from the target system. The following scheduled jobs are used for reconciliation of deleted user records data:

- *RESOURCE* Target Resource User Delete Reconciliation

  This scheduled job is used to reconcile data about deleted user records in the target resource (account management) mode of the connector.

- *RESOURCE*Trusted User Delete Reconciliation

  This scheduled job is used to reconcile data about deleted user records in the trusted source (identity management) mode of the connector.

You must specify values for the attributes of the user reconciliation scheduled jobs. Table 3-10 describes the attributes of both scheduled jobs.

Table 3-10 Attributes of the Delete User Reconciliation Scheduled Jobs

Attribute	Description
Filter	No value should be provided in filter.
ITResource Name	Enter the name of the IT resource for the target system installation from which you want to reconcile user records. Sample value: `DBAT`
Object Type	Enter the type of object you want to reconcile. Sample value: `User` **Note:** User is the only object that is supported. Therefore, do not change the value of the attribute.
Resource Object Name	Enter the name of the resource object that is used for reconciliation. Sample value: `DBAT User`

3.5.2.3 Scheduled Jobs for Incremental Reconciliation

When you create a DBAT application, then the scheduled job for incremental reconciliation is automatically created in Oracle Identity Governance. To configure incremental reconciliation, you need to specify a value for the `changeLogColumn` property in the Basic Configurations section of the application.

The following scheduled jobs are used for incremental reconciliation:

- *RESOURCE* Target Incremental Resource User Reconciliation

  This scheduled job is used to perform incremental reconciliation in the target resource (account management) mode of the connector.

- *RESOURCE* Trusted Incremental Resource User Reconciliation

  This scheduled job is used to perform incremental reconciliation in the trusted source (identity management) mode of the connector.

Table 3-9 describes the attributes of both scheduled jobs.

Table 3-11 Attributes of the Scheduled Jobs for Incremental Reconciliation

Attribute	Description
ITResource Name	Enter the name of the IT resource for the target system installation from which you want to reconcile user records. Sample value: `DBAT`
Object Type	Enter the type of object you want to reconcile. Default value: `User` **Note:** User is the only object that is supported. Therefore, do not change the value of the attribute.
Resource Object Name	Enter the name of the resource object that is used for reconciliation. Sample value: `DBAT User`

Table 3-11 (Cont.) Attributes of the Scheduled Jobs for Incremental Reconciliation

Attribute	Description
Scheduled Task Name	Name of the scheduled task that is used for reconciliation. Default value: *RESOURCE* `Target Incremental Resource User Reconciliation`
Sync Token	Depending on the value specified for the changeLogColumn property in the Config entry of the DBATConfiguration.groovy file, this attribute holds one of the following values: • For date or time stamp based columns: This attribute holds the date or time stamp at which the last reconciliation run started. • For columns that are *not* date or time stamp based (for example, numeric or strings): This attribute holds the newest or the most recent value of the changeLog column of the record that was last reconciled. Sample value: `<String>3</String>` **Note:** - *Do not* enter a value for this attribute. The reconciliation engine automatically enters a value in this attribute. - This attribute stores values in an XML serialized format.

4

Performing the Postconfiguration Tasks

These are the tasks that you can perform after creating an application in Oracle Identity Governance.

- Configuring the Connector for a Target System with an Autoincrement Primary Key
- **Configuring Oracle Identity Governance**
- Harvesting Entitlements and Sync Catalog
- Managing Logging for Oracle Identity Governance
- Configuring the IT Resource for the Connector Server
- Localizing Field Labels in UI Forms
- Configuring Secure Communication Between the Target System and Oracle Identity Governance
- Configuring Secure Communication Between the Connector Server and Oracle Identity Governance
- Configuring the Connector for Stored Procedures and Groovy Scripts
- Configuring the Datasource and JNDI Properties

4.1 Configuring the Connector for a Target System with an Autoincrement Primary Key

> **Note:**
>
> Perform the procedure described in this section *only* if both the conditions are true:
>
> - You have configured your target system as a target resource.
> - The key column of the target system is configured with an autoincrement option.

Perform the following steps to configure the connector for a target system with an autoincrement primary key:

- By default, the key column of the target system is mapped to the OIM User Login field in the reconciliation rule. Before you perform any connector operation, you can modify the reconciliation rule to map the OIM User Login field to a different target system column.
- If the key column of the child table has been configured with the autoincrement option, then modify the child form by removing the 'required=true' property for the key field of the child table by using the Design Console.

- If the prepopulate adapter contains a mapping for the key column, then either disable the prepopulate adapter or modify it to remove the connector key column by using the Design Console.

4.2 Configuring Oracle Identity Governance

During application creation, if you did not choose to create a default form, then you must create a UI form for the application that you created by using the connector.

> **Note:**
>
> Perform the procedures described in this section only if you did not choose to create the default form during creating the application.

The following topics describe the procedures to configure Oracle Identity Governance:

- Creating and Activating a Sandbox
- Creating a New UI Form
- Publishing a Sandbox
- Updating an Existing Application Instance with a New Form

4.2.1 Creating and Activating a Sandbox

You must create and activate a sandbox to begin using the customization and form management features. You can then publish the sandbox to make the customizations available to other users.

See Creating a Sandbox and Activating a Sandbox in *Oracle Fusion Middleware Developing and Customizing Applications for Oracle Identity Governance*.

4.2.2 Creating a New UI Form

You can use Form Designer in Oracle Identity System Administration to create and manage application instance forms.

See Creating Forms By Using the Form Designer in *Oracle Fusion Middleware Administering Oracle Identity Governance*.

While creating the UI form, ensure that you select the resource object corresponding to the newly created application that you want to associate the form with. In addition, select the **Generate Entitlement Forms** check box.

4.2.3 Publishing a Sandbox

Before publishing a sandbox, perform this procedure as a best practice to validate all sandbox changes made till this stage as it is difficult to revert the changes after a sandbox is published.

1. In Identity System Administration, deactivate the sandbox.
2. Log out of Identity System Administration.

3. Log in to Identity Self Service using the xelsysadm user credentials and then activate the sandbox that you deactivated in Step 1.
4. In the Catalog, ensure that the application instance form for your resource appears with correct fields.
5. Publish the sandbox. See Publishing a Sandbox in *Oracle Fusion Middleware Developing and Customizing Applications for Oracle Identity Governance*.

4.2.4 Updating an Existing Application Instance with a New Form

For any changes that you do in the schema of your application in Identity Self Service, you must create a new UI form and update the changes in an application instance.

To update an existing application instance with a new form:

1. Create and activate a sandbox.
2. Create a new UI form for the resource.
3. Open the existing application instance.
4. In the Form field, select the new UI form that you created.
5. Save the application instance.
6. Publish the sandbox.

See Also:

- Creating a Sandbox and Activating a Sandbox in *Oracle Fusion Middleware Developing and Customizing Applications for Oracle Identity Governance*
- Creating Forms By Using the Form Designer in *Oracle Fusion Middleware Administering Oracle Identity Governance*
- Publishing a Sandbox in *Oracle Fusion Middleware Developing and Customizing Applications for Oracle Identity Governance*

4.3 Harvesting Entitlements and Sync Catalog

To harvest entitlements and sync catalog:

1. Run the scheduled jobs for lookup field synchronization listed in Scheduled Job for Lookup Field Synchronization.
2. Run the Entitlement List scheduled job to populate Entitlement Assignment schema from child process form table. See Predefined Scheduled Tasks in *Oracle Fusion Middleware Administering Oracle Identity Governance* for more information about this scheduled job.
3. Run the Catalog Synchronization Job scheduled job. See Predefined Scheduled Tasks in *Oracle Fusion Middleware Administering Oracle Identity Governance* for more information about this scheduled job.

4.4 Managing Logging for Oracle Identity Governance

Oracle Identity Governance uses the Oracle Diagnostic Logging (ODL) logging service for recording all types of events pertaining to the connector.

The following topics provide detailed information about logging:

- Understanding Log Levels
- Enabling Logging

4.4.1 Understanding Log Levels

When you enable logging, Oracle Identity Governance automatically stores in a log file information about events that occur during the course of provisioning and reconciliation operations.

ODL is the principle logging service used by Oracle Identity Governance and is based on java.util.logger. To specify the type of event for which you want logging to take place, you can set the log level to one of the following:

- SEVERE.intValue()+100

  This level enables logging of information about fatal errors.
- SEVERE

  This level enables logging of information about errors that might allow Oracle Identity Governance to continue running.
- WARNING

  This level enables logging of information about potentially harmful situations.
- INFO

  This level enables logging of messages that highlight the progress of the application.
- CONFIG

  This level enables logging of information about fine-grained events that are useful for debugging.
- FINE, FINER, FINEST

  These levels enable logging of information about fine-grained events, where FINEST logs information about all events.

These message types are mapped to ODL message type and level combinations as shown in Table 4-1.

Table 4-1 Log Levels and ODL Message Type:Level Combinations

Java Level	ODL Message Type:Level
SEVERE.intValue()+100	INCIDENT_ERROR:1
SEVERE	ERROR:1
WARNING	WARNING:1

Table 4-1 (Cont.) Log Levels and ODL Message Type:Level Combinations

Java Level	ODL Message Type:Level
INFO	NOTIFICATION:1
CONFIG	NOTIFICATION:16
FINE	TRACE:1
FINER	TRACE:16
FINEST	TRACE:32

The configuration file for OJDL is logging.xml, which is located at the following path:

DOMAIN_HOME/config/fmwconfig/servers/*OIM_SERVER*/logging.xml

Here, *DOMAIN_HOME* and *OIM_SERVER* are the domain name and server name specified during the installation of Oracle Identity Governance.

4.4.2 Enabling Logging

To enable logging in Oracle WebLogic Server:

1. Edit the logging.xml file as follows:
   a. Add the following blocks in the file:

```
<log_handler name='dbat-handler' level='[LOG_LEVEL]'
class='oracle.core.ojdl.logging.ODLHandlerFactory'>
<property name='logreader:' value='off'/>
     <property name='path' value='[FILE_NAME]'/>
     <property name='format' value='ODL-Text'/>
     <property name='useThreadName' value='true'/>
     <property name='locale' value='en'/>
     <property name='maxFileSize' value='5242880'/>
     <property name='maxLogSize' value='52428800'/>
     <property name='encoding' value='UTF-8'/>
   </log_handler>

<logger name="ORG.IDENTITYCONNECTORS.DATABASETABLE" level="[LOG_LEVEL]"
useParentHandlers="false">
     <handler name="dbat-handler"/>
     <handler name="console-handler"/>
   </logger>
```

   b. Replace both occurrences of `[LOG_LEVEL]` with the ODL message type and level combination that you require. Table 4-1 lists the supported message type and level combinations.

   Similarly, replace `[FILE_NAME]` with the full path and name of the log file in which you want log messages to be recorded.

   The following blocks show sample values for `[LOG_LEVEL]` and `[FILE_NAME]`:

```
<log_handler name='dbat-handler' level='NOTIFICATION:1'
class='oracle.core.ojdl.logging.ODLHandlerFactory'>
<property name='logreader:' value='off'/>
     <property name='path' value='/<%OIM_DOMAIN%>/servers/oim_server1/logs/
DBATlogs.log'/>
     <property name='format' value='ODL-Text'/>
```

```
        <property name='useThreadName' value='true'/>
        <property name='locale' value='en'/>
        <property name='maxFileSize' value='5242880'/>
        <property name='maxLogSize' value='52428800'/>
        <property name='encoding' value='UTF-8'/>
      </log_handler>

<logger name="ORG.IDENTITYCONNECTORS.DATABASETABLE"
level="NOTIFICATION:1" useParentHandlers="false">
     <handler name="dbat-handler"/>
     <handler name="console-handler"/>
   </logger>
```

With these sample values, when you use Oracle Identity Governance, all messages generated for this connector that are of a log level equal to or higher than the `NOTIFICATION:1` level are recorded in the specified file.

2. Save and close the file.
3. Set the following environment variable to redirect the server logs to a file:

   For Microsoft Windows:

   ```
   set WLS_REDIRECT_LOG=FILENAME
   ```

   For UNIX:

   ```
   export WLS_REDIRECT_LOG=FILENAME
   ```

   Replace ***FILENAME*** with the location and name of the file to which you want to redirect the output.
4. Restart the application server.

4.5 Configuring the IT Resource for the Connector Server

If you have used the Connector Server, then you must configure values for the parameters of the Connector Server IT resource.

After you create the application for your target system, you must create an IT resource for the Connector Server as described in Creating IT Resource of *Oracle Fusion Middleware Administering Oracle Identity Governance*. While creating the IT resource, ensure to use to select **Connector Server** from the **IT Resource Type** list.

In addition, specify values for the parameters of the IT resource for the Connector Server listed in Table 4-2.

Table 4-2 Parameters of the IT Resource for the Connector Server

Parameter	Description
Host	Enter the host name or IP address of the computer hosting the Connector Server. Sample value: `myhost.com`
Key	Enter the key for the Connector Server.

Table 4-2 (Cont.) Parameters of the IT Resource for the Connector Server

Parameter	Description
Port	Enter the number of the port at which the Connector Server is listening. By default, this value is blank. You must enter the port number that is displayed on the terminal when you start the Connector Server. Sample value: `8759`
Timeout	Enter an integer value which specifies the number of milliseconds after which the connection between the Connector Server and Oracle Identity Governance times out. Recommended value: `0` A value of 0 means that the connection never times out.
UseSSL	Enter `true` to specify that you will configure SSL between Oracle Identity Governance and the Connector Server. Otherwise, enter `false`. Default value: `false` **Note:** It is recommended that you configure SSL to secure communication with the connector server. To configure SSL, see Configuring the Java Connector Server with SSL in *Oracle Fusion Middleware Developing and Customizing Applications for Oracle Identity Governance*.

4.6 Localizing Field Labels in UI Forms

To localize a field label that is added to the UI forms:

1. Create a properties file (for example, DBAT_ja.properties) containing localized versions for the column names in your target system (to be displayed as text strings for GUI elements and messages in the Administrative and User Console).
2. Log in to Oracle Enterprise Manager.
3. In the left pane, expand **Application Deployments** and then select **oracle.iam.console.identity.sysadmin.ear.**
4. In the right pane, from the Application Deployment list, select **MDS Configuration.**
5. On the MDS Configuration page, click **Export** and save the archive to the local computer.
6. Extract the contents of the archive, and open the following file in a text editor:

   SAVED_LOCATION/xliffBundles/oracle/iam/ui/runtime/BizEditorBundle_en.xlf
7. Edit the BizEditorBundle.xlf file in the following manner:
   a. Search for the following text:

   ```
   <file source-language="en"
   original="/xliffBundles/oracle/iam/ui/runtime/BizEditorBundle.xlf"
   datatype="x-oracle-adf">
   ```

   b. Replace with the following text:

   ```
   <file source-language="en" target-language="LANG_CODE"
   original="/xliffBundles/oracle/iam/ui/runtime/BizEditorBundle.xlf"
   datatype="x-oracle-adf">
   ```

In this text, replace *LANG_CODE* with the code of the language that you want to localize the form field labels. The following is a sample value for localizing the form field labels in Japanese:

```
<file source-language="en" target-language="ja"
original="/xliffBundles/oracle/iam/ui/runtime/BizEditorBundle.xlf"
datatype="x-oracle-adf">
```

c. Search for the application instance code. This procedure shows a sample edit for Database Application Tables application instance. The original code is:

```
<trans-unit id="$
{adfBundle['oracle.adf.businesseditor.model.util.BaseRuntimeResourceBundl
e']
['persdef.sessiondef.oracle.iam.ui.runtime.form.model.user.entity.userEO.
UD_ACMEDBAP_APP_DFLT_HOME__c_description']}">
<source>APP_DFLT_HOME</source>
<target/>
</trans-unit>
<trans-unit
id="sessiondef.oracle.iam.ui.runtime.form.model.ACMEFORM.entity.ACMEFORME
O.UD_ACMEDBAP_APP_DFLT_HOME__c_LABEL">
<source>APP_DFLT_HOME</source>
<target/>
</trans-unit>
```

d. Open the properties file created in Step 1 and get the value of the attribute, for example, global.udf.D_ACMEDBAP_APP_DFLT_HOME=\u4567d.

e. Replace the original code shown in Step 7.c with the following:

```
<trans-unit id="$
{adfBundle['oracle.adf.businesseditor.model.util.BaseRuntimeResourceBundl
e']
['persdef.sessiondef.oracle.iam.ui.runtime.form.model.user.entity.userEO.
UD_ACMEDBAP_APP_DFLT_HOME__c_description']}">
<source>APP_DFLT_HOME</source>
<target>\u4567d</target>
</trans-unit>
<trans-unit
id="sessiondef.oracle.iam.ui.runtime.form.model.ACMEFORM.entity.ACMEFORME
O.UD_ACMEDBAP_APP_DFLT_HOME__c_LABEL">
<source>APP_DFLT_HOME</source>
<target>\u4567d</target>
</trans-unit>
```

f. Repeat Steps 7.a through 7.d for all attributes of the process form.

g. Save the file as BizEditorBundle_*LANG_CODE*.xlf. In this file name, replace *LANG_CODE* with the code of the language to which you are localizing.

   Sample file name: BizEditorBundle_ja.xlf.

8. Repackage the ZIP file and import it into MDS.

> **See Also:**
>
> Deploying and Undeploying Customizations in *Oracle Fusion Middleware Developing and Customizing Applications for Oracle Identity Governance* for more information about exporting and importing metadata files

9. Log out of and log in to Oracle Identity Governance.

4.7 Configuring Secure Communication Between the Target System and Oracle Identity Governance

> **Note:**
>
> It is recommended that you perform the procedure described in this section to secure communication between the target system and Oracle Identity Governance.

The procedure to secure communication depends on the database that you are using:

- Configuring Secure Communication Between IBM DB2 and Oracle Identity Governance
- Configuring Secure Communication Between Microsoft SQL Server and Oracle Identity Governance
- Configuring Secure Communication Between MySQL and Oracle Identity Governance
- Configuring Secure Communication Between Oracle Database and Oracle Identity Governance

4.7.1 Configuring Secure Communication Between IBM DB2 and Oracle Identity Governance

> **Note:**
>
> - IBM DB2 version 11.*x* and later support secure communication over SSL.
> - Before configuring secure communication between IBM DB2 and Oracle Identity Governance, you must install the IBM Global Security Kit (GSKit).
>
> See the IBM DB2 documentation for more information about enabling SSL communication between IBM DB2 and a client system. In this context, the client is Oracle Identity Governance.

To configure secure communication between IBM DB2 and Oracle Identity Governance:

1. Generate the certificate store by running the GSKit tool. To do so, run the following command:

   ```
   GSKCAPICMD -keydb -create -db "KEY_DATABASE_LOCATION" -pw KEY_DATABASE_PASSWORD -
   stash
   ```

   In the command, replace:

   - *GSKCAPICMD* with the full path and name of the GSKit tool. For example, for the target system running on a 64-bit Microsoft Windows platform, replace *GSKCAPICMD* with `C:\Program Files (x86)\IBM\GSK8\bin\gsk8capicmd_64.exe`.

- *KEY_DATABASE_LOCATION* with the full path and name of the key database to be created.
- *KEY_DATABASE_PASSWORD* with the password for the key database.

The following is a sample command that generates a certificate store (db2oim.kdb):

```
C:\DB2>"\Program Files\IBM\gsk8\bin\gsk8capicmd_64.exe" -keydb -create
-db "c:\db2\db2oim.kdb" -pw PASSWORD -stash
```

2. Generate the self-signed certificate by running the following command:

```
GSKCAPICMD -cert -create -db  "KEY_DATABASE_LOCATION" -pw
KEY_DATABASE_PASSWORD -label "CERT_LABEL" -dn "DISTINCT_NAME"
```

In the command, replace:

- *GSKCAPICMD* with the full path and name of the GSKit tool. For example, for the target system running on a 64-bit Microsoft Windows platform, replace *GSKCAPICMD* with `C:\Program Files (x86)\IBM\GSK8\bin\gsk8capicmd_64.exe`.
- *KEY_DATABASE_LOCATION* with the full path and name of the key database to store the certificate.
- *KEY_DATABASE_PASSWORD* with the password for the key database.
- *CERT_LABEL* with a label that is used to uniquely identify the certificate.
- *DISTINCT_NAME* with the distinguished name that uniquely identifies the certificate.

The following is a sample command that generates a self-signed certificate:

```
C:\DB2>"\Program Files\IBM\gsk8\bin\gsk8capicmd_64.exe" -cert -create
-db "c:\db2\db2oim.kdb" -pw PASSWORD -label "db2oim" -dn
"CN=example.com,O=org,OU=myorg,L=myLocation,ST=CA,C=USA"
```

3. Export the server certificate by running the following command:

```
GSKCAPICMD -cert -extract -db "KEY_DATABASE_LOCATION" -pw
KEY_DATABASE_PASSWORD -label "CERT_LABEL" -target "LOCATION" -format FORMAT -
fips
```

In the command, replace:

- *GSKCAPICMD* with the full path and name of the GSKit tool. For example, for the target system running on a 64-bit Microsoft Windows platform, replace *GSKCAPICMD* with `C:\Program Files (x86)\IBM\GSK8\bin\gsk8capicmd_64.exe`.
- *KEY_DATABASE_LOCATION* with the full path and name of the key database.
- *KEY_DATABASE_PASSWORD* with the password for the key database.
- *CERT_LABEL* with the label that is used to uniquely identify the certificate to be extracted.
- *LOCATION* with the full path and name of the file to which the certificate is to be extracted.
- *FORMAT* with the certificate format, which can be either `ascii` or `binary`.

The following is a sample command that exports the server certificate to db2oim.arm:

```
C:\DB2>"\Program Files\IBM\gsk8\bin\gsk8capicmd_64.exe" -cert -extract -db
"c:\db2\db2oim.kdb" -pw PASSWORD -label "db2oim" -target "c:\db2\db2oim.arm"
-format ascii -fips
```

4. Configure the database to enable both SSL and TCP/IP communication protocols by running the following command:

   ```
   db2set.exe DB2COMM=SSL,TCPIP
   ```

5. Check protocols by using db2set.exe to validate that the SSL and TCP/IP protocols are enabled in DB2COMM.

   DB2PROCESSORS=0,1

   DB2INSTPROF=C:\ProgramData\IBM\DB2\DB2COPY1

   DB2COMM=SSL,TCPIP

6. Verify your SSL settings by running the db2 GET DATABASE MANAGER CONFIGURATION command.

7. Import the certificate into the Java keystore of the application server on which Oracle Identity Governance is running.

   To import the certificate into the Java keystore, run the following command:

   ```
   keytool -importcert -file FILE_LOCATION -alias ALIAS -storepass STORE_PASSWORD -
   keystore STORE_LOCATION
   ```

   In this command, replace:

   - *FILE_LOCATION* with the full path and name of the certificate file.
   - *ALIAS* with an alias for the certificate.
   - *STORE_PASSWORD* with a password for the truststore.
   - *STORE_LOCATION* with one of the truststore paths from

   The following is a sample command that imports the certificate into the Java keystore:

   ```
   C:\DB2>keytool -importcert -file db2oim.arm -alias db2oim -storepass PASSWORD -
   keystore C:\Users\example_user\.keystore
   ```

   The certificate is imported into the keystore.

4.7.2 Configuring Secure Communication Between Microsoft SQL Server and Oracle Identity Governance

To configure secure communication between Microsoft SQL Server and Oracle Identity Governance:

1. Refer to Microsoft SQL Server documentation for information about enabling SSL communication between Microsoft SQL Server and a client system. In this context, the client is Oracle Identity Governance.

   Export the certificate on the Microsoft SQL Server host computer.

2. Copy the certificate to the Oracle Identity Governance host computer.

3. Import the certificate into the JVM truststore of the application server on which Oracle Identity Governance is running.

To import the certificate into the truststore, run the following command:

```
..\..\bin\keytool -import -file FILE_LOCATION -keystore TRUSTSTORE_LOCATION -
storepass TRUSTSTORE_PASSWORD -trustcacerts -alias ALIAS
```

In this command:

- Replace *FILE_LOCATION* with the full path and name of the certificate file.
- Replace *ALIAS* with an alias for the certificate.
- Replace *TRUSTSTORE_PASSWORD* with a password for the truststore.
- Replace *TRUSTSTORE_LOCATION* with the following truststore path:

  JAVA_HOME/jre/lib/security/cacerts

4.7.3 Configuring Secure Communication Between MySQL and Oracle Identity Governance

To configure secure communication between MySQL and Oracle Identity Governance:

1. See MySQL documentation for information about enabling SSL communication between MySQL and a client system. In this context, the client is Oracle Identity Governance.
2. Export the certificate on the MySQL host computer.
3. Restart the MySQL database service by using the certificate exported in the preceding step. See MySQL documentation for information on restarting the database service.
4. Copy the ca-cert.pem and client-cert.pem certificates to the Oracle Identity Governance host computer.
5. Import the certificates into the JVM truststore of the application server on which Oracle Identity Governance is running.

   To import the certificates into the truststore, run the following command for each certificate:

   ```
   keytool -import -file FILE_LOCATION -keystore TRUSTSTORE_LOCATION -storepass
   TRUSTSTORE_PASSWORD -trustcacerts -alias ALIAS
   ```

   In this command:

   - Replace *FILE_LOCATION* with the full path and name of the certificate file.
   - Replace *ALIAS* with an alias for the certificate.
   - Replace *TRUSTSTORE_PASSWORD* with a password for the truststore.
   - Replace *TRUSTSTORE_LOCATION* with the following truststore path:

     JAVA_HOME/jre/lib/security/cacerts

> **Note:**
>
> In an Oracle Identity Governance cluster, you must import the file into the truststore on each node of the cluster.

4.7.4 Configuring Secure Communication Between Oracle Database and Oracle Identity Governance

To secure communication between Oracle Database and Oracle Identity Governance, you can perform either one or both of the following procedures:

- Configuring Data Encryption and Integrity in Oracle Database
- Configuring SSL Communication in Oracle Database

4.7.4.1 Configuring Data Encryption and Integrity in Oracle Database

See Configuring Network Data Encryption and Integrity in *Oracle Database Security Guide* for information about configuring data encryption and integrity.

4.7.4.2 Configuring SSL Communication in Oracle Database

To enable SSL communication between Oracle Database and Oracle Identity Governance:

> **Note:**
>
> See Enabling Secure Sockets Layer in *Oracle Database Security Guide* for detailed information about enabling SSL communication between Oracle Database and Oracle Identity Governance.

1. Export the certificate on the Oracle Database host computer.
2. Copy the certificate to Oracle Identity Governance.
3. Import the certificate into the JVM truststore of the application server on which Oracle Identity Governance is running.

   To import the certificate into the truststore, run the following command:

   ```
   keytool -import -file FILE_LOCATION -keystore TRUSTSTORE_LOCATION -storepass
   TRUSTSTORE_PASSWORD -trustcacerts -alias ALIAS
   ```

   In this command:

   - Replace *FILE_LOCATION* with the full path and name of the certificate file.
   - Replace *ALIAS* with an alias for the certificate.
   - Replace *TRUSTSTORE_PASSWORD* with a password for the truststore.
   - Replace *TRUSTSTORE_LOCATION* with the following truststore path:

     JAVA_HOME/jre/lib/security/cacerts

> **Note:**
>
> In an Oracle Identity Governance cluster, you must import the file into the truststore on each node of the cluster.

4.8 Configuring Secure Communication Between the Connector Server and Oracle Identity Governance

If you have deployed this connector on a Connector Server, then it is recommended that you secure communication between the Connector Server and Oracle Identity Governance. The procedure to configure secure communication is the same as the procedure described in section Configuring Secure Communication Between the Target System and Oracle Identity Governance. While performing the procedure described in that section, consider the Connector Server as a separate system, similar to the target system.

Before you configure secure communication:

- Ensure that the Connector Server is running under a user that has the appropriate rights to access the keystore.
- Ensure that the keystore on the Connector Server is present and accessible.
- Ensure that the keystore on the Connector Server contains the expected certificates.
- If you are not using the default Java keystore on the Connector Server, then modify the keystore paths and password in the IT resource URL or the jndiProperties property (of the DBATConfiguration.groovy file) to match the location on the Connector Server.

4.9 Configuring the Connector for Stored Procedures and Groovy Scripts

The connector runs default SQL queries and SQL statements when you use it to perform reconciliation and provisioning operations, respectively. Instead of default SQL statements and queries, if you want the connector to use custom stored procedures for performing reconciliation or provisioning operations, then you must perform the procedure described in this section.

> **See Also:**
>
> Sample Stored Procedures and Groovy Scripts for sample stored procedures and Groovy scripts

This section contains the following topics:

- Configuring the Connector for Custom Stored Procedures
- Groovy Script Arguments

- Sample Groovy Script
- Entries Specific to Groovy Script Configuration

4.9.1 Configuring the Connector for Custom Stored Procedures

To configure the connector for custom stored procedures:

1. On the target system, create the stored procedures that must be used for performing provisioning operations. The following are sample stored procedures (created on Oracle Database) that run the DELETE SQL statement for deleting the groups and roles child data. For target systems other than Oracle Database, the syntax of this sample procedure may vary.

   The stored procedure for `DELETE_USERGROUP` is as follows:

   ```
   create or replace PROCEDURE DELETE_USERGROUP
   (  userin IN VARCHAR2, gId IN VARCHAR2
   ) AS
   BEGIN
   DELETE from USER_GROUP where USERID=userin and GROUPID=gId;
   END DELETE_USERGROUP;
   ```

   The stored procedure for `DELETE_USERROLE` is as follows:

   ```
   create or replace PROCEDURE DELETE_USERROLE
   (  userin IN VARCHAR2, rId IN VARCHAR2
   ) AS
   BEGIN
   DELETE  from USER_ROLE where USERID=userin and ROLEID=rId;
   END DELETE_USERROLE;
   ```

2. On the Oracle Identity Governance host computer, create Groovy scripts that call the relevant stored procedures on the target system to perform provisioning operations. See Groovy Script Arguments for information about the arguments that can be directly used in the groovy script.

   > **Note:**
   >
   > See Sample Groovy Script for a sample Groovy script that calls the DELETE_USERGROUP and DELETE_USERROLE stored procedure.

3. Update the Advanced configuration details definition to include information about the Groovy scripts as listed in Table 4-3.

   > **Note:**
   >
   > Instead of the file URL of the Groovy script, you can directly enter the Groovy script. In such a case, ensure that the corresponding attribute does not contain `[LOADFROMURL]`. For example, if you directly enter the Groovy script for the create user account provisioning operation, then the corresponding attribute name must be `createScript`, instead of `createScript[LOADFROMURL]`.

The following is a sample value for the removeMultiValuedAttributeScript[LOADFROMURL] entry:

```
file:///home/myname/dbat/scripts/removechilddata.groovy
```

4. To reset the password during the update procedure, do the following:

   a. Check whether script argument "attributes" contains password (__PASSWORD__) attribute.

   ```
   import org.identityconnectors.common.security.GuardedString;
   GuardedString pass = attributes.get("__PASSWORD__")!=null?
   attributes.get"__PASSWORD__").getValue().get(0):null;
   ```

   b. If "attributes" contains __PASSWORD__ attribute (not null), call targetstore procedure/sql query to reset password.

   ```
   upstmt = conn.prepareStatement("UPDATE PASSWORD....
   if(pass!=null){
          pass.access(new GuardedString.Accessor(){
              public void access(char[] clearChars){
                  upstmt.setString(1, new String(clearChars));
              }
          });
   } else {
             //Update other attributes
   }
        upstmt.executeUpdate();
   ```

4.9.2 Groovy Script Arguments

The following arguments can be directly used in the Groovy script:

- connector - The Database Application Tables connector object.
- conn - JDBC connection.
- timing - When the Groovy script is called. In addition, the timing attribute also explains the type of operation being performed. For example, if it is search operation, then the object class being search is also returned.

  The following is the format of the timing argument for lookup field synchronization:

  executeQuery:*OBJECT_CLASS*

  In this format, *OBJECT_CLASS* is replaced with the type of object being reconciled.

  For example, for a lookup field synchronization scheduled job that contains the object type "Role", the value of the timing argument will be as follows:

  ```
  executeQuery:Role
  ```

- attributes - All attributes.
- trace - Logger as a script trace bridge to the application.
- where - String where condition for execute query, or null.
- handler - resultSetHandler or SyncResultsHandler for the connector objects produced by the execute query, sync operation or null return.
- quoting - The type of table name quoting to be used in SQL. The default value is an empty string. The value of this argument is obtained from the IT resource.

- nativeTimestamps - Specifies whether the script retrieves the timestamp data of the columns as java.sql.Timestamp type from the database table. This information is obtained from the IT resource.
- allNative - Specifies whether the script must retrieve the data type of the columns in a native format from the database table. The value of this argument is obtained from the IT resource.
- rethrowAllSQLExceptions - The value of this argument is also obtained from the IT resource. The value of this argument specifies whether the script must throw exceptions when a zero (0x00) error code is encountered.
- enableEmptyString - Specifies whether support for writing an empty string instead of a NULL value must be enabled. The value of this argument is obtained from the IT resource.
- filterString - String filter condition for execute query, or null.
- filterParams - List of filter parameters. Each parameter is present in the *COLUMN_NAME*:*VALUE* format. For example, FIRSTNAME:test.
- syncattribute - Name of the database column configured for incremental reconciliation. This argument is available in the sync script, which is called during an incremental reconciliation run.
- synctoken - Value of the sync attribute. This argument is available in the sync script.

4.9.3 Sample Groovy Script

The following is a sample Groovy script that calls the DELETE_USERGROUP and DELETE_USERROLE stored procedure created in step 1 of Configuring the Connector for Custom Stored Procedures.

```
import org.identityconnectors.framework.common.objects.*;
System.out.println("[removeMultiValuedAttributeScript] Removing Child data::"+
attributes);

try {
childDataEOSet = null;
delSt = null;
//Get UID
String id = attributes.get("__UID__").getValue().get(0);
if(attributes.get("USER_GROUP")!=null)
{
childDataEOSet=attributes.get("USER_GROUP").getValue();
//Delete child data using stored procedure
delSt= conn.prepareCall("{call DELETE_USERGROUP(?,?)}");
    if(childDataEOSet !=null){
System.out.println("[removeMultiValuedAttributeScript] Removing Group data.");
//Iterate through child data and delete
for( iterator = childDataEOSet.iterator(); iterator.hasNext(); )
{
eo = iterator.next();
attrsSet = eo.getAttributes();
grpattr=AttributeUtil.find("GROUPID",attrsSet);
if(grpattr!=null){
groupid=grpattr.getValue().get(0);
delSt.setString(1, id);
delSt.setString(2, groupid);
delSt.executeUpdate();
System.out.println("[removeMultiValuedAttributeScript] Deleted Group::"+ grpattr);
```

```
} }; } }
} finally {
if (delSt != null)
delSt.close();
};
try {
childDataEOSet = null;
delSt = null;
String id      = attributes.get("__UID__").getValue().get(0);
if(attributes.get("USER_ROLE")!=null)
{
childDataEOSet=attributes.get("USER_ROLE").getValue();
delSt= conn.prepareCall("{call DELETE_USERROLE(?,?)}");
    if(childDataEOSet !=null){
System.out.println("[removeMultiValuedAttributeScript] Removing Role data.");
for( iterator = childDataEOSet.iterator(); iterator.hasNext(); )
{
eo = iterator.next();
attrsSet = eo.getAttributes();
roleattr=AttributeUtil.find("ROLEID",attrsSet);
if(roleattr!=null){
rolename=roleattr.getValue().get(0);
delSt.setString(1, id);
delSt.setString(2, rolename);
delSt.executeUpdate();
System.out.println("[removeMultiValuedAttributeScript] Deleted Role::"+
rolename);
} }; } }
} finally {
if (delSt != null)
delSt.close();
};
```

4.9.4 Entries Specific to Groovy Script Configuration

Table 4-3 describes the lookup entries specific to groovy script configuration.

Table 4-3 Entries Specific to Groovy Script Configuration

Code Key	Decode
createScript[LOADFROMURL]	Enter the file URL of the Groovy script created for the create user account provisioning operation.
updateScript[LOADFROMURL]	Enter the file URL of the Groovy script created for the update user account provisioning operation.
deleteScript[LOADFROMURL]	Enter the file URL of the Groovy script created for the delete user account provisioning operation.
executeQueryScript[LOADFROMURL]	Enter the file URL of the Groovy script created for full and filtered reconciliation.
lookupScript[LOADFROMURL]	Enter the file URL of the Groovy script created for lookup field synchronization.
syncScript[LOADFROMURL]	Enter the file URL of the Groovy script created for incremental reconciliation.
addMultiValuedAttributeScript[LOADFROMURL]	Enter the file URL of the Groovy script created for the add multivalued attributes provisioning operation.
removeMultiValuedAttributeScript[LOADFROMURL]	Enter the file URL of the Groovy script created for the remove multivalued attributes provisioning operation.

4.10 Configuring the Datasource and JNDI Properties

Perform the procedure described in this topic if the connector uses datasource configuration to connect to your target system.

To configure the datasource and JNDI properties

1. Login to Oracle WebLogic Server Administration Console.
2. On the Domain Structure left navigation pane, expand **Services**, and click **Data Sources**.
3. Click **Lock & Edit**.
4. In the Configuration tab, below Data Source, click the **New** menu, and select **Generic Data Source**.
5. In the Create a New JDBC Data Source page, provide the following values, and then click **Next**.
   - **Name:** Enter the datasource name.
   - **JNDI Name:** Enter the JNDI name in the format `jdbc/DATASOURCE_NAME`.
   - **Database Type:** Select a database type. For example, if you are using an Oracle database, then select **Oracle**.
6. From the JDBC Driver list, select ***Oracle's Driver (Thin) for Service connections; Versions:Any**, and then click **Next**.
7. Deselect the **Supports Global Transactions** option, and then click **Next**.
8. In the Connection Properties page, provide the following values, and then click **Next**.
   - **Database Name:** Enter the name of the database that you want to connect to.
   - **Host Name:** Enter the name or IP address of the database server.
   - **Port:** Enter the port number of the database server.
   - **Database User Name:** Enter the user name for connecting to the database.
   - **Password:** Enter the password for connecting to the database.
   - **Confirm Password:** Re-enter the password.
9. Click **Test Configuration**.

   A message states that the connection test is successful.
10. Click **Next**.
11. Under Servers, select **AdminServer** and **oim_server1**, and then click **Finish**.
12. Click **Activate Changes** to activate the datasource creation.

> **Note:**
>
> Add the Java property `-Dweblogic.jdbc.remoteEnabled=true` in Weblogic OIM Domain Environment script, and restart the WebLogic server.

4.11 Configuring the Datasource and JNDI Properties for SAP HANA DB

Perform the procedure described in this topic to configure the datasource and JNDI properties for SAP HANA DB:

1. Login to Oracle WebLogic Server Administration Console.
2. On the Domain Structure left navigation pane, expand **Services**, and click **DataSources**.
3. Click **Lock & Edit**.
4. In the Configuration tab, below Data Source, click the **New** menu, and select **Generic Data Source**.
5. In the Create a New JDBC Data Source page, provide the following values, and then click **Next**.
   - Name: Enter the datasource name.
   - JNDI Name: Enter the JNDI name in the format *jdbc/DATASOURCE_NAME*.
   - Database Type: Select a database type. For example, if you are using an SAP Hana database, then select **Other** and then click **Next**.
6. From the JDBC Driver list, select **Other** for Service connections and then click **Next**.
7. Deselect the **Supports Global Transactions** option, and then click **Next**.
8. In the Connection Properties page, provide the following values, and then click **Next**.
   - Database Name: Enter the name of the database that you want to connect to.
   - Password: Enter the password for connecting to the database, and then click **Next**.
9. Test Database Connection:
   - Driver Class Name: Enter the name or IP address of the database server.
   - url: Enter the port number of the database server.
   - Database User Name: Enter the user name for connecting to the database.
   - Password: Enter the password for connecting to the database.
   - Confirm Password: Re-enter the password.
10. Click **Test Configuration**. A message states that the connection test is successful.
11. Click **Next**.
12. Under Servers, select **AdminServer** and **oim_server1**, and then click **Finish**.
13. Click **Activate Changes** to activate the Data Source creation.

5

Using the Database Application Tables Connector

You can use the connector for performing reconciliation and provisioning operations after configuring it to meet your requirements.

This chapter provides information about the following topics:

- Configuring Reconciliation
- Performing Provisioning Operations
- Reconciliation Scheduled Jobs
- Uninstalling the Connector

5.1 Configuring Reconciliation

Reconciliation involves duplicating in Oracle Identity Governance the creation of and modifications to user accounts on the target system. This section discusses the following topics related to configuring reconciliation:

- Performing Full Reconciliation and Incremental Reconciliation
- Performing Limited Reconciliation

5.1.1 Performing Full Reconciliation and Incremental Reconciliation

Full reconciliation involves reconciling all existing user records from the target system into Oracle Identity Governance. After you deploy the connector, you must first perform full reconciliation. In addition, you can switch from incremental reconciliation to full reconciliation whenever you want to ensure that all target system records are reconciled in Oracle Identity Governance.

You can perform a full reconciliation run in one of the following manners:

- Ensure that no value is specified for the Filter attribute of the scheduled job for user data reconciliation. See Scheduled Jobs for Reconciliation of User Records for information about the Filter attribute.
- Ensure the Sync Token attribute of the scheduled job for incremental reconciliation does not contain any value. See Scheduled Jobs for Incremental Reconciliation for information about the Sync Token attribute.

In incremental reconciliation, only records created or modified after the latest date/ timestamp the last reconciliation was run are considered for reconciliation. To perform incremental reconciliation, configure and run the scheduled job for incremental reconciliation. The first time you run the scheduled job for incremental reconciliation, note that a full reconciliation is performed. Note that the scheduled job for incremental reconciliation is generated only if you specify a last update column value for the changeLogColumn property in the DBATConfiguration.groovy file.

5.1.2 Performing Limited Reconciliation

By default, all target system records that are added or modified after the last reconciliation run are reconciled during the current reconciliation run. You can customize this process by specifying the subset of added or modified target system records that must be reconciled. You do this by creating filters for the reconciliation module.

You can configure limited reconciliation by performing the procedures described in one of the following sections:

- Specifying a Value for the Filter Attribute
- Specifying a Value for the customizedQuery Parameter

5.1.2.1 Specifying a Value for the Filter Attribute

You can perform limited reconciliation by creating filters for the reconciliation module. This connector provides a Filter attribute (a scheduled task attribute) that allows you to use any of the Database Application Tables resource attributes to filter the target system records.

When you specify a value for the Filter attribute, only the target system records that match the filter criterion are reconciled into Oracle Identity Governance. If you do not specify a value for the Filter attribute, then all the records in the target system are reconciled into Oracle Identity Governance.

You specify a value for the Filter attribute while configuring the user reconciliation scheduled job.

For detailed information about Filters, see ICF Filter Syntax in *Oracle Fusion Middleware Developing and Customizing Applications for Oracle Identity Governance*.

5.1.2.2 Specifying a Value for the customizedQuery Parameter

If you want to filter values that are being retrieved from different tables by using native SQL queries, then use the customizedQuery property to configure limited reconciliation. You can configure limited reconciliation by specifying a value for either the customizedQuery property in the DBATConfiguration.groovy file or customizedQuery IT resource parameter.

You must specify a WHERE clause specifying the subset of newly added or modified records that you want to reconcile as the value of the customizedQuery parameter. For example, specifying the following WHERE clause as the value of the customizedQuery parameter returns all user records whose first name is John:

```
WHERE FIRST_NAME='JOHN'
```

The following is another example of a WHERE clause that returns all user records whose location contains "land":

```
WHERE LOCATION LIKE '%LAND'
```

> **Note:**
>
> If you are configuring limited reconciliation by using the customizedQuery property, then first test the query by running it on a staging server to ensure that data in the production server is altered as desired.

5.2 Configuring Provisioning

Learn about performing provisioning operations in Oracle Identity Governance and the guidelines that you must apply while performing these operations.

- Guidelines on Performing Provisioning Operations
- Performing Provisioning Operations

5.2.1 Guidelines on Performing Provisioning Operations

These guidelines provide information on what to do when performing provisioning operations.

For a Create User provisioning operation, you must specify a value for the User Name field. For example, John Doe. It is a mandatory field.

5.2.2 Performing Provisioning Operations

You create a new user in Identity Self Service by using the Create User page. You provision or request for accounts on the Accounts tab of the User Details page.

To perform provisioning operations in Oracle Identity Governance:

1. Log in to Identity Self Service.
2. Create a user as follows:
   a. In Identity Self Service, click **Manage**. The Home tab displays the Manage options. Click **Users**. The Manage Users page is displayed.
   b. From the Actions menu, select **Create**. Alternatively, you can click **Create** on the toolbar. The Create User page is displayed with input fields for user profile attributes.
   c. Enter details of the user in the Create User page.
3. On the Account tab, click **Request Accounts.**
4. In the Catalog page, search for and add to cart the application instance created for the connector that you created earlier, and then click **Checkout.**
5. Specify values for fields in the application form, and then click **Ready to Submit**.
6. Click **Submit.**

> **Note:**
>
> See Creating a User in *Oracle Fusion Middleware Performing Self Service Tasks with Oracle Identity Governance* for information about the fields on the Create User page.

5.3 Configuring Reconciliation Jobs

Configure reconciliation jobs to perform reconciliation runs that check for new information on your target system periodically and replicates the data in Oracle Identity Governance.

You can apply this procedure to configure the reconciliation jobs for users and entitlements.

To configure a reconciliation job:

1. Log in to Identity System Administration.
2. In the left pane, under System Management, click **Scheduler**.
3. Search for and open the scheduled job as follows:
   a. In the Search field, enter the name of the scheduled job as the search criterion. Alternatively, you can click **Advanced Search** and specify the search criterion.
   b. In the search results table on the left pane, click the scheduled job in the Job Name column.
4. On the Job Details tab, you can modify the parameters of the scheduled task:
   - **Retries**: Enter an integer value in this field. This number represents the number of times the scheduler tries to start the job before assigning the Stopped status to the job.
   - **Schedule Type**: Depending on the frequency at which you want the job to run, select the appropriate schedule type. See Creating Jobs in *Oracle Fusion Middleware Administering Oracle Identity Governance*.

   In addition to modifying the job details, you can enable or disable a job.
5. On the **Job Details** tab, in the Parameters region, specify values for the attributes of the scheduled task.

   > **Note:**
   >
   > Values (either default or user-defined) must be assigned to all the attributes. If even a single attribute value is left empty, then reconciliation is not performed.

6. Click **Apply** to save the changes.

> **Note:**
>
> You can use the Scheduler Status page in Identity System Administration to either start, stop, or reinitialize the scheduler.

5.4 Uninstalling the Connector

Uninstalling the connector deletes all the account-related data associated with its resource objects.

If you want to uninstall the connector for any reason, then run the Uninstall Connector utility. Before you run this utility, ensure that you set values for `ObjectType` and `ObjectValues` properties in the ConnectorUninstall.properties file. For example, if you want to delete resource objects, scheduled tasks, and scheduled jobs associated with the connector, then enter `"ResourceObject", "ScheduleTask", "ScheduleJob"` as the value of the ObjectType property and a semicolon-separated list of object values corresponding to your connector (for example, `Databasetable User; Databasetable Group`) as the value of the `ObjectValues` property.

> **Note:**
>
> If you set values for the `ConnectorName` and `Release` properties along with the `ObjectType` and `ObjectValue` properties, then the deletion of objects listed in the `ObjectValues` property is performed by the utility and the Connector information is skipped.

For more information, see Uninstalling Connectors in *Oracle Fusion Middleware Administering Oracle Identity Governance*.

6

Extending the Functionality of the Database Application Tables Connector

You can extend the functionality of the connector to address your specific business requirements.

This chapter contains the following topics:

6.1 Configuring Transformation and Validation of Data

Configure transformation and validation of user account data by writing Groovy script logic while creating your application.

You can configure transformation of reconciled single-valued user data according to your requirements. For example, you can use First Name and Last Name values to create a value for the Full Name field in Oracle Identity Governance.

Similarly, you can configure validation of reconciled and provisioned single-valued data according to your requirements. For example, you can validate data fetched from the First Name attribute to ensure that it does not contain the number sign (#). In addition, you can validate data entered in the First Name field on the process form so that the number sign (#) is not sent to the target system during provisioning operations.

To configure transformation or validation of user account data, you must write Groovy scripts while creating your application. For more information about writing Groovy script-based validation and transformation logic, see Validation and Transformation of Provisioning and Reconciliation Attributes of *Oracle Fusion Middleware Performing Self Service Tasks with Oracle Identity Governance*.

6.2 Configuring Action Scripts

You can configure **Action Scripts** by writing your own Groovy scripts while creating your application.

These scripts can be configured to run before or after the create, update, or delete an account provisioning operations. For example, you can configure a script to run before every user creation operation.

For information on adding or editing action scripts, see Updating the Provisioning Configuration in *Oracle Fusion Middleware Performing Self Service Tasks with Oracle Identity Governance*.

6.3 Configuring the Connector for Multiple Installations of the Target System

You must create copies of configurations of your base application to configure it for multiple installations of the target system.

The following example illustrates this requirement:
The London and New York offices of Example Multinational Inc. have their own installations of the target system, including independent schema for each. The company has recently installed Oracle Identity Governance, and they want to configure it to link all the installations of the target system.

To meet the requirement posed by such a scenario, you must clone your application which copies all configurations of the base application into the cloned application. For more information about cloning applications, see Cloning Applications in *Oracle Fusion Middleware Performing Self Service Tasks with Oracle Identity Governance*.

7

Defining and Upgrading the DBAT Connector

Define and upgrade the Database Application Tables connector using Oracle Identity System Administration.

7.1 Defining the Connector

Using Oracle Identity System Administration, you can define a customized or reconfigured connector. Defining a connector is equivalent to registering the connector with Oracle Identity Governance.

A connector is automatically defined when you install it by using the Install Connectors feature or when you upgrade it using the Upgrade Connectors feature. You must manually define a connector if:

- You import the connector by using the Deployment Governance.
- You customize or reconfigure the connector.
- You upgrade Oracle Identity Governance.

The following events take place when you define a connector:

- A record representing the connector is created in the Oracle Identity Governance database. If this record already exists, then it is updated.
- The status of the newly defined connector is set to Active. In addition, the status of a previously installed release of the same connector is automatically set to Inactive.

See Defining Connectors With Oracle Identity Manager in *Oracle Fusion Middleware Administering Oracle Identity Governance* for detailed information about the procedure to define connectors.

7.2 Upgrading the Connector

If you have already deployed the 11.1.1.6.0 version of the DBAT connector, then you can upgrade the connector to version 12.2.1.3.0 by uploading the new connector JAR files to the Oracle Identity Governance database.

> **Note:**
>
> Before you perform the upgrade procedure:
>
> - Create a backup of the Oracle Identity Governance database. See the database documentation for information about creating a backup.
> - As a best practice, initially perform the upgrade procedure in a test environment.

The following topics describe the procedure to upgrade the connector:

- Upgrade Steps
- Postupgrade Steps

> **Note:**
>
> See Upgrading Connectors in *Oracle Fusion Middleware Administering Oracle Identity Governance* for detailed information about upgrade steps.

7.2.1 Upgrade Steps

This is a summary of the procedure to upgrade the connector for both staging and production environments.

Depending on the environment in which you are upgrading the connector, perform one of the following steps:

- Staging environment: Perform the upgrade procedure by using the wizard mode.

  > **Note:**
  >
  > Do not upgrade IT resource type definition. In order to retain the default setting, you must map the IT resource definition to 'None'.

- Production environment: Perform the upgrade procedure by using the silent mode.

7.2.2 Postupgrade Steps

Postupgrade steps involve uploading new connector JAR to Oracle Identity Governance database.
Perform the following steps:

1. Delete the old Connector JARs. Run the Oracle Identity Governance Delete JARs (`$ORACLE_HOME/bin/DeleteJars.sh`) utility to delete the existing ICF bundle `org.identityconnectors.databasetable-1.2.2.jar` from the Oracle Identity Governance database.

When you run the Delete JARs utility, you are prompted to enter the login credentials of the Oracle Identity Governance administrator, URL of the Oracle Identity Governance host computer, context factory value, type of JAR file being deleted, and the name of the JAR file to be removed. Specify `4` as the value of the JAR type.

2. Upload the new connector JAR files. To do so:
   a. Run the Oracle Identity Governance Upload JARs (`$ORACLE_HOME/bin/UploadJars.sh`) utility to upload the connector JARs.
   b. Upload the `org.identityconnectors.databasetable-12.3.0.jar` bundle as an ICF Bundle. Run the Oracle Identity Governance Upload JARs utility to post the new ICF bundle `org.identityconnectors.databasetable-12.3.0.jar` file to the Oracle Identity Governance database.
   When you run the Upload JARs utility, you are prompted to enter the login credentials of the Oracle Identity Governance administrator, URL of the Oracle Identity Governance host computer, context factory value, type of JAR file being uploaded, and the location from which the JAR file is to be uploaded. Specify 4 as the value of the JAR type.
3. Restart Oracle Identity Governance.
4. If the connector is deployed on a connector server, then:
   a. Stop the connector server.
   b. Replace the existing bundle JAR file `org.identityconnectors.databasetable-1.2.2.jar` with the new bundle JAR file `org.identityconnectors.databasetable-12.3.0.jar`.
   c. Start the connector server.

8
Known Issues and Workarounds

The following are issues and workarounds associated with this release of the connector:

8.1 The Custom Schema Feature of IBM DB2 is not Supported

You can create a custom schema in the IBM DB2 database. Currently, the connector does not support custom schema and thus you cannot generate the DBAT connector using custom schema attributes of IBM DB2.

If you configure the DBATConfiguration.groovy file by creating a table (for example, VCDOG44B_SECR_SRC) using custom schema attributes and try to run the DBAT Generator, the following error is encountered:

"FINE DatabaseTableConfiguration: Get for a key MSG_INVALID_TABLE_NAME connector message Invalid table name (*TABLE_NAME*). FINE SchemaApiOp: Exception: java.lang.IndexOutOfBoundsException: Invalid table name (*TABLE_NAME*)."

As a workaround, you must configure the default or user schema that is defined in the DBATConfiguration.groovy file and then run the DBAT Generator.

8.2 Unable to Generate CI Build When DBATConfiguration.groovy File is Configured Using Data Source and JNDI Properties

While generating CI build using DBATConfiguration.groovy file configured with data source and JNDI properties, the build generation fails with the following error:

```
java.lang.UnsupportedOperationException: Remote JDBC disabled
```

The cause of this issue is missing wlfulclient.jar, which is required for build generation. This JAR file has been deprecated in Oracle Identity Governance 12c and later releases. Currently, there is no alternative JAR file available.

8.3 Connector Operations Using Connector Server Fail

When the Database Application Tables connector is configured using connector server in basic configurations and by configuring data source and JNDI properties in advance configurations, the connector operations fails with the following error:

```
java.lang.UnsupportedOperationException: Remote JDBC disabled
```

The cause of this issue is missing wlfulclient.jar, which is required to be placed in the connector server. This JAR file has been deprecated from Oracle Identity Governance 12c and later releases. Currently, there is no alternative JAR file available. However, you can use the respective JDBC URL template instead of DataSource and JNDI properties.

A

Sample Stored Procedures and Groovy Scripts

This appendix lists sample stored procedures and Groovy scripts for some of the provisioning operations. Depending on your requirement, you can either extend these stored procedures and groovy scripts or create new ones. Note that the sample stored procedures and groovy scripts listed in this appendix can be created only on an Oracle Database target system.

The appendix includes the following topics:

A.1 Sample Groovy Script for a Create Provisioning Operation

The following is a sample groovy script for performing a create provisioning operation.

Register the create script as follows:

```
import java.sql.PreparedStatement;
import org.identityconnectors.framework.common.objects.*;
import java.text.*;

// START HERE
System.out.println("[Create-Groovy] Attributes::"+attributes);

//Get all the attributes from script argument
String uid = attributes.get("__NAME__")!=null?
attributes.get("__NAME__").getValue().get(0):null;
String firstName=attributes.get("FIRSTNAME")!=null?
attributes.get("FIRSTNAME").getValue().get(0):null;
String lastName=attributes.get("LASTNAME")!=null?
attributes.get("LASTNAME").getValue().get(0):null;
String email=attributes.get("EMAIL")!=null?
attributes.get("EMAIL").getValue().get(0):null;
String description=attributes.get("DESCRIPTION")!=null?
attributes.get("DESCRIPTION").getValue().get(0):null;
salary=attributes.get("SALARY")!=null? attributes.get("SALARY").getValue().get(0):null;
joindate = attributes.get("JOININGDATE")!=null?
```

```
attributes.get("JOININGDATE").getValue().get(0):null;
enableValue = attributes.get("__ENABLE__")!=null?
attributes.get("__ENABLE__").getValue().get(0):true;

PreparedStatement createStmt = null;
try {
    //Initialize the prepare statement to insert the data into database table
    createStmt =   conn.prepareStatement("INSERT INTO
USERINFO(USERID,FIRSTNAME,LASTNAME,EMAIL,DESCRIPTION,SALARY,JOININGDATE,STATUS)
VALUES(?,?,?,?,?,?,?,?)");
    //Set the input parameters
    createStmt.setString(1, uid);
    createStmt.setString(2, firstName);
    createStmt.setString(3, lastName);
    createStmt.setString(4, email);
    createStmt.setString(5, description);
    createStmt.setBigDecimal(6, salary);
    dateStr = null;
    //Convert the joindate into oracle date format
    if( joindate != null) {
        SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd hh:mm:ss.S");
        java.util.Date date= df.parse(joindate);
        DateFormat targetFormat = new SimpleDateFormat("dd-MMM-yy");
        dateStr = targetFormat.format(date);
    }
    createStmt.setString(7,dateStr);
        if(enableValue)
        createStmt.setString(8,"Enabled");
    else
        createStmt.setString(8,"Disabled");
    //Execute sql statement
    createStmt.executeUpdate();
} finally {
    //close the sql statements
    if (createStmt != null)
        createStmt.close();
}
System.out.println("[Create] Created User::"+uid);
//Return Uid from the script
return new Uid(uid);
```

A.2 Sample Groovy Script for an Update Provisioning Operation

The following is a sample groovy script for performing an update provisioning operation.

Register the update script as follows:

```
import org.identityconnectors.framework.common.objects.*;
import java.text.*;
import org.identityconnectors.framework.common.exceptions.*;

System.out.println("[Update-Groovy] Atrributes::"+ attributes);

/** During an Update operation,OIM  sends the UID attribute along with updated
attributes.
Get all the values of attributes **/
```

```
String id = attributes.get("__UID__")!=null?
attributes.get("__UID__").getValue().get(0):null;
String firstName=attributes.get("FIRSTNAME")!=null?
attributes.get("FIRSTNAME").getValue().get(0):null;
String lastName=attributes.get("LASTNAME")!=null?
attributes.get("LASTNAME").getValue().get(0):null;
String email=attributes.get("EMAIL")!=null?
attributes.get("EMAIL").getValue().get(0):null;
String description=attributes.get("DESCRIPTION")!=null?
attributes.get("DESCRIPTION").getValue().get(0):null;
salary=attributes.get("SALARY")!=null? attributes.get("SALARY").getValue().get(0):null;
joindate = attributes.get("JOININGDATE")!=null?
attributes.get("JOININGDATE").getValue().get(0):null;
status = attributes.get("STATUS")!=null?
attributes.get("STATUS").getValue().get(0):null;
enableValue = attributes.get("__ENABLE__")!=null?
attributes.get("__ENABLE__").getValue().get(0):true;

//Throw exception if uid is null
if(id==null) throw new ConnectorException("UID Cannot be Null");
stmt = null;
try {
//Create prepare statement to update the USERINFO table
        stmt = conn.prepareStatement("UPDATE USERINFO SET FIRSTNAME=COALESCE(?,
FIRSTNAME),LASTNAME =COALESCE(?, LASTNAME), EMAIL= COALESCE(?,
EMAIL),DESCRIPTION=COALESCE(?, DESCRIPTION),SALARY=COALESCE(?,
SALARY),JOININGDATE=COALESCE(to_date(?,'dd-Mon-yy'), JOININGDATE),STATUS=COALESCE(?,
STATUS) WHERE USERID =?");
        //Set sql input parameters
        stmt.setString(1, firstName);
        stmt.setString(2, lastName);
        stmt.setString(3, email);
        stmt.setString(4, description);
        stmt.setBigDecimal(5, salary);
        dateStr = null;
        //Convert the joindate into oracle date format
        if( joindate != null) {
            SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd hh:mm:ss.S");
            java.util.Date date= df.parse(joindate);
            DateFormat targetFormat = new SimpleDateFormat("dd-MMM-yy");
            dateStr = targetFormat.format(date); }
        stmt.setString(6,dateStr);
        if(enableValue)
            stmt.setString(7,"Enabled");
        else
            stmt.setString(7,"Disabled");
        stmt.setString(8, id);
        stmt.executeUpdate();
} finally {
    if (stmt != null)
        stmt.close();
};
System.out.println("[Update] Updated user::"+ id);
return new Uid(id);
```

A.3 Sample Groovy Script for a Delete Provisioning Operation

The following is a sample groovy script for performing a delete provisioning operation.

Register the delete script as follows:

```
import java.sql.PreparedStatement;
import org.identityconnectors.framework.common.objects.*;

//Get the UID from the input map 'attributes'
String uid = attributes.get("__UID__").getValue().get(0);
System.out.println("[Delete-Groovy] Deleting user:: "+ uid);

try {
    //Delete data from child tables and then, main table
    //Delete user roles
    st = conn.prepareStatement("DELETE FROM USER_ROLE WHERE USERID=?");
    st.setString(1, uid);
    st.executeUpdate();
    st.close();

    //Delete user groups
    st = conn.prepareStatement("DELETE FROM USER_GROUP WHERE USERID=?");
    st.setString(1, uid);
    st.executeUpdate();
    st.close();

    //Delete user account
    st = conn.prepareStatement("DELETE FROM USERINFO WHERE USERID=?");
    st.setString(1, uid);
    st.executeUpdate();
} finally {
    if (st != null)
        st.close(); };
System.out.println("Deleted user:: "+ uid);
```

A.4 Sample Groovy Script for an Add Child Data Provisioning Operation

The following is a sample groovy script for adding child data.

Register the add child data script as follows:

```
import org.identityconnectors.framework.common.objects.*;
import java.text.*;

System.out.println("[addMultiValuedAttributeScript-Groovy] Adding Child data::"+
attributes);
childst =null;
try {
    //Adding Group data
    childDataEOSet = null;

    /**The child attributes are returned as a set of embedded objects. Each
Embedded    object will provide a row of data in the child table.
    For example, if DBAT contains USER_GROUP as a child in OIM and contains two
rows of groups data then, we will get a set of embedded objects with count 2 and
each embedded    object represents a row in child data.
    This groovy script is based on a child table named USER_GROUP and containing
USERID, GROUP_ID  as its columns.**/

    if(attributes.get("USER_GROUP")!=null)
    {
        childDataEOSet=attributes.get("USER_GROUP").getValue();
        childst = conn.prepareStatement("INSERT INTO USER_GROUP VALUES (?,?)");
```

```
        String id = attributes.get("__UID__").getValue().get(0);
        if(childDataEOSet !=null){
            //Iterate through child data and insert into table
            System.out.println("[addMultiValuedAttributeScript] Adding Group data.");
            for( iterator = childDataEOSet.iterator(); iterator.hasNext(); )
            {
                eo = iterator.next();
                attrsSet = eo.getAttributes();
                grpattr=AttributeUtil.find("GROUPID",attrsSet);
                if(grpattr!=null){
                    groupid=grpattr.getValue().get(0);
                    childst.setString(1, id);
                    childst.setString(2, groupid);
                    childst.executeUpdate();
                    childst.clearParameters();
                } };
        } } } finally {
    if (childst != null)
    childst.close();
};

try {
    //Adding Role data
    childDataEOSet = null;
    if(attributes.get("USER_ROLE")!=null){
        SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd hh:mm:ss.S");
        DateFormat targetFormat = new SimpleDateFormat("dd-MMM-yy");
        childDataEOSet=attributes.get("USER_ROLE").getValue();
        childst = conn.prepareStatement("INSERT INTO USER_ROLE VALUES (?,?,?,?)");
        String id = attributes.get("__UID__").getValue().get(0);
        if(childDataEOSet !=null){
            System.out.println("[addMultiValuedAttributeScript] Adding Role data.");
            for( iterator = childDataEOSet.iterator(); iterator.hasNext(); ) {
                eo = iterator.next();
                attrsSet = eo.getAttributes();
                roleattr=AttributeUtil.find("ROLEID",attrsSet);
                fromdateAttr=AttributeUtil.find("FROMDATE",attrsSet);
                todateAttr=AttributeUtil.find("TODATE",attrsSet);
                                if(roleattr!=null){
                    roleid=roleattr.getValue().get(0);
                    childst.setString(1, id);
                    childst.setString(2, roleid);
                    fromdate = null;
                    if(fromdateAttr!= null)
                    {
                        java.util.Date date= df.parse(fromdateAttr.getValue().get(0));
                        fromdate = targetFormat.format(date);
                    }
                    childst.setString(3, fromdate);
                    todate = null;
                    if(todateAttr!= null)
                    {
                        java.util.Date date= df.parse(todateAttr.getValue().get(0));
                        todate = targetFormat.format(date);
                    }
                    childst.setString(4, todate);
                    childst.executeUpdate();
                    childst.clearParameters();
                } };
    } } } finally {
    if (childst != null)
```

```
    childst.close();
};
```

A.5 Sample Stored Procedure and Groovy Script for a Delete Child Data Provisioning Operation

The following is a sample groovy script for deleting child data.

The delete child data procedure is called as follows:

```
delSt= conn.prepareCall("{call DELETE_USERGROUP(?,?)}");

delSt= conn.prepareCall("{call DELETE_USERROLE(?,?)}");
```

The procedure for `DELETE_USERGROUP` is as follows:

```
create or replace PROCEDURE DELETE_USERGROUP
(  userin IN VARCHAR2, gId IN VARCHAR2
) AS
BEGIN
DELETE from USER_GROUP where USERID=userin and GROUPID=gId;
END DELETE_USERGROUP;
```

The procedure for `DELETE_USERROLE` is as follows:

```
create or replace PROCEDURE DELETE_USERROLE
(  userin IN VARCHAR2, rId IN VARCHAR2
) AS
BEGIN
DELETE  from USER_ROLE where USERID=userin and ROLEID=rId;
END DELETE_USERROLE;
```

Register the delete child data script as follows:

```
import org.identityconnectors.framework.common.objects.*;
System.out.println("[removeMultiValuedAttributeScript] Removing Child data::"+
attributes);

try {
    childDataEOSet = null;
    delSt = null;
    //Get UID
    String id = attributes.get("__UID__").getValue().get(0);
    if(attributes.get("USER_GROUP")!=null)
    {
        childDataEOSet=attributes.get("USER_GROUP").getValue();
        //Delete child data using stored procedure
        delSt= conn.prepareCall("{call DELETE_USERGROUP(?,?)}");
        if(childDataEOSet !=null){
            System.out.println("[removeMultiValuedAttributeScript] Removing
Group data.");
            //Iterate through child data and delete
            for( iterator = childDataEOSet.iterator(); iterator.hasNext(); )
            {
                eo = iterator.next();
                attrsSet = eo.getAttributes();
                grpattr=AttributeUtil.find("GROUPID",attrsSet);
                if(grpattr!=null){
                    groupid=grpattr.getValue().get(0);
```

```
                    delSt.setString(1, id);
                    delSt.setString(2, groupid);
                    delSt.executeUpdate();
                    System.out.println("[removeMultiValuedAttributeScript] Deleted
Group::"+ grpattr);
                } }; } }
} finally {
    if (delSt != null)
    delSt.close();
};
try {
    childDataEOSet = null;
    delSt = null;
    String id      = attributes.get("__UID__").getValue().get(0);
    if(attributes.get("USER_ROLE")!=null)
    {
        childDataEOSet=attributes.get("USER_ROLE").getValue();
        delSt= conn.prepareCall("{call DELETE_USERROLE(?,?)}");
        if(childDataEOSet !=null){
            System.out.println("[removeMultiValuedAttributeScript] Removing Role
data.");
            for( iterator = childDataEOSet.iterator(); iterator.hasNext(); )
            {
                    eo = iterator.next();
                attrsSet = eo.getAttributes();
                roleattr=AttributeUtil.find("ROLEID",attrsSet);
                if(roleattr!=null){
                    rolename=roleattr.getValue().get(0);
                    delSt.setString(1, id);
                    delSt.setString(2, rolename);
                    delSt.executeUpdate();
                    System.out.println("[removeMultiValuedAttributeScript] Deleted
Role::"+ rolename);
                }             };         }     }
} finally {
    if (delSt != null)
        delSt.close();
};
```

A.6 Sample Stored Procedure and Groovy Script for Lookup Field Synchronization

The following is a sample groovy script for performing lookup field synchronization.

The Lookup field procedures are called as follows:

```
st = conn.prepareCall("{call GET_ROLES(?)}");

st = conn.prepareCall("{call GET_GROUPS(?)}");
```

The procedure for `GET_ROLES` is as follows:

```
create or replace PROCEDURE GET_ROLES
( user_cursor OUT TYPES.cursorType
) AS
BEGIN
OPEN user_cursor FOR
SELECT ROLENAME,ROLEID from ROLES;
END GET_ROLES;
```

The procedure for `GET_GROUPS` is as follows:

```
create or replace PROCEDURE GET_GROUPS
( user_cursor OUT TYPES.cursorType
) AS
BEGIN
OPEN user_cursor FOR
SELECT GROUPNAME,GROUPID from GROUPS;
END GET_GROUPS;
```

Register the lookup field synchronization script as follows:

```
import org.identityconnectors.framework.common.objects.*;
rs = null;
st = null;
try {
    System.out.println("[Lookup] Lookup Recon timing::"+ timing);
    System.out.println("[Lookup] Attributes to Get::"+ ATTRS_TO_GET);
    // This script is common for all lookups. Read the timing ( input) and
return the data accordingly
    // The format of timing is : executeQuery:<objectclass>
        String codekey =  ATTRS_TO_GET[0];
    String decodekey = ATTRS_TO_GET[1];
    if( timing.equals("executeQuery:Role"))
    {
        System.out.println("[Lookup] Getting Roles.");
        st = conn.prepareCall("{call GET_ROLES(?)}");
    }
    else
    {
        System.out.println("[Lookup] Getting Groups.");
        st = conn.prepareCall("{call GET_GROUPS(?)}"); }
    st.registerOutParameter(1, oracle.jdbc.driver.OracleTypes.CURSOR);
    st.execute();
    rs = st.getObject(1);
    while (rs.next()) {
        cob = new ConnectorObjectBuilder();
        Attribute codeattr= AttributeBuilder.build(decodekey,rs.getString(2));
        Attribute decodeattr= AttributeBuilder.build(codekey,rs.getString(1));
        cob.addAttribute(codeattr);
    cob.addAttribute(decodeattr);
        cob.setUid(rs.getString(2));
        cob.setName(rs.getString(2));
        handler.handle(cob.build());
    } } finally {
    if( null != rs)
        rs.close();
    if( null != st)
        st.close();
}
```

A.7 Sample Stored Procedure and Groovy Script for Full or Filter Reconciliation

The following is a sample groovy script for performing full or filter reconciliation.

The full reconciliation procedure is called as follows:

```
st = conn.prepareCall("{call EXECUTE_QUERY(?)}");
```

The filtered reconciliation procedure is called as follows:

```
st = conn.prepareCall("{call EXECUTE_QUERY_WITH_FILTER(?,?,?)}");
```

The get user role procedure is called as follows:

```
roleStmt = conn.prepareCall("{call GET_USERROLE(?,?)}");
```

The get user group procedure is called as follows:

```
groupStmt = conn.prepareCall("{call GET_USERGROUP(?,?)}");
```

The procedure for `EXECUTE_QUERY` is as follows:

```
create or replace PROCEDURE EXECUTE_QUERY
( user_cursor OUT TYPES.cursorType
) AS
BEGIN
OPEN user_cursor FOR
SELECT USERINFO.USERID, USERINFO.FIRSTNAME , USERINFO.LASTNAME,
USERINFO.EMAIL ,USERINFO.DESCRIPTION,USERINFO.SALARY,USERINFO.JOININGDATE ,USERINFO.STA
TUS FROM USERINFO;
END EXECUTE_QUERY;
```

The procedure for `EXECUTE_QUERY_WITH_FILTER` is as follows:

```
create or replace PROCEDURE EXECUTE_QUERY_WITH_FILTER
( user_cursor OUT TYPES.cursorType, columnName IN VARCHAR2, columnValue IN VARCHAR2
) AS
BEGIN
open user_cursor for 'SELECT USERINFO.USERID, USERINFO.FIRSTNAME , USERINFO.LASTNAME,
USERINFO.EMAIL ,USERINFO.DESCRIPTION,USERINFO.SALARY,USERINFO.JOININGDATE ,USERINFO.STA
TUS FROM USERINFO USERINFO where '|| columnName ||' like '''||columnValue||'''';
END EXECUTE_QUERY_WITH_FILTER;
```

The procedure for `GET_USERROLE` is as follows:

```
create or replace PROCEDURE GET_USERROLE
( user_cursor OUT TYPES.cursorType, userin IN VARCHAR2
) AS
BEGIN
OPEN user_cursor FOR
SELECT ROLEID,FROMDATE,TODATE from USER_ROLE where USERID=userin;
END GET_USERROLE;
```

The procedure for `GET_USERGROUP` is as follows:

```
create or replace PROCEDURE GET_USERGROUP
( user_cursor OUT TYPES.cursorType, userin IN VARCHAR2
) AS
BEGIN
OPEN user_cursor FOR
SELECT GROUPID from USER_GROUP where USERID=userin;
END GET_USERGROUP;
```

Register the full or filtered reconciliation script as follows:

```
import org.identityconnectors.framework.common.objects.*;
import java.lang.reflect.*;
import java.lang.String;
import org.identityconnectors.common.security.GuardedString;
import java.text.*;
```

```
rs = null;
st = null;
try {
    if( filterString != "")
    {
        System.out.println("[Execute Query] Performing Recon with Filter. Filter
is::"+ filterString+" And Filer Params are::"+filterParams);
        String[] filter = filterParams.get(0).split(":");
        st = conn.prepareCall("{call EXECUTE_QUERY_WITH_FILTER(?,?,?)}");
        st.setString(2, filter[0]);
        st.setString(3, filter[1]);
    }
    else
    {
        System.out.println("[Execute Query] Performing Full Recon.");
        st = conn.prepareCall("{call EXECUTE_QUERY(?)}");
}
    st.registerOutParameter(1, oracle.jdbc.driver.OracleTypes.CURSOR);
    st.execute();
    rs = st.getObject(1);
    SimpleDateFormat targetFormat = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss
z");
    DateFormat df = new SimpleDateFormat("yyyy-MM-dd");

    while (rs.next()) {
        cob = new ConnectorObjectBuilder();
        cob.setObjectClass(ObjectClass.ACCOUNT);
        Attribute fname= AttributeBuilder.build(new
String("FIRSTNAME"),rs.getString(2));
        Attribute lname= AttributeBuilder.build(new
String("LASTNAME"),rs.getString(3));
        Attribute uid= AttributeBuilder.build(new
String("__UID__"),rs.getString(1));
        Attribute name= AttributeBuilder.build(new
String("__NAME__"),rs.getString(1));
        Attribute email= AttributeBuilder.build(new
String("EMAIL"),rs.getString(4));
        Attribute salary= AttributeBuilder.build(new
String("SALARY"),rs.getBigDecimal(6));
        Attribute description= AttributeBuilder.build(new
String("DESCRIPTION"),rs.getString(5));
        dbDate = rs.getDate(7);
        joinDateStr = null;
        if( null != dbDate)
        {
            java.util.Date date= df.parse(dbDate.toString());
            joinDateStr = targetFormat.format(date);
        }
        Attribute joindate= AttributeBuilder.build(new
String("JOININGDATE"),joinDateStr);
        Attribute status= AttributeBuilder.build(new
String("STATUS"),rs.getString(8));
                cob.addAttribute(fname);
        cob.addAttribute(lname);
        cob.addAttribute(uid);
        cob.addAttribute(name);
        cob.addAttribute(email);
        cob.addAttribute(salary);
        cob.addAttribute(description);
        cob.addAttribute(joindate);
```

```
        cob.addAttribute(status);
        roleStmt = conn.prepareCall("{call GET_USERROLE(?,?)}");
        roleStmt.registerOutParameter(1, oracle.jdbc.driver.OracleTypes.CURSOR);
    roleStmt.setString(2, rs.getString(1));
    roleStmt.execute();
        roleResultSet = roleStmt.getObject(1);
        java.util.List<EmbeddedObject> eoList = new ArrayList<EmbeddedObject>();
        while (roleResultSet.next()) {
            Attribute roleId= AttributeBuilder.build(new
String("ROLEID"),roleResultSet.getString(1));
                        dbDate = roleResultSet.getDate(2);
            fromDateStr = null;
            if( null != dbDate)
            {
                java.util.Date date= df.parse(dbDate.toString());
                fromDateStr = targetFormat.format(date);
            }
            dbDate = roleResultSet.getDate(2);
            toDateStr = null;
            if( null != dbDate)
            {
                java.util.Date date= df.parse(dbDate.toString());
                toDateStr = targetFormat.format(date);
            }

            Attribute fromdate= AttributeBuilder.build(new
String("FROMDATE"),fromDateStr);
            Attribute todate= AttributeBuilder.build(new String("TODATE"),toDateStr);
                    EmbeddedObjectBuilder roleEA = new EmbeddedObjectBuilder();
            roleEA.addAttribute(roleId);
            roleEA.addAttribute(fromdate);
            roleEA.addAttribute(todate);
            roleEA.setObjectClass(new ObjectClass("USER_ROLE"));
            eoList.add(roleEA.build());
        }
        roleResultSet.close();
                roleStmt.close();
        EmbeddedObject[] roleEm = eoList.toArray(new EmbeddedObject[eoList.size()]);
cob.addAttribute(AttributeBuilder.build("USER_ROLE", (Object[]) roleEm));
        groupStmt = conn.prepareCall("{call GET_USERGROUP(?,?)}");
        groupStmt.registerOutParameter(1, oracle.jdbc.driver.OracleTypes.CURSOR);
    groupStmt.setString(2, rs.getString(1));
    groupStmt.execute();
        groupResultSet = groupStmt.getObject(1);
        java.util.List<EmbeddedObject> geoList = new ArrayList<EmbeddedObject>();
        while (groupResultSet.next()) {
            Attribute groupId= AttributeBuilder.build(new
String("GROUPID"),groupResultSet.getString(1));
            EmbeddedObjectBuilder groupEA = new EmbeddedObjectBuilder();
            groupEA.addAttribute(groupId);
            groupEA.setObjectClass(new ObjectClass("USER_GROUP"));
            geoList.add(groupEA.build());
        }
                groupResultSet.close();
        groupStmt.close();
        EmbeddedObject[] groupEm = geoList.toArray(new EmbeddedObject[geoList.size()]);
cob.addAttribute(AttributeBuilder.build("USER_GROUP", (Object[]) groupEm));

        if(!handler.handle(cob.build())) return;
    } } finally {
    if( null != rs)
```

```
    rs.close();
    if( null != st)
    st.close();
}
```

A.8 Sample Stored Procedure and Groovy Script for Incremental Reconciliation

The following is a sample groovy script for performing incremental reconciliation.

The incremental reconciliation procedure is called as follows:

```
st = conn.prepareCall("{call EXECUTE_QUERY_INCREMENTAL(?,?,?)}");
```

The get user role procedure is called as follows:

```
roleStmt = conn.prepareCall("{call GET_USERROLE(?,?)}");
```

The get user group procedure is called as follows:

```
groupStmt = conn.prepareCall("{call GET_USERGROUP(?,?)}");
```

The procedure for `EXECUTE_QUERY_INCREMENTAL` is as follows:

```
create or replace PROCEDURE EXECUTE_QUERY_INCREMENTAL
( user_cursor OUT TYPES.cursorType, columnName IN VARCHAR2, columnValue IN
VARCHAR2
) AS
BEGIN
if columnValue is NULL then
open user_cursor for 'SELECT
USERID,FIRSTNAME,LASTNAME,EMAIL,DESCRIPTION,SALARY,JOININGDATE,STATUS,
to_char(LASTUPDATED) FROM USERINFO';
else
open user_cursor for 'SELECT
USERID,FIRSTNAME,LASTNAME,EMAIL,DESCRIPTION,SALARY,JOININGDATE,STATUS,
to_char(LASTUPDATED) FROM USERINFO where '|| columnName ||' >  to_timestamp
('''||columnValue||''')';
end if;
END EXECUTE_QUERY_INCREMENTAL;
```

The procedure for `GET_USERROLE` is as follows:

```
create or replace PROCEDURE GET_USERROLE
( user_cursor OUT TYPES.cursorType, userin IN VARCHAR2
) AS
BEGIN
OPEN user_cursor FOR
SELECT ROLEID,FROMDATE,TODATE from USER_ROLE where USERID=userin;
END GET_USERROLE;
```

The procedure for `GET_USERGROUP` is as follows:

```
create or replace PROCEDURE GET_USERGROUP
( user_cursor OUT TYPES.cursorType, userin IN VARCHAR2
) AS
BEGIN
OPEN user_cursor FOR
SELECT GROUPID from USER_GROUP where USERID=userin;
END GET_USERGROUP;
```

Register the incremental reconciliation script as follows:

```
import org.identityconnectors.framework.common.objects.*;
import java.lang.reflect.*;
import org.identityconnectors.common.security.GuardedString;
import java.text.*;
import java.lang.String;
rs = null;
st = null;
try {
System.out.println("[Sync] Performing Incremental Recon.");
System.out.println("[Sync] Sync Attribute::"+syncattribute);
System.out.println("[Sync] Sync token:: "+synctoken);
st = conn.prepareCall("{call EXECUTE_QUERY_INCREMENTAL(?,?,?)}");
st.setString(2, syncattribute);
st.setString(3, synctoken!=null? synctoken.getValue():null);
st.registerOutParameter(1, oracle.jdbc.driver.OracleTypes.CURSOR);
st.execute();
rs = st.getObject(1);
SimpleDateFormat targetFormat = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss z");
DateFormat df = new SimpleDateFormat("yyyy-MM-dd");
while (rs.next()) {
        cob = new ConnectorObjectBuilder();
        cob.setObjectClass(ObjectClass.ACCOUNT);
        Attribute fname= AttributeBuilder.build(new
String("FIRSTNAME"),rs.getString(2));
        Attribute lname= AttributeBuilder.build(new
String("LASTNAME"),rs.getString(3));
        Attribute uid= AttributeBuilder.build(new String("__UID__"),rs.getString(1));
        Attribute name= AttributeBuilder.build(new String("__NAME__"),rs.getString(1));
        Attribute email= AttributeBuilder.build(new String("EMAIL"),rs.getString(4));
        Attribute salary= AttributeBuilder.build(new
String("SALARY"),rs.getBigDecimal(6));
        Attribute description= AttributeBuilder.build(new
String("DESCRIPTION"),rs.getString(5));
        dbDate = rs.getDate(7);
        joinDateStr = null;
        if( null != dbDate)
        {
            java.util.Date date= df.parse(dbDate.toString());
            joinDateStr = targetFormat.format(date);
        }
        Attribute joindate= AttributeBuilder.build(new
String("JOININGDATE"),joinDateStr);
        Attribute status= AttributeBuilder.build(new String("STATUS"),rs.getString(8));
                cob.addAttribute(fname);
        cob.addAttribute(lname);
        cob.addAttribute(uid);
        cob.addAttribute(name);
        cob.addAttribute(email);
        cob.addAttribute(salary);
        cob.addAttribute(description);
        cob.addAttribute(joindate);
        cob.addAttribute(status);
        roleStmt = conn.prepareCall("{call GET_USERROLE(?,?)}");
        roleStmt.registerOutParameter(1, oracle.jdbc.driver.OracleTypes.CURSOR);
    roleStmt.setString(2, rs.getString(1));
    roleStmt.execute();
        roleResultSet = roleStmt.getObject(1);
        java.util.List<EmbeddedObject> eoList = new ArrayList<EmbeddedObject>();
        while (roleResultSet.next()) {
```

```
            Attribute roleId= AttributeBuilder.build(new
String("ROLEID"),roleResultSet.getString(1));
            dbDate = roleResultSet.getDate(2);
            fromDateStr = null;
            if( null != dbDate)
            {
                java.util.Date date= df.parse(dbDate.toString());
                fromDateStr = targetFormat.format(date);
            }
            dbDate = roleResultSet.getDate(2);
            toDateStr = null;
            if( null != dbDate)
            {
                java.util.Date date= df.parse(dbDate.toString());
                toDateStr = targetFormat.format(date);
            }
    Attribute fromdate= AttributeBuilder.build(new
String("FROMDATE"),fromDateStr);
            Attribute todate= AttributeBuilder.build(new
String("TODATE"),toDateStr);
                    EmbeddedObjectBuilder roleEA = new EmbeddedObjectBuilder();
            roleEA.addAttribute(roleId);
            roleEA.addAttribute(fromdate);
            roleEA.addAttribute(todate);
            roleEA.setObjectClass(new ObjectClass("USER_ROLE"));
            eoList.add(roleEA.build());
        }
        roleResultSet.close();
                roleStmt.close();
        EmbeddedObject[] roleEm = eoList.toArray(new
EmbeddedObject[eoList.size()]);
cob.addAttribute(AttributeBuilder.build("USER_ROLE", (Object[]) roleEm));
                groupStmt = conn.prepareCall("{call GET_USERGROUP(?,?)}");
        groupStmt.registerOutParameter(1, oracle.jdbc.driver.OracleTypes.CURSOR);
    groupStmt.setString(2, rs.getString(1));
    groupStmt.execute();
        groupResultSet = groupStmt.getObject(1);
        java.util.List<EmbeddedObject> geoList = new ArrayList<EmbeddedObject>();
        while (groupResultSet.next()) {
            Attribute groupId= AttributeBuilder.build(new
String("GROUPID"),groupResultSet.getString(1));
            EmbeddedObjectBuilder groupEA = new EmbeddedObjectBuilder();
            groupEA.addAttribute(groupId);
            groupEA.setObjectClass(new ObjectClass("USER_GROUP"));
            geoList.add(groupEA.build());
        }
                groupResultSet.close();
        groupStmt.close();
        EmbeddedObject[] groupEm = geoList.toArray(new
EmbeddedObject[geoList.size()]);
cob.addAttribute(AttributeBuilder.build("USER_GROUP", (Object[]) groupEm));
        Attribute timestamp= AttributeBuilder.build(new
String("LASTUPDATED"),rs.getString(9));
        token = AttributeUtil.getSingleValue(timestamp);
        SyncToken syncToken  = new SyncToken(token);
        SyncDeltaBuilder bld = new SyncDeltaBuilder();
        bld.setObject(cob.build());
        bld.setToken(syncToken);
        bld.setDeltaType(SyncDeltaType.CREATE_OR_UPDATE);
        handler.handle(bld.build());
    } } finally {
```

```
    if( null != rs)
        rs.close();
    if( null != st)
        st.close();
}
```

A.9 Tables Used for Sample Groovy and Configuration Scripts

The tables that are used by sample groovy scripts and configuration scrips are listed below:

- Lookup tables for roles and groups:

```
create table ROLES(
roleid varchar2(50),
rolename varchar2(50));

create table GROUPS(
groupid varchar2(50),
groupname varchar2(50));
```

- Tables for user accounts:
  - Parent Table:

```
create table USERINFO(
UserId varchar2(50),
FirstName varchar2(50),
LastName varchar2(50),
email varchar2(50),
Description varchar2(50),
Salary NUMBER,
JoiningDate date,
status varchar2(50),
lastupdated timestamp,
PRIMARY KEY ( UserId ));
```

  - Child Table:

```
create table USER_ROLE(
userid varchar2(50),
roleid varchar2(50),
fromdate date,
todate date);

create table USER_GROUP(
userid varchar2(50),
groupid varchar2(50));

ALTER TABLE USER_GROUP ADD CONSTRAINT GROUP_PK PRIMARY KEY ("USERID",
"GROUPID") ENABLE;

ALTER TABLE USER_ROLE ADD CONSTRAINT ROLE_PK PRIMARY KEY ("USERID", "ROLEID")
ENABLE;
```

B
Performing Common Connector Operations

This appendix summarizes the procedure for some of the common operations that can be performed by using this connector. This appendix discusses the following topics:

- Running Incremental Trusted Source Reconciliation
- Running Incremental Target Resource Reconciliation
- Configuring and Performing Lookup Field Synchronization
- Provisioning Child Data

B.1 Running Incremental Trusted Source Reconciliation

Perform the following tasks for an incremental trusted source reconciliation run:

1. The target system (database) must have users.
2. Create the DBAT application. See Creating an Application By Using the Connector for detailed information about creating an application.

> **Note:**
>
> While creating the application, make sure to specify a value for the `changeLogColumn` property.

3. Run the *RESOURCE* Trusted Resource User Reconciliation scheduled job to perform a full trusted source reconciliation run to fetch all user records in the target system to Oracle Identity Governance. See Scheduled Jobs for Reconciliation of User Records for more information about this scheduled job.
4. Perform some changes to user records in your target system.
5. Run the *RESOURCE*Trusted Incremental Resource User Reconciliation scheduled job to perform an incremental reconciliation run to fetch only the user records that were modified in the target system since the last reconciliation run. See Scheduled Jobs for Incremental Reconciliation for more information about this scheduled job.

B.2 Running Incremental Target Resource Reconciliation

Perform the following tasks for an incremental target resource reconciliation run:

1. The target system (database) must have users.
2. Create the DBAT application. See Creating an Application By Using the Connector for detailed information about creating an application.

> **Note:**
>
> While creating the application, make sure to specify a value for the `changeLogColumn` property.

3. Run the *RESOURCE* Target Resource User Reconciliation scheduled job to perform a full target resource reconciliation run to fetch all user records in the target system to Oracle Identity Governance.

   See Scheduled Jobs for Reconciliation of User Records for more information about the scheduled job.

4. Perform some changes to user records in your target system.

5. Run the *RESOURCE* Target Incremental Resource User Reconciliation scheduled job to perform an incremental reconciliation run to fetch only the user records that were modified in the target system since the last reconciliation run. See Scheduled Jobs for Incremental Reconciliation for more information about this scheduled job.

B.3 Configuring and Performing Lookup Field Synchronization

This section describes the procedure to configure and perform lookup field synchronization to use lookup definitions as the input source for some of the fields on the process form during provisioning operations. The following are the tasks to achieve this:

1. Create the DBAT application. See Creating an Application By Using the Connector for detailed information about creating an application.
2. Ensure that you have empty lookup definitions such as Lookup.*RESOURCE*.Example to store values from child tables in your target system.
3. Update the form and lookup definition to include information that specifies the field is a lookup field.
4. Run the *RESOURCE*Target Lookup Reconciliation scheduled job to perform lookup field synchronization. See Scheduled Job for Lookup Field Synchronization for more information about this scheduled job.

B.4 Provisioning Child Data

To perform provisioning operations on child data:

1. Create the DBAT application. See Creating an Application By Using the Connector for detailed information about creating an application.
2. To provision child data, see Performing Provisioning Operations for more information.

C

Files and Directories of the DBAT Connector

This appendix lists the tables that describe the files and directories corresponding to the DBAT connector. It contains the following topics:

- Files and Directories on the Installation Media
- Files and Directories in the Generated Connector Package

C.1 Files and Directories on the Installation Media

Table C-1 describes the files and directories on the installation media.

Table C-1 Files and Directories on the Installation Media

Files in the Installation Media Directory	Description
org.identityconnectors.databasetable-12.3.0.jar	This JAR file is the ICF connector bundle.
generator/dbat-generator-12.2.1.3.0.zip	This zip file contains the DBAT generator. The DBAT generator discovers the target system schema and generates the connector package. The Connector Installer uses the XML file in this package to create connector components that are used for connector operations. The directory structure of the connector package is described in Table C-3.
Files in the resources directory	Each of these resource bundles contains language-specific information that is used by the connector. During connector deployment, this file is copied to the Oracle Identity Governance database. **Note**: A **resource bundle** is a file containing localized versions of the text strings that include GUI element labels and messages.

Table C-2 describes the files and directories in the dbat-generator-12.2.1.3.0.zip file.

Table C-2 Files and Directories in the dbat-generator-12.2.1.3.0.zip File

Files and Directories in the dbat-generator-12.2.1.3.0.zip File	Description
bin/classpath.cmd bin/classpath-append.cmd	These files contain the commands that add the JAR files (located in the lib directory) to the classpath on Microsoft Windows.
bin/DBATGenerator.cmd bin/DBATGenerator.sh	This file contains commands to run the DBAT generator:Note that the .cmd file is the Microsoft Windows version of the DBAT Generator. Similarly, the .sh file is the UNIX version of the DBAT Generator.

Table C-2 (Cont.) Files and Directories in the dbat-generator-12.2.1.3.0.zip File

Files and Directories in the dbat-generator-12.2.1.3.0.zip File	Description
bin/logging.properties	This file contains the default logging configurations of the DBAT generator.
lib/connector-framework-internal	This JAR files contains class files that implement ICF.
lib/connector-framework	This JAR file contains class files that define the ICF Application Programming Interface (API). This API is used communicate between Oracle Identity Governance and this connector.
lib/dbat-generator-oim-integration	This JAR file contains the class files of the DBAT generator.
lib/groovy-all	This JAR file contains the groovy libraries required for running the DBAT generator.
lib/org.identityconnectors.databasetable-12.3.0.jar	This JAR file is the Identity Connector bundle. During connector installation, this file is copied to the Oracle Identity Governance database.
resources/DBATConfiguration.groovy	This file contains properties that store basic information about the target system schema, which is used for configuring your target system either as a trusted source or target resource. In addition, it stores information about the manner in which the connector must connect to the target system.
xml/DBAT-auth-template.xml	This file contains definitions for the connector objects required for creating an authoritative application. It includes certain details required to connect Oracle Identity Governance with the target system. It also includes configuration details specific to your target system, attribute mappings, correlation rules, and reconciliation jobs.
xml/DBAT-target-template.xml	This file contains definitions for the connector objects required for creating a target application. It includes certain details required to connect Oracle Identity Governance with the target system. It also includes configuration details specific to your target system, attribute mappings, correlation rules, and reconciliation jobs.

C.2 Files and Directories in the Generated Connector Package

Table C-3 describes the files and directories in the generated connector package.

Table C-3 Files and Directories in the Generated Connector Package

File in the Connector Package	Description
bundle/org.identityconnectors.databasetable-12.3.0.jar	This JAR file contains the connector bundle.
configuration/*IT_RES_DEF-CI*.xml	This XML file contains configuration information that is used by the Connector Installer during the connector installation process.
dataset	If you have entered values for the provisionDatasetFile, modifyResourceDatasetFile, or requestDMDatasetsFile entries of the groovy file, then the dataset directory contains the Dataset.xml file. Otherwise this directory is empty. The Dataset.xml file contains dataset-related definitions for the create and modify user provisioning operations. This file is used if you want to enable request-based provisioning.
resources/dbat-generator.properties	This property file contains locale-specific properties. You can use this file as a template to add or update locale-related properties.
xml/*IT_RES_DEF-*ConnectorConfig.xml file	This XML file contains definitions for connector components such as IT resource, lookup definitions, scheduled tasks, process forms, and resource objects. This file is also referred to as the connector configuration file.