

Oracle® Identity Manager

Connector Guide for Database Application Tables



Release 11.1.1

E20277-16

June 2022

The Oracle logo, consisting of a solid red square with the word "ORACLE" in white, uppercase, sans-serif font centered within it.

ORACLE®

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software" or "commercial computer software documentation" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle, Java, and MySQL are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

Preface

Audience	x
Documentation Accessibility	x
Related Documents	x
Conventions	x

What's New in Oracle Identity Manager Connector for Database Application Tables?

Software Updates	xii
Documentation-Specific Updates	xiii

1 About the Database Application Tables Connector

1.1	Introduction to the Database Application Tables Connector	1-1
1.2	Understanding Target System Discovery in the DBAT Connector	1-2
1.3	Certified Components for the DBAT Connector	1-3
1.4	Usage Recommendation for the DBAT Connector	1-4
1.5	About Certified Languages for the DBAT Connector	1-5
1.6	About Supported Data Types	1-5
1.7	DBAT Connector Architecture	1-7
1.8	Flowchart of the DBAT Connector Deployment Process	1-10
1.9	Understanding the DBAT Connector Features	1-11
1.9.1	Full and Incremental Reconciliation	1-11
1.9.2	Limited (Filtered) Reconciliation	1-11
1.9.3	Support for Both Target Resource and Trusted Source Reconciliation	1-11
1.9.4	Support for Reconciliation of Deleted User Records	1-11
1.9.5	Transformation and Validation of Account Data	1-12
1.9.6	Support for Adding User-Defined Fields for Reconciliation and Provisioning	1-12
1.9.7	Support for Configuring the Connector for Stored Procedures	1-12

2 Generating the Database Application Tables Connector

2.1	Overview of Configuring the Groovy File	2-1
2.1.1	About the Groovy File	2-1
2.1.2	Configuring the DBATConfiguration.groovy File	2-2
2.1.3	Entries in the Predefined Sections	2-2
2.1.3.1	itResourceDefName	2-3
2.1.3.2	itResourceName	2-3
2.1.3.3	connectorDir	2-4
2.1.3.4	xmlFile	2-4
2.1.3.5	configFile	2-4
2.1.3.6	propertiesFile	2-5
2.1.3.7	version	2-5
2.1.3.8	trusted	2-5
2.1.3.9	provisionDatasetFile	2-5
2.1.3.10	modifyResourceDatasetFile	2-5
2.1.3.11	requestDMDatasetsFile	2-6
2.1.3.12	bundleJar	2-6
2.1.3.13	config	2-6
2.1.3.14	alias	2-15
2.1.3.15	prepopulate	2-17
2.1.4	Configuring the DBATConfiguration.groovy File for a Target System with an Autoincrement Primary Key	2-18
2.1.5	Determining the Value for the jdbcUrlTemplate Property	2-19
2.1.5.1	jdbcUrlTemplate Property for IBM DB2	2-19
2.1.5.2	jdbcUrlTemplate Property for Microsoft SQL Server	2-19
2.1.5.3	jdbcUrlTemplate Property for MySQL	2-20
2.1.5.4	jdbcUrlTemplate Property for Oracle Database	2-20
2.1.5.5	jdbcUrlTemplate Property for Oracle RAC	2-21
2.1.5.6	jdbcUrlTemplate Property for Sybase Adaptive Server Enterprise	2-21
2.2	Discover the Schema and Generate the Connector	2-21
2.2.1	Running the DBAT Generator	2-22
2.2.2	Understanding the Generated Connector Package	2-22

3 Installing and Configuring the Database Application Tables Connector

3.1	Prerequisites for Installing the DBAT Connector	3-1
3.1.1	About Creating the Target System User Account for Connector Operations	3-1
3.1.2	About Configuring IBM DB2 Running on IBM z/OS	3-2
3.1.3	About Configuring Oracle Database	3-2
3.2	Overview of Installing DBAT Connector	3-2
3.2.1	Installing the Connector in Oracle Identity Manager	3-2

3.2.2	About Deploying the Connector Bundle in a Connector Server	3-5
3.2.2.1	Installing and Configuring the Connector Server	3-5
3.2.2.2	Running the Connector Server	3-7
3.2.2.3	Installing the Connector on the Connector Server	3-7
3.3	Postinstallation	3-7
3.3.1	Configuring the IT Resource for the Target System	3-7
3.3.2	Configuring the Connector for Date Format	3-10
3.3.3	Configuring the Connector for a Target System with an Autoincrement Primary Key	3-11
3.3.4	About Configuring Oracle Identity Manager	3-11
3.3.4.1	Creating and Activating a Sandbox	3-12
3.3.4.2	Creating a New UI Form	3-12
3.3.4.3	Creating an Application Instance	3-12
3.3.4.4	Publishing a Sandbox	3-13
3.3.4.5	Harvesting Entitlements and Sync Catalog	3-13
3.3.5	Localizing Field Labels in UI Forms	3-13
3.3.6	Clearing Content Related to Connector Resource Bundles from the Server Cache	3-15
3.3.7	Managing Logging	3-15
3.3.7.1	Understanding Log Levels	3-16
3.3.7.2	Enabling Logging	3-17
3.3.8	Using Lookup Definitions	3-18
3.3.9	Setting Up Process Form Fields as Entitlements	3-20
3.3.10	Configuring Process Form Fields as Date Fields	3-20
3.3.11	About Configuring Secure Communication Between the Target System and Oracle Identity Manager	3-20
3.3.11.1	Configuring Secure Communication Between IBM DB2 and Oracle Identity Manager	3-21
3.3.11.2	Configuring Secure Communication Between Microsoft SQL Server and Oracle Identity Manager	3-23
3.3.11.3	Configuring Secure Communication Between MySQL and Oracle Identity Manager	3-24
3.3.11.4	Configuring Secure Communication Between Oracle Database and Oracle Identity Manager	3-25
3.3.11.5	Configuring Secure Communication Between Sybase Adaptive Server Enterprise and Oracle Identity Manager	3-26
3.3.12	Configuring Secure Communication Between the Connector Server and Oracle Identity Manager	3-26
3.3.13	Configuring the Connector for Stored Procedures and Groovy Scripts	3-27
3.4	Upgrading the DBAT Connector	3-31

4 Using the Database Application Tables Connector

4.1	Guidelines to Apply While Using the DBAT Connector	4-1
-----	--	-----

4.2	Overview of Lookup Definitions Used During Connector Operations	4-2
4.2.1	About Predefined Lookup Definitions	4-2
4.2.1.1	Lookup.Configuration.RESOURCE	4-2
4.2.1.2	Lookup.RESOURCE.UM.Configuration	4-3
4.2.1.3	Lookup.RESOURCE.UM.ReconAttrMap	4-4
4.2.1.4	Lookup.RESOURCE.UM.ProvAttrMap	4-5
4.2.1.5	Lookup.RESOURCE.UM.ReconAttrMap.Defaults	4-5
4.2.2	Understanding Custom Lookup Definitions Synchronized with the Target System	4-6
4.3	Understanding Reconciliation Scheduled Jobs	4-7
4.3.1	Scheduled Job for Lookup Field Synchronization	4-7
4.3.2	About Attributes of the Scheduled Jobs	4-8
4.3.2.1	Scheduled Jobs for Reconciliation of User Records	4-8
4.3.2.2	Scheduled Jobs for Reconciliation of Deleted Users Records	4-9
4.3.2.3	Scheduled Jobs for Incremental Reconciliation	4-10
4.3.3	About Configuring Scheduled Jobs for DBAT Connector	4-12
4.3.3.1	Scheduled Jobs for Lookup Field Synchronization and Reconciliation	4-12
4.3.3.2	Configuring Scheduled Jobs	4-13
4.4	About Configuring Reconciliation for DBAT Connector	4-14
4.4.1	Connector Objects Used During Target Resource Reconciliation and Provisioning	4-14
4.4.1.1	User Attributes for Target Resource Reconciliation and Provisioning	4-15
4.4.1.2	Supported Target Resource Reconciliation Functions	4-16
4.4.1.3	Understanding Reconciliation Rule for Target Resource Reconciliation	4-16
4.4.1.4	About Reconciliation Action Rules for Target Resource Reconciliation	4-17
4.4.1.5	Understanding Provisioning Functions	4-19
4.4.2	Overview of Connector Objects Used During Trusted Source Reconciliation	4-19
4.4.2.1	User Attributes for Trusted Source Reconciliation	4-20
4.4.2.2	Viewing Reconciliation Rule for Trusted Source Reconciliation	4-20
4.4.2.3	Viewing the Reconciliation Action Rules for Trusted Source Reconciliation	4-22
4.4.3	About Performing Full Reconciliation and Incremental Reconciliation	4-23
4.4.4	About Performing Limited Reconciliation	4-23
4.4.4.1	Specifying a Value for the Filter Attribute	4-24
4.4.4.2	Specifying a Value for the customizedQuery Parameter	4-24
4.5	Performing Provisioning Operations	4-25
4.6	Configuring Action Scripts	4-25
4.6.1	About Action Scripts	4-26
4.6.2	Lookup Entries for Running Action Scripts	4-26
4.6.3	Running a CMD Script Before a Create Operation	4-27
4.7	About Uninstalling the DBAT Connector	4-27

5	Extending the Functionality of the Database Application Tables Connector	
5.1	Adding Custom OIM User Fields for Trusted Source Reconciliation	5-1
5.2	Adding Custom Fields for Target Resource Reconciliation	5-3
5.3	Adding Custom Fields for Provisioning	5-5
5.4	Configuring Transformation of Data During User Reconciliation	5-7
5.5	Configuring Validation of Data During Reconciliation and Provisioning	5-9
6	Known Issues and Workarounds	
6.1	The Custom Schema Feature of IBM DB2 is not Supported	6-1
A	Sample Stored Procedures and Groovy Scripts	
A.1	Sample Groovy Script for a Create Provisioning Operation	A-1
A.2	Sample Groovy Script for an Update Provisioning Operation	A-2
A.3	Sample Groovy Script for a Delete Provisioning Operation	A-3
A.4	Sample Groovy Script for an Add Child Data Provisioning Operation	A-4
A.5	Sample Stored Procedure and Groovy Script for a Delete Child Data Provisioning Operation	A-6
A.6	Sample Stored Procedure and Groovy Script for Lookup Field Synchronization	A-7
A.7	Sample Stored Procedure and Groovy Script for Full or Filter Reconciliation	A-8
A.8	Sample Stored Procedure and Groovy Script for Incremental Reconciliation	A-12
A.9	Tables Used for Sample Groovy and Configuration Scripts	A-15
B	Performing Common Connector Operations	
B.1	Running Incremental Trusted Source Reconciliation	B-1
B.2	Running Incremental Target Resource Reconciliation	B-2
B.3	Configuring and Performing Lookup Field Synchronization	B-3
B.4	Provisioning Child Data	B-3
C	Files and Directories in the Database Application Tables Connector	
C.1	Files and Directories in the DBAT Connector Installation Media	C-1
C.2	Files and Directories in the Generated Connector Package	C-2

Index

List of Figures

1-1	Connector Architecture	1-7
1-2	Overall Flow of the Connector Deployment Process	1-10
4-1	Reconciliation Rule for Target Resource Reconciliation	4-17
4-2	Reconciliation Action Rules for Target Resource Reconciliation	4-19
4-3	Reconciliation Rule for Trusted Source Reconciliation	4-21
4-4	Reconciliation Action Rules for Trusted Source Reconciliation	4-23

List of Tables

1-1	Certified Components	1-3
2-1	Properties of the Config Entry	2-7
3-1	IT Resource Parameters	3-8
3-2	Log Levels and ODL Message Type:Level Combinations	3-16
3-3	Entries Specific to Groovy Script Configuration	3-30
4-1	Entries in the Lookup.Configuration.RESOURCE Lookup Definition	4-2
4-2	Entries in the Lookup.RESOURCE.UM.Configuration Lookup Definition for a Target Resource Configuration	4-3
4-3	Entries in the Lookup.RESOURCE.UM.Configuration Lookup Definition for a Trusted Source Configuration	4-3
4-4	Entries in the Lookup.RESOURCE.UM.ReconAttrMap.Defaults Lookup Definition	4-6
4-5	Attributes of the RESOURCE Lookup Reconciliation Scheduled Job	4-8
4-6	Attributes of the User Reconciliation Scheduled Jobs	4-9
4-7	Attributes of the Delete User Reconciliation Scheduled Jobs	4-10
4-8	Attributes of the Scheduled Jobs for Incremental Reconciliation	4-10
4-9	Scheduled Jobs for Lookup Field Synchronization and Reconciliation	4-12
4-10	User Attributes for Target Resource Reconciliation and Provisioning	4-15
4-11	Action Rules for Target Resource Reconciliation	4-18
4-12	User Attributes for Trusted Source Reconciliation	4-20
4-13	Action Rules for Trusted Source Reconciliation	4-22
4-14	Lookup Entries for Running Action Scripts	4-26
C-1	Files and Directories on the Installation Media	C-1
C-2	Files and Directories in the Generated Connector Package	C-2

Preface

This guide describes the connector that is used to integrate Oracle Identity Manager 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 Manager, visit the following Oracle Help Center page:

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

For information about Oracle Identity Manager Connectors 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.
<i>italic</i>	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.

Convention	Meaning
<code>monospace</code>	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 Oracle Identity Manager Connector for Database Application Tables?

This chapter provides an overview of the updates made to the software and documentation for the Database Application Tables connector in release 11.1.1.6.0.

The updates discussed 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 this guide. These changes are not related to software updates.

Software Updates

The following section discusses software updates:

- [Software Updates in Release 11.1.1.6.0](#)
- [Software Updates in Release 11.1.1.5.0](#)

Software Updates in Release 11.1.1.6.0

The following are the software updates in release 11.1.1.6.0:

- [Support for Using the Connector with a Target System Containing Autoincrement Columns](#)
- [Additional Properties Related to Scripting Have Been Added](#)

Support for Using the Connector with a Target System Containing Autoincrement Columns

You can create and use connector for a target system that has the primary key column defined with the autoincrement option.

See the following sections for more information about the configuring the connector for autoincrementing columns:

- [Configuring the DBATConfiguration.groovy File for a Target System with an Autoincrement Primary Key](#)
- [Configuring the Connector for a Target System with an Autoincrement Primary Key](#)

Additional Properties Related to Scripting Have Been Added

In the earlier release, the `DBATConfiguration.groovy` file contained the following script properties:

- `CreateScript`
- `UpdateScript`
- `DeleteScript`
- `ExecuteQueryScript`

This meant that you could configure the connector to use custom stored procedures or SQL statements for the Create User, Update User, Delete User provisioning operations, and custom SQL queries for user reconciliation.

In this release, in addition to the preceding properties, the following properties have been added:

- `SyncScript` - Used for specifying custom stored procedures or SQL queries to perform incremental reconciliation.
- `lookupScript` - Used for specifying custom stored procedures or SQL queries to perform lookup field synchronization.
- `addMultiValuedAttributeScript` and `removeMultiValuedAttributeScript` - Used for specifying custom stored procedures or SQL statements to perform provisioning operations corresponding to adding and removing multivalued attributes.

See [Table 2-1](#) for detailed information about each of these properties.

Software Updates in Release 11.1.1.5.0

This is the first release of the Oracle Identity Manager connector for Database Application Tables based on ICF architecture. Therefore, there are no software updates in this release.

Documentation-Specific Updates

The following sections discuss documentation-specific updates:

- [Documentation-Specific Updates in Release 11.1.1.6.0](#)
- [Documentation-Specific Updates in Release 11.1.1.5.0](#)

Documentation-Specific Updates in Release 11.1.1.6.0

The following documentation-specific update has been made in revision "14" of this guide:

The "Target systems" row of [Table 1-1](#) has been updated to include support for Microsoft SQL Server 2016.

The following documentation-specific update has been made in revision "13" 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 12c (12.2.1.4.0).

The following documentation-specific updates have been made in revision "12" of this guide:

- Step 4 has been added regarding password reset while configuring the connector for groovy scripts in [Configuring Secure Communication Between the Connector Server and Oracle Identity Manager](#).
- [Known Issues and Workarounds](#) has been added to the guide.

The following documentation-specific update has been made in revision "11" of this guide:

The "Target Systems" row of [Table 1-1](#) has been updated to include IBM DB2 version 11.x.

The following documentation-specific update has been made in revision "10" of this guide:

- Information pertaining to "Datasource Configuration" under the entry "Config" of [Entries in the Predefined Sections](#) has been updated.
- The "datasource" and "jndiProperties" rows of the following tables have been modified:
 - [Table 2-1](#)
 - [Table 3-1](#)

The following documentation-specific update has been made in revision "09" of this guide:

The "Oracle Identity Manager" row of [Table 1-1](#) has been renamed as "Oracle Identity Governance or Oracle Identity Manager" and also updated for Oracle Identity Governance 12c (12.2.1.3.0) certification.

The following documentation-specific updates have been made in revision "08" of this guide:

- The "Filter" row of [Table 4-7](#) has been updated.
- The code lines have been modified in the following sections:
 - [Sample Stored Procedure and Groovy Script for Full or Filter Reconciliation](#)
 - [Sample Stored Procedure and Groovy Script for Incremental Reconciliation](#)

The following documentation-specific updates have been made in revision "07" of this guide:

- The "Target Systems" row of [Table 1-1](#) has been updated to include support for pluggable database (PDB) in Oracle Database 12c.
- [jdbcUrlTemplate Property for Oracle Database](#) has been updated to reflect jdbcUrlTemplate for 12c PDB.
- The "[Related Documents](#)" section in [Preface](#) has been updated.
- The "Connector Server" and "JDK" rows have been added to [Table 1-1](#).

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

The "Target systems" and "JDBC Drivers" rows of [Table 1-1](#) have been updated.

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

The "Oracle Identity Manager" row of [Table 1-1](#) has been updated.

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

A "Note" has been added at the beginning of [Extending the Functionality of the Database Application Tables Connector](#).

The following documentation-specific update has been made in revision "03" of this guide:

Major changes have been made in the structure of the guide. The objective of these changes is to synchronize the guide with the way the connector works and to improve the usability of information provided by the guide.

Documentation-Specific Updates in Release 11.1.1.5.0

The following are the documentation-specific updates in revision "02" of this guide:

- Description of *LOOKUP_FIELD_ID* has been modified in Section 1.7.2, "Custom Lookup Definitions Synchronized with the Target System."
- Section 1.7.1.5, "Lookup.*RESOURCE*.UM.ReconAttrMap.Defaults" has been modified.
- The following sections have been added:
 - Section 2.2.2.2, "Configuring the DBATConfiguration.groovy File for a Target System with an Autoincrement Primary Key"
 - Section 2.3.1, "Configuring the Connector for a Target System with an Autoincrement Primary Key"

1

About the Database Application Tables Connector

This chapter introduces the Database Application Tables connector.

Topics:

- [Introduction to the Database Application Tables Connector](#)
- [Understanding Target System Discovery in the DBAT Connector](#)
- [Certified Components for the DBAT Connector](#)
- [Usage Recommendation for the DBAT Connector](#)
- [About Certified Languages for the DBAT Connector](#)
- [About Supported Data Types](#)
- [DBAT Connector Architecture](#)
- [Flowchart of the DBAT Connector Deployment Process](#)
- [Understanding the DBAT Connector Features](#)

1.1 Introduction to the Database Application Tables Connector

Database Application Tables connector (DBAT connector) enables the exchange of user data between the database and Oracle Identity Manager.

Oracle Identity Manager (OIM) platform automates access rights management, security, and provisioning of IT resources. Oracle Identity Manager connects users to resources, and revokes and restricts unauthorized access to protect sensitive corporate information. Oracle Identity Manager connectors are used to integrate Oracle Identity Manager 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 Manager. After you integrate the tables with Oracle Identity Manager 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 Manager.

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 Manager.

 **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, 11.1.1.6.0.

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 Manager connectors.

In general, there are two broad categories of target systems for which Oracle Identity Manager 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 for the DBAT Connector

These are the software components and their versions required for installing and using the DBAT connector.

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

Table 1-1 Certified Components

Item	Requirement
Oracle Identity Governance or Oracle Identity Manager	You can use one of the following releases of Oracle Identity Governance or Oracle Identity Manager: <ul style="list-style-type: none"> • Oracle Identity Governance 12c (12.2.1.4.0) • Oracle Identity Governance 12c (12.2.1.3.0) • Oracle Identity Manager 11g Release 2 PS3 (11.1.2.3.0) • Oracle Identity Manager 11g Release 2 PS2 (11.1.2.2.0) and any later BP in this release track • Oracle Identity Manager 11g Release 2 PS1 (11.1.2.1.0) and any later BP in this release track • Oracle Identity Manager 11g Release 2 BP10 (11.1.2.0.10) and any later BP in this release track
Target systems	The target system can be database tables from any one of the following RDBMSs: <ul style="list-style-type: none"> • IBM DB2 Version 9.x, 10.x, 11.x • Microsoft SQL Server 2005, 2008, 2012, 2014, 2016, 2017 • MySQL 5.x • Oracle Database 10g and 11g as either single database or Oracle RAC implementation • Oracle Database 12c as single database, pluggable database (PDB), or Oracle RAC implementation • 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 • Sybase Adaptive Server Enterprise 15.x • IBM DB2 version 12 or later

Table 1-1 (Cont.) Certified Components

Item	Requirement
JDBC drivers	<p>Depending on the target system that you use, download one of the following sets of JDBC drivers from the Vendor's Web site:</p> <p>For IBM DB2:</p> <ul style="list-style-type: none"> For all platforms: db2jcc For IBM DB2 with the autoincrement option set on the primary key column: db2jcc4 <p>For Microsoft SQL Server:</p> <ul style="list-style-type: none"> For Microsoft SQL Server 2014: sqljdbc4 version 4.0 For Microsoft SQL Server 2012: sqljdbc4 version 4.0 For Microsoft SQL Server 2008: sqljdbc4 version 2.0 For Microsoft SQL Server 2005: sqljdbc version 1.2 <p>For MySQL: mysql-connector-java-5.1.12-bin</p> <p>For Oracle Database or Oracle RAC:</p> <ul style="list-style-type: none"> If you are using JDK 1.6: ojdbc6 If you are using JDK 1.5: ojdbc5 For all other platforms: ojdbc14 For Oracle Database with autoincrement columns: ojdbc5 or ojdbc6 <p>For Sybase Adaptive Server Enterprise: jconn3</p>
Connector Server	11.1.2.1.0
Connector Server JDK	JDK 1.6 or later
Format in which user data is stored in the target system	<p>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:</p> <ul style="list-style-type: none"> 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.
Other requirements of the target system	<p>The target system must meet the following requirement:</p> <p>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.</p>

1.4 Usage Recommendation for the DBAT Connector

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

You must deploy and use one of the following connectors:

- If you are using an Oracle Identity Manager release that is earlier than Oracle Identity Manager 11g Release 2 BP10 (11.1.2.0.10), then you must use the 9.1.0.x version of this connector.
- If you are using any of the Oracle Identity Manager releases listed in [Table 1-1](#), then you must use the latest 11.1.1.x version of this connector.

1.5 About Certified Languages for the DBAT Connector

The DBAT 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 About Supported Data Types

Learn more about the data types supported for reconciliation and provisioning.

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

For Sybase Database:

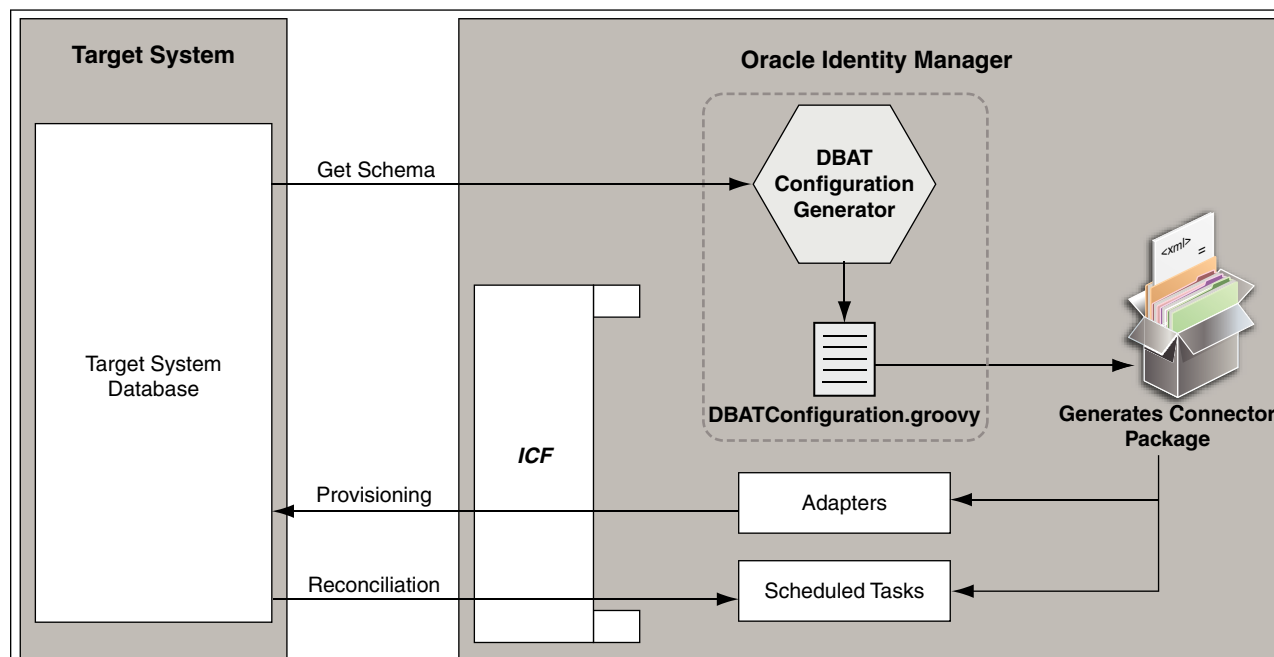
- CHAR
- DATE
- VARCHAR
- TINYINT
- SMALLINT
- INT
- NUMERIC
- DECIMAL
- FLOAT
- REAL
- DATETIME

1.7 DBAT Connector Architecture

The DBAT 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. The ICF is shipped along with Oracle Identity Governance. Therefore, you need not configure or modify it.

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 Manager 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 Manager.

The DBAT connector is not shipped with any metadata as it is a connector for a discovered target. Depending on the database schema, the connector artifacts are generated during connector deployment.

The following is a high-level description of the stages into which the connector deployment and usage procedure is divided into:

- **Generating the connector**

Discovering the schema is one of the important aspects in generating the connector. The DBAT connector includes a groovy file in which you can specify information about your target system. This information is used by the DBAT Generator, one of the deployment utilities shipped with the connector, to discover your schema and generate the connector.

In other words, when you run the DBAT generator on the groovy file, the connector package is generated. This package contains an XML file that contains definitions for connector components such as adapters, process tasks, scheduled tasks, lookup definitions, and IT resource. Connector operations such as provisioning and reconciliation are performed using these connector components.

- **Installing and configuring the connector**

In this stage, you install the generated connector by running the connector installer and then perform configuration tasks such as configuring the IT resource, enabling logging and so on.

- **Using the connector**

In this stage, you start using the connector to perform connector operations such as reconciliation and provisioning.

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 Manager.

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 Manager. 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 Manager, 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 Manager 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 Manager.

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.

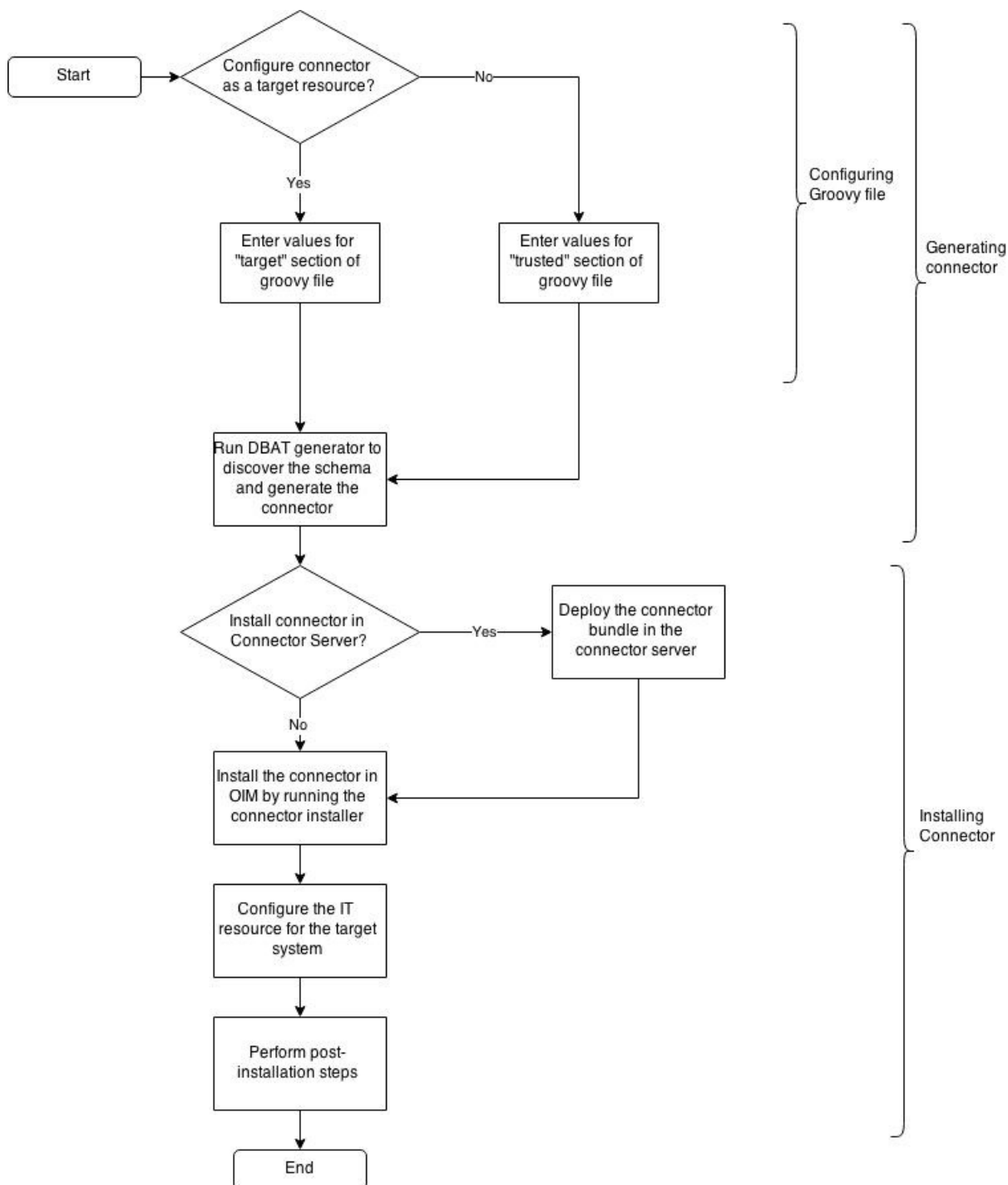
 **Note:**

- 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 Manager* for detailed information about connector deployment configurations.

1.8 Flowchart of the DBAT Connector Deployment Process

This topic provides a flowchart of the DBAT connector deployment process. Each of the steps in this deployment process is discussed in the subsequent chapters.

Figure 1-2 Overall Flow of the Connector Deployment Process



1.9 Understanding the DBAT Connector Features

Features in DBAT connector include full reconciliation, incremental reconciliation, limited reconciliation, validation of account data, and so on.

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 Manager. 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 Manager.

See [About 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 Manager 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 [About 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 Manager.

See [Understanding 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 Manager during reconciliation and provisioning. In addition, you can configure transformation of account data that is brought into Oracle Identity Manager during reconciliation. The following sections provide more information:

- [Configuring Transformation of Data During User Reconciliation.](#)
- [Configuring Validation of Data During Reconciliation and Provisioning.](#)

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 Manager.

The following sections provide more information:

- [Adding Custom OIM User Fields for Trusted Source Reconciliation](#)
- [Adding Custom Fields for Target Resource Reconciliation](#)
- [Adding Custom Fields for Provisioning](#)

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

Generating the Database Application Tables Connector

To generate the DBAT connector, you must configure the Groovy File and run the DBAT Generator to discover the schema.

The procedure to generate the Database Application Tables (DBAT) connector is divided into the following stages:

- [Overview of Configuring the Groovy File](#)
- [Discover the Schema and Generate the Connector](#)

2.1 Overview of Configuring the Groovy File

The groovy file shipped with the connector can be used to specify values for properties that can store basic information about the target system schema.

This section includes the following topics:

- [About the Groovy File](#)
- [Configuring the DBATConfiguration.groovy File](#)
- [Entries in the Predefined Sections](#)
- [Configuring the DBATConfiguration.groovy File for a Target System with an Autoincrement Primary Key](#)
- [Determining the Value for the jdbcUrlTemplate Property](#)

2.1.1 About the Groovy File

The DBAT connector is shipped with a groovy file named `DBATConfiguration.groovy`. This file is located in the `dbat-RELEASE_NUMBER/generator/dbat-generator-RELEASE_NUMBER` directory of the connector installation ZIP. You use the `DBATConfiguration.groovy` file to specify values for properties that can store basic information about your target system schema. This file is used by the DBAT Generator to perform the following tasks:

- Discover the schema
- Configure the mode (trusted source or target resource) in which you want to run the connector
- Generate the connector package specific to your target system

The procedure for running the DBAT Generator and directory structure of the generated connector package is discussed later in this chapter.

The `DBATConfiguration.groovy` file contains sample configuration (one each for trusted source and target resource) with pre-populated values for most of the entries. Depending upon your requirements, specify or modify values for entries in this file or create new sections

for your configuration. The following are the predefined sections in the DBATConfiguration.groovy file:

- **trusted**
You specify values for the entries in this section if you want to configure the connector for the trusted source mode.
- **target**
You specify values for the entries in this section if you want to configure the connector for the target resource mode.

2.1.2 Configuring the DBATConfiguration.groovy File

To configure the DBATConfiguration.groovy file:

1. Download the connector installation ZIP file from Oracle Technology Network.
2. Extract the contents of the connector installation ZIP to any directory on the computer hosting OIM. This creates a directory named `dbat-RELEASE_NUMBER`. See [Files and Directories in the Database Application Tables Connector](#) for information about all the files and directories in the connector installation ZIP.
3. Extract the contents of the `dbat-RELEASE_NUMBER/generator/dbat-generator-RELEASE_NUMBER.zip` file to any directory. This creates a directory named `dbat-generator-RELEASE_NUMBER`.
4. In a text editor, open the DBATConfiguration.groovy file located in the `dbat-generator-RELEASE_NUMBER/resources` directory.
5. Specify values for entries in one of the following predefined sections:
 - **trusted** - for configuring your target system as a trusted source.
 - **target** - for configuring your target system as a target resource.

 **Note:**

The entries in these predefined sections are described later in this section.

6. Save and close the DBATConfiguration.groovy file.

2.1.3 Entries in the Predefined Sections

This section describes the entries in the predefined sections, **trusted** and **target**, of the DBATConfiguration.groovy file.

 **Note:**

- Unless specified, all entries described here are common to both sections.
- If you do not want to specify a value for any of the optional entries or attributes in the `DBATConfiguration.groovy` file, then comment out that entry or attribute by prefixing it with the double-slash symbol (`//`).
- Some of the entries and properties in the `DBATConfiguration.groovy` file contain sample values such as `USERINFO`, `USER_INFO`, `USER_GROUP`, and so on. These sample values must be replaced with the actual table names present in your target system.

- `itResourceDefName`
- `itResourceName`
- `connectorDir`
- `xmlFile`
- `configFile`
- `propertiesFile`
- `version`
- `trusted`
- `provisionDatasetFile`
- `modifyResourceDatasetFile`
- `requestDMDatasetsFile`
- `bundleJar`
- `config`
- `alias`
- `prepopulate`

2.1.3.1 `itResourceDefName`

This is a mandatory entry. Enter the name of the IT resource type for the target system. Note that the value that you specify for this entry determines the name of the connector package, connector configuration file, and connector installer file. For example, if you specify `DBAT` as the value of this entry, then the name of the connector package directory is `DBAT.zip`. See [Understanding the Generated Connector Package](#) for the directory structure of the connector package.

2.1.3.2 `itResourceName`

This is an optional entry. Enter the name of the IT resource for the target system. If this entry is commented, then the IT resource name will be the same as the value of the `ITResourceDefName` entry.

Sample value: `DBAT`

 **Note:**

The value of this entry must be unique for each connector that you create for your target system database. In addition, this value will be a part of the names for all connector components (defined in the connector configuration XML file, which is created after you run the DBAT Generator) such as lookup definitions, resource objects, process forms, and scheduled tasks.

For example, if you specify `DBAT` as the value of `itResourceName` entry, then after you deploy the connector, the configuration lookup definition is created and its name will be `Lookup.Configuration.DBAT`.

2.1.3.3 connectorDir

This is an optional entry. This entry is the name of the directory that contains the connector package that is generated when you run the DBAT Generator. By default, the value of this entry is the same as the value of the `itResourceName` entry.

2.1.3.4 xmlFile

This is an optional entry. Enter the name and relative path of the XML file that must contain definitions of the connector objects. If you do not specify a value for this entry, then the file name is generated in the following format:

`IT_RES_DEF_NAME-ConnectorConfig.xml`

In this format, `IT_RES_DEF_NAME` is the value of the `itResourceDefName` entry.

For example, if you have not specified a value for this entry and `DBAT` is the value of the `itResourceDefName` entry, then the name of the XML file that is generated is `DBAT-ConnectorConfig.xml`.

 **Note:**

To easily identify files of a specific target system installation, it is recommended that the names of this generated XML file be prefixed with the name of the IT resource for the target system.

Sample value: `DBAT-ConnectorConfig.xml`

2.1.3.5 configFile

This is an optional entry. Enter the name and relative path of the XML file that contains the configuration information of the connector objects. If you do not specify a value for this entry, then the file name is generated in the following format:

`IT_RES_DEF_NAME-CI.xml`

In this format, `IT_RES_DEF_NAME` is the value of the `itResourceDefName` entry.

For example, if you have not specified a value for this entry and `DBAT` is the value of the `itResourceDefName` entry, then the name of the XML file that is generated is `DBAT-CI.xml`.

2.1.3.6 propertiesFile

This is an optional entry. Enter the name and relative path of the `.properties` file which contains the resource bundle translations. If you do not specify a value for this entry, then the file name is generated in the following format:

`IT_RES_DEF_NAME-generator.properties`

In this format, `IT_RES_DEF_NAME` is the value of the `itResourceDefName` entry.

For example, if you have not specified a value for this entry and `DBAT` is the value of the `itResourceDefName` entry, then the name of the properties file that is generated is `DBAT-generator.properties`.

2.1.3.7 version

This is an optional entry. Enter the release number of the connector.

2.1.3.8 trusted

This is a mandatory entry and present only in the section for trusted source configuration. Set the value of the entry to `true`, if you are configuring your target system as a trusted source.

2.1.3.9 provisionDatasetFile

This is an optional entry and present only in the section for target resource configuration. Specify a value for this entry only if you are using Oracle Identity Manager release 11.1.1.x and you want to generate request datasets to perform request-based provisioning operations.

Enter the name and relative path of the request dataset (an XML file) that specifies information to be submitted by the requester during a create user provisioning operation.



Note:

To easily identify files of a specific target system installation, it is recommended that the names of this generated XML file (for both provisioning and modification) be prefixed with the name of the IT resource for the target system.

Sample value: `ProvisionResource_DBATUser.xml`

2.1.3.10 modifyResourceDatasetFile

This is an optional entry and present only in the section for target resource configuration. Specify a value for this entry only if you are using Oracle Identity Manager release 11.1.1.x and you want to generate request datasets to perform request-based provisioning operations.

Enter the name and relative path of the request dataset (an XML file) that specifies information to be submitted by the requester during an update user provisioning operation.

Sample value: `ModifyProvisionedResource_DBATUser.xml`

2.1.3.11 requestDMDatasetsFile

This is also an optional entry present only in the section for target resource configuration. Specify a value for this entry only if you are using Oracle Identity Manager release 11.1.1.x and you want to generate request datasets that can be imported by using the Deployment Manager to perform request-based provisioning.

Enter the name and relative path of the request dataset (an XML file) that specifies information to be submitted by the requester during provisioning operations.

Sample value: `Datasets.xml`

2.1.3.12 bundleJar

This is a mandatory entry. Enter the name and relative path of the JAR file containing the ICF bundle that the DBAT generator will use.

Default value: `../lib/org.identityconnectors.databasetable-1.2.2.jar`

Do not change the value of this entry.

2.1.3.13 config

This is a mandatory entry in which you specify information about the connector configuration. This connector configuration contains information about the manner in which the connector must behave and connect to the target system. Note that this connector can be configured to connect to the target system by using one of the following methods:

JDBC Driver Configuration

The following are the two ways in which the connector can be configured to connect to the target system by using JDBC driver configurations:

- Specify a value for the `jdbcUrlTemplate` property that contains wildcard values such as `%h`, `%p`, and `%d`. The wildcard values are replaced with the actual values specified for the host, port, and database properties. In this configuration method, specifying values for the host, port, and database properties are mandatory.
- Specify an exact URL template as the value of the `jdbcUrlTemplate` property. In other words, the URL template must not contain wildcard values. In addition, specify a value for the user property.

DataSource Configuration

In this configuration method, you must,

- Specify a value for the `datasource` property. In addition, specify a value for the `jndiProperties` property if the `datasource` is bound to JNDI entities.
- Ensure `wlfullclient.jar` and JDBC driver files exist in the `lib` directory of `dbat` generator directory along with other jars.
- Add Java property `-Dweblogic.jdbc.remoteEnabled=true` in Weblogic OIM Domain Environment script and restart Weblogic server.

[Table 2-1](#) lists and describes the properties of the Config entry.

Table 2-1 Properties of the Config Entry

Property	Type	Required for JDBC Driver Configuration?	Required for DataSource Configuration?	Default Value	Description
table	String	Yes	Yes	NA	Name of the parent table or view that contains user records.
keyColumn	String	Yes	Yes	NA	Name of the column that uniquely identifies each row in the parent table.
passwordColumn	String	No	No	NA	Name of the column in the parent table that holds the passwords of the target system records. You must specify a value for this property if both the following conditions are true: - There exists a column in the parent table that holds passwords of the target system records. - You want to perform the reset account password provisioning operation.
user	String	Yes	No	NA	User ID of the database user account that Oracle Identity Manager will use to connect to the target system
password	String	Yes	No	NA	Password of the database user account that Oracle Identity Manager must use to connect to the target system. For security reasons, this property is commented by default. Therefore, do not uncomment this property. When you run the DBATGenerator.cmd file, you will be prompted to enter the password. See Discover the Schema and Generate the Connector for more information about running the DBAT Generator.
jdbcUrlTemplate	String	Yes(%h, %p, and %d)	No	NA	JDBC URL template of the target database with the %h, %p, and %d wildcards. For IBM DB2: jdbc:db2://%h:%p/%d For Microsoft SQL Server: jdbc:sqlserver://%h:%p;databaseName=%d For MySQL: jdbc:mysql://%h:%p/%d For Oracle Database: jdbc:oracle:thin:@%h:%p:%d For Sybase Adaptive Server Enterprise: jdbc:sybase:Tds:%h:%p/%d See Determining the Value for the jdbcUrlTemplate Property for more information.

Table 2-1 (Cont.) Properties of the Config Entry

Property	Type	Required for JDBC Driver Configuration?	Required for DataSource Configuration?	Default Value	Description
jdbcDriver	String	Yes	Yes	NA	JDBC driver class name. For IBM DB2: com.ibm.db2.jcc.DB2Driver For Microsoft SQL Server: com.microsoft.sqlserver.jdbc.SQLServerDriver For MySQL: com.mysql.jdbc.Driver For Oracle Database: oracle.jdbc.driver.OracleDriver For Sybase Adaptive Server Enterprise: com.sybase.jdbc3.jdbc.SybDriver
relationTables	String	No	No	NA	If user data is spread across parent and child tables, then enter a comma-separated list of child table names. Sample value: USER_ROLE, USER_GROUP Note: In addition to a foreign key, if your child table does not contain any other column that is unique, not null, or primary, then generation of the XML file fails. See Discover the Schema and Generate the Connector for information about generating the XML file.
statusColumn	Boolean	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.
enableValue	String	No	No	NA	Value used on the target system that depicts that a user record is in the enabled status.
disableValue	String	No	No	NA	Value used on the target system that depicts that a user record is in the disabled status.
database	String	Yes, when %d	No	NA	Name of the target system database. Used in place of the %d wild card in the jdbcUrlTemplate property.
host	String	Yes, when %h	No	NA	Host name or IP address of the computer hosting the target system. Used in place of the %h wild card in the jdbcUrlTemplate property.
port	String	Yes, when %p	No	NA	Port number at which the target system is listening. Used in place of the %p wild card in the jdbcUrlTemplate property.
datasource	String	No	Yes	NA	Name of the data source when the connector uses the datasource configuration to connect to the target system. Sample Value: jdbc/operationsDB

Table 2-1 (Cont.) Properties of the Config Entry

Property	Type	Required for JDBC Driver Configuration?	Required for DataSource Configuration?	Default Value	Description
jndiProperties	String	No	Yes	NA	<p>Properties used to establish a connection with the target system by using JDBC drivers or look up a DataSource using JNDI.</p> <p>Sample value:</p> <pre>java.naming.factory.initial=weblogic.jndi.WLInitialContextFactory, java.naming.provider.url=t3:// acme.com:15000, java.naming.security.principal=weblogic, java.naming.security.credentials=weblogic1</pre>
customizedQuery	String	No	No	NA	<p>A WHERE clause specifying the subset of newly added or modified records that you want to reconcile.</p> <p>See Specifying a Value for the customizedQuery Parameter for more information on the customizedQuery property.</p>
rethrowAllSQLExceptions	Boolean	No	No	false	<p>If the value of this property is set to <code>false</code>, then SQL exceptions with a zero (0x00) error code are considered a success. In other words, SQL exceptions with the zero error code are caught and suppressed by the SQL statement.</p> <p>If the value of this property is set to <code>true</code>, then all SQL exceptions with the zero error code result.</p> <p>Note: No other exceptions SQL exceptions are influenced by this property.</p>
allNative	Boolean	No	No	false	<p>If the value of this property is set to <code>false</code>, then attribute data is converted to Strings by using the JDBC driver.</p> <p>Set the value of this property to <code>true</code> to use the appropriate JDBC types and to force the connector to perform the conversion.</p> <p>The <code>dateFormat</code> and <code>timestampFormat</code> properties invalidate this setup.</p>
validConnectionQuery	String	No	No	NA	<p>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:</p> <pre>SELECT 1 FROM DUMMY</pre>

Table 2-1 (Cont.) Properties of the Config Entry

Property	Type	Required for JDBC Driver Configuration?	Required for DataSource Configuration?	Default Value	Description
dateFormat	String	No	No	dd/MM/yyyy	Allows the user to format how date data is converted to strings. However, if you want to change the date format from text to a date editor, then you must set the value of the dateFormat IT resource parameter to the date format specified in the system configuration of Oracle Identity Manager. Specifying a value for this property invalidates the allNative property.
timestampFormat	String	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.
enableEmptyString	Boolean	No	No	false	Set the value of this property to <code>true</code> if you want to enable support for writing an empty string instead of a NULL value. If the value of this property is set to <code>false</code> , then empty strings are written as NULL values. Note: This property can be applied only to mandatory String attributes.
quoting	String	No	No	None	If you are using protected keywords as names of columns in your target system, then use quoting, which displays column names between quoting properties (such as single quotes, double quotes, and so on) in the generated SQL when accessing the database. If you do not want to use quoting, then specify <code>None</code> . If you want use to use quoting, then specify one of the following quoting properties: - For DB2 running on IBM Mainframes: Single - For DB2 running on MS Windows: Double - For MS SQL Server: Brackets or Double - For MySQL: Back - For Oracle: Double - For Sybase: Double Note: Using the "WHERE" keyword as a column name is not supported.

Table 2-1 (Cont.) Properties of the Config Entry

Property	Type	Required for JDBC Driver Configuration?	Required for DataSource Configuration?	Default Value	Description
changeLogColumn	String	No	No	NA	<p>Optional name of the column where the last update-related number, non-decreasing, date or timestamp-based values are stored. Can also be a column name storing values that are not date or time stamp based (for example, numeric or strings).</p> <p>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 About Supported Data Types for information about data types supported for your target system.</p> <p>The values in this column are used during incremental reconciliation to determine the newest or most youngest record reconciled from the target system.</p> <p>Note: You must specify a value for this property if you want to perform incremental reconciliation.</p>
nativeTimestamps	Boolean	No	No	false	<p>If the value of this property is set to <code>false</code>, then timestamp data is read as Strings by using the JDBC driver, which can cause a loss of time in milliseconds.</p> <p>If the value of this property is set to <code>true</code>, then timestamp data is retrieved as <code>java.sql.Timestamp</code> type by the connector.</p>
createScript	String	No	No	None	<p>This property is present only in the section for target resource configuration.</p> <p>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.</p> <p>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.</p> <p>You must enter the file URL in the following format:</p> <p><code>file:///URL</code></p> <p>Sample value: <code>file:///home/jdoe/dbat/scripts/create_user.groovy</code></p> <p>See Configuring the Connector for Stored Procedures and Groovy Scripts for more information on configuring the connector to use custom stored procedures.</p>

Table 2-1 (Cont.) Properties of the Config Entry

Property	Type	Required for JDBC Driver Configuration?	Required for DataSource Configuration?	Default Value	Description
updateScript	String	No	No	None	<p>This property is present only in the section for target resource configuration.</p> <p>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.</p> <p>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.</p> <p>You must enter the file URL in the following format: file:///URL</p> <p>Sample value: file:///home/jdoe/dbat/scripts/update_user.groovy</p> <p>See Configuring the Connector for Stored Procedures and Groovy Scripts for more information on configuring the connector to use custom stored procedures.</p>
deleteScript	String	No	No	None	<p>This property is present only in the section for target resource configuration.</p> <p>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.</p> <p>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.</p> <p>You must enter the file URL in the following format: file:///URL</p> <p>Sample value: file:///home/jdoe/dbat/scripts/delete_user.groovy</p> <p>See Configuring the Connector for Stored Procedures and Groovy Scripts for more information on configuring the connector to use custom stored procedures.</p>

Table 2-1 (Cont.) Properties of the Config Entry

Property	Type	Required for JDBC Driver Configuration?	Required for DataSource Configuration?	Default Value	Description
lookupScript	String	No	No	None	<p>This property is present only in the section for target resource configuration.</p> <p>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.</p> <p>Enter the Groovy script or the file URL of the Groovy script created for lookup field synchronization.</p> <p>You must enter the file URL in the following format: file:///URL</p> <p>Sample value: file:///home/jdoe/dbat/scripts/lookup_field_sync.groovy</p> <p>See Configuring the Connector for Stored Procedures and Groovy Scripts for more information on configuring the connector to use custom stored procedures.</p>
addMultiValuedAttributeScript	String	No	No	None	<p>This property is present only in the section for target resource configuration.</p> <p>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.</p> <p>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.</p> <p>You must enter the file URL in the following format: file:///URL</p> <p>Sample value: file:///home/jdoe/dbat/scripts/add_mulval_attr.groovy</p> <p>See Configuring the Connector for Stored Procedures and Groovy Scripts for more information on configuring the connector to use custom stored procedures.</p>

Table 2-1 (Cont.) Properties of the Config Entry

Property	Type	Required for JDBC Driver Configuration?	Required for DataSource Configuration?	Default Value	Description
removeMultiValuedAttributeScript	String	No	No	None	<p>This property is present only in the section for target resource configuration.</p> <p>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.</p> <p>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.</p> <p>You must enter the file URL in the following format: file:///URL</p> <p>Sample value: file:///home/jdoe/dbat/scripts/remove_mulval_attr.groovy</p> <p>See Configuring the Connector for Stored Procedures and Groovy Scripts for more information on configuring the connector to use custom stored procedures.</p>
executeQueryScript	String	No	No	None	<p>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.</p> <p>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).</p> <p>You must enter the file URL in the following format: file:///URL</p> <p>Sample value: file:///home/jdoe/dbat/scripts/recon_user.groovy</p> <p>See Configuring the Connector for Stored Procedures and Groovy Scripts for more information on configuring the connector to use custom stored procedures.</p>

Table 2-1 (Cont.) Properties of the Config Entry

Property	Type	Required for JDBC Driver Configuration?	Required for DataSource Configuration?	Default Value	Description
syncScript	String	No	No	None	<p>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.</p> <p>Enter the Groovy script or the file URL of the Groovy script created for incremental reconciliation.</p> <p>You must enter the file URL in the following format: file:///URL</p> <p>Sample value: file:///home/jdoe/dbat/scripts/increm_recon_user.groovy</p> <p>See Configuring the Connector for Stored Procedures and Groovy Scripts for more information on configuring the connector to use custom stored procedures.</p>

2.1.3.14 alias

The DBAT generator uses aliases to create relationships between the column or table names in the target system and resource object field names or process form field names in Oracle Identity Manager. In addition, the DBAT generator uses aliases to shorten long database names to meet the character-length restrictions on form names and form field names in Oracle Identity Manager. Aliasing can be used on column name, table name, form name, and form field name levels. Note that target system columns are represented as connector attributes.

Depending on the type of configuration, specify values for one of the following sections:

- For trusted source configuration

In the trusted source configuration section, you use the alias entry to map connector attributes or target system column names to OIM User form field names, including UDFs. The mappings that you specify here are used to populate entries in the Lookup.*RESOURCE.UM.ReconAttrMap* lookup definition for trusted source reconciliation.

Note that some of the OIM User form field names do not have the same display name internally. For such fields, you must ensure that you map the connector attribute or target system column name to the internal name rather than the display name. The following table lists the names of the OIM User form display names and their corresponding internal names:

Display Name	Internal Name
Organization	Organization Name
Manager	Manager Login
E-mail	Email

The following is the default value of the alias entry:

```
['__UID__':'User Login', '__NAME__':'Last Name',
'Organization':'Organization Name', 'Xellerate Type':'Xellerate Type',
'__ENABLE__':'Status', 'Role':'Role']
```

In the default value, note that the "Organization" connector attribute has been mapped to "Organization Name", which is the internal name.

You cannot delete existing mappings in the default value. However, you can modify these mappings.

The following is the format in which you must specify values for the alias entry:

```
['CONN_ATTR1 or COL_NAME1': 'OIM_FIELD1', 'CONN_ATTR2 or COL_NAME2':
'OIM_FIELD2', . . . 'CONN_ATTRn or COL_NAMEn': 'OIM_FIELDn']
```

In this format:

- *CONN_ATTR* is the connector attribute name.
- *COL_NAME* is the target system column name.
- *OIM_FIELD* is the name of OIM User field.

See [Table 4-12](#) for information about the `__UID__` and `__NAME__` attributes.

- For target resource configuration

In the target resource configuration section, you use the alias entry for one or all of the following purposes:

- To set an alias for table names specified in the `relationTables` property.
- To set an alias (a unique and shortened name) for the IT resource name specified in the `itResourceName` entry.
- To map connector attributes or target system column names to fields of the process form. The mappings that you specify here are used to populate entries in the `Lookup.RESOURCE.UM.ProvAttrMap` and `Lookup.RESOURCE.UM.ReconAttrMap` lookup definitions. For all fields for which no alias is provided, a mapping is created based on the target system column names.

By default, the `DBATConfiguration.groovy` file contains the following as the value of the alias entry:

```
['USER_ROLE':'RO', 'USER_GROUP':'GR']
```

You can modify the default value to meet the requirements in your target system. For target system fields for which no alias is provided, will be created with the target attribute names

If you want to add mappings for fields other than the ones already present in the alias entry, then you can add them either to the existing values in the alias entry, or add them to the alias + entry.

The following is the default value of the alias + entry:

```
['__NAME__':'User ID']
```

The following is the format in which you must specify values for the alias and alias + entry:

```
['CONN_ATTR1 or COL_NAME1': 'ALIAS_FIELD1', 'CONN_ATTR2 or COL_NAME2':
'ALIAS_FIELD2', . . . 'CONN_ATTRn or COL_NAMEn': 'ALIAS_FIELDn', . . .
```

```
'TABLE_NAME':'ALIAS','TABLE_NAME1':'ALIAS1','TABLE_NAME2':'ALIAS2', . . .
'TABLE_NAMEn':'ALIASn']
```

In this format:

- *CONN_ATTR* is the connector attribute name.
- *COL_NAME* is the target system column name.
- *ALIAS_FIELD* is the alias corresponding to the connector attribute or target system column name.
- *TABLE_NAME* is the name of the table specified in the relationTables attribute.
- *ALIAS* is the alias corresponding to the table name.

See [Table 4-10](#) for information about connectors attributes such as `__NAME__`.

When you run the DBAT Generator (which is described later in this chapter), by using the information specified in the `DBATConfiguration.groovy` file, the connector package is created. The connector package contains an XML folder that in turn contains an XML file. This XML file contains definitions for connector objects such as IT resource type, IT resource, scheduled tasks, process forms, and adapters. The process form names are derived from the database table names, and then prefixed with **UD_RESOURCE_NAME**, where *RESOURCE_NAME* is replaced with the value of the `itResourceName` entry.

The DBAT Generator automatically truncates a process form name if the number of characters in the process form name is more than 8. This might result in two forms having the same name, but the DBAT Generator prevents this from happening by using autonumbering. Therefore, you can specify an alias (a shortened name) to gain control over the autogenerated form name.

This is illustrated by the following example:

Assume that the resource name is DB and database table name is `USER_ROLES`. In addition, assume that Oracle Identity Manager contains a form by the name `UD_DB_US`.

After you run the DBAT Generator, the process form is created and the form name is `UD_DB_USER_ROLES`. The number of characters in this process form name is 16. While importing the connector XML file into Oracle Identity Manager, the DBAT Generator truncates the process form name because it contains more than 8 characters. The truncated form name is `UD_DB_US`. Note that in Oracle Identity Manager, a form by the name `UD_DB_US` already exists.

To avoid encountering such issues or forms with same names, specify a value for the alias attribute, as follows:

```
alias = ['USER_ROLES':'RO']
```

After you run the DBAT Generator, the process form name is `UD_DB_RO`, which contains 8 characters. Therefore, the process form name is not truncated.

2.1.3.15 prepopulate

This is an optional entry present only in the section for target resource configuration. Specify a value for this entry if you want Oracle Identity Manager to prepopulate OIM User fields while provisioning a target system resource.

By default, the value of this entry is as follows:

```
['USERID':'User Login', 'FIRSTNAME':'First Name', 'LASTNAME':'Last Name',
'EMAIL':'Mail Id', 'DESCRIPTION':'Description', 'SALARY':'Salary',
'JOININGDATE':'Join date']
```

This means that the groovy file is configured to prepopulate the following fields:

User Login

First Name

Last Name

Mail Id

Description

Salary

Join date

You can add fields to or remove fields from the preceding list. The following is the format in which you must specify values for the prepopulate entry:

```
['CONN_ATTR1 or COL_NAME1': 'OIM_FIELD1', 'CONN_ATTR2 or COL_NAME2':  
'OIM_FIELD2', . . . 'CONN_ATTRn or COL_NAMEn': 'OIM_FIELDn']
```

In this format:

- `CONN_ATTR` is the connector attribute name.
- `COL_NAME` is the target system column name.
- `OIM_FIELD` is the name of the OIM User field.

See *Working with Prepopulate Adapters in Oracle Fusion Middleware Developing and Customizing Applications for Oracle Identity Manager* for more information about attaching and removing prepopulate adapters.

2.1.4 Configuring the DBATConfiguration.groovy File for a Target System with an Autoincrement Primary Key

In addition to configuring the DBATConfiguration.groovy file as discussed in [Entries in the Predefined Sections](#), perform the following procedure if the key column of your target system has been configured with an autoincrement option:

- For a target system that has been configured as a target resource:
Ensure that the prepopulate entry of the DBATConfiguration.groovy file does not contain any mapping for the key column. Alternatively, do not provide the `__NAME__` attribute mapping in the prepopulate entry.
The same information has been discussed in the prepopulate entry of [Entries in the Predefined Sections](#).
- For a target system that has been configured as a trusted source:
By default, the connector maps the key column of the table specified in the IT resource to the User Login field in Oracle Identity Manager. However, if the key column is configured with the autoincrement option, then in the alias entry, you can map the OIM User Login field to a different target system column.
If you change the default mapping, then ensure that the `__UID__` attribute is mapped to any corresponding field in Oracle Identity Manager.
The same information has been discussed in the alias entry of [Entries in the Predefined Sections](#).

2.1.5 Determining the Value for the jdbcUrlTemplate Property

This section discusses the `jdbcUrlTemplate` property. You apply the information in this section while performing the instructions described in [Overview of Configuring the Groovy File](#).

The values that you specify for the `jdbcUrlTemplate` property depends on the target system:

- [jdbcUrlTemplate Property for IBM DB2](#)
- [jdbcUrlTemplate Property for Microsoft SQL Server](#)
- [jdbcUrlTemplate Property for MySQL](#)
- [jdbcUrlTemplate Property for Oracle Database](#)
- [jdbcUrlTemplate Property for Oracle RAC](#)
- [jdbcUrlTemplate Property for Sybase Adaptive Server Enterprise](#)

2.1.5.1 jdbcUrlTemplate Property for IBM DB2

Enter the following component of the connection URL as the value of the `jdbcUrlTemplate` property:

```
jdbc:db2://[SERVER_NAME[\INSTANCE_NAME][:PORT_NUMBER]]/DATABASE_NAME
```

If you configure secure communication between Oracle Identity Manager and the target system, then enter the following as the value of the `jdbcUrlTemplate` property:

```
jdbc:db2://[SERVER_NAME[\INSTANCE_NAME][:PORT_NUMBER]]/  
DATABASE_NAME:sslConnection=true;sslTrustStoreLocation=LOCATION;sslTrustStorePassword=PASSWORD;
```

In both formats:

- `SERVER_NAME` is the IP address (not the host name) of the target system host computer.
- `INSTANCE_NAME` is the name of the target system database.
- `PORT_NUMBER` is the port at which the target system database is listening.
- `DATABASE_NAME` is the name of the database.
- `LOCATION` is the full path and name of the trust store
- `PASSWORD` is the password of the trust store location.

The following is a sample value for the `jdbcUrlTemplate` property:

```
jdbc:db2://192.168.16.76:50000/acmedb
```

2.1.5.2 jdbcUrlTemplate Property for Microsoft SQL Server

Enter the following component of the connection URL as the value of the `jdbcUrlTemplate` property:

```
jdbc:sqlserver://[SERVER_NAME[\INSTANCE_NAME][:PORT_NUMBER]];DATABASE=DATABASE_NAME
```

In this format:

- *SERVER_NAME* is the IP address (not the host name) of the target system host computer.
- *INSTANCE_NAME* is the name of the target system database.
- *PORT_NUMBER* is the port at which the target system database is listening.
- *DATABASE_NAME* is the name of the target system database

The following is a sample value for the `jdbcUrlTemplate` property:

```
jdbc:sqlserver://192.168.16.76:1433;Database=acmedb
```

2.1.5.3 jdbcUrlTemplate Property for MySQL

Enter the following component of the connection URL as the value of the `jdbcUrlTemplate` property:

```
jdbc:mysql://[SERVER_NAME][:PORT_NUMBER]/[DATABASE_NAME]
```

If you configure the connector to use custom stored procedures for performing provisioning operations and have specified a value for the `updateScript` property (see [Table 2-1](#) for more information), then enter the following as the value of the `jdbcUrlTemplate` property:

```
jdbc:mysql://[SERVER_NAME][:PORT_NUMBER]/[DATABASE_NAME]?  
noAccessToProcedureBodies=true
```

If you configure secure communication between Oracle Identity Manager and the target system, then enter the following as the value of the `jdbcUrlTemplate` property:

```
jdbc:mysql://[SERVER_NAME][:PORT_NUMBER]/[DATABASE_NAME]?  
useSSL=true&trustCertificateKeyStoreType=KEYSTORE_TYPE&trustCertificateKeyStorePa  
s  
sword=PASSWORD&trustCertificateKeyStoreUrl=URL
```

In all the formats:

- *SERVER_NAME* is the IP address (not the host name) of the target system host computer.
- *PORT_NUMBER* is the port at which the target system database is listening.
- *DATABASE_NAME* is the name of the target system database.
- *KEYSTORE_TYPE* is the type of key store used for trusted certificates. The default is JKS.
- *PASSWORD* is the password for the key store of trusted certificates.
- *URL* is the URL to the key store of the trusted certificate.

The following is a sample value for the `jdbcUrlTemplate` property:

```
jdbc:mysql://192.168.1.251/mysql
```

2.1.5.4 jdbcUrlTemplate Property for Oracle Database

Enter the following component of the connection URL as the value of the `jdbcUrlTemplate` property:

```
jdbc:oracle:thin:@TARGET_HOST_NAME_or_IP_ADDRESS:PORT_NUM:sid
```

The following is a sample value for the `jdbcUrlTemplate` property:

```
jdbc:oracle:thin:@ten.mydomain.com:1521:cust_db
```

If you are using Oracle 12c PDB mode, specify the following as the `jdbcUrlTemplate`:

```
jdbc:oracle:thin:@//host:port/serviceName
```

2.1.5.5 jdbcUrlTemplate Property for Oracle RAC

Enter the following component of the connection URL as the value of the `jdbcUrlTemplate` property:



Note:

The `jdbcUrlTemplate` property must not exceed 200 characters because of a restriction in the number of characters for the corresponding parameter in Oracle Identity Manager.

```
jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=HOST1_NAME.DOMAIN)
(PORT=PORT1_NUMBER))(ADDRESS=(PROTOCOL=TCP)(HOST=HOST2_NAME.DOMAIN)(PORT=PORT2_NUMBER))
(ADDRESS=(PROTOCOL=TCP)(HOST=HOST3_NAME.DOMAIN)(PORT=PORT3_NUMBER)) . . .
(ADDRESS=(PROTOCOL=TCP)(HOST=HOSTn_NAME.DOMAIN)(PORT=PORTn_NUMBER))
(CONNECT_DATA=(SERVICE_NAME=ORACLE_DATABASE_SERVICE_NAME)))
```

The following is a sample value for the `jdbcUrlTemplate` property:

```
jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST= host1.example.com)
(PORT=1521))(ADDRESS=(PROTOCOL=TCP)(HOST= host2.example.com)(PORT=1521))
(CONNECT_DATA=(SERVICE_NAME= srvce1)))
```

2.1.5.6 jdbcUrlTemplate Property for Sybase Adaptive Server Enterprise

Enter the following component of the connection URL as the value of the `jdbcUrlTemplate` property:

```
jdbc:sybase:Tds:SERVER_NAME:PORT_NUMBER/DATABASE_NAME
```

In this format:

- `SERVER_NAME` is the IP address (not the host name) of the target system host computer.
- `PORT_NUMBER` is the port at which the target system database is listening.
- `DATABASE_NAME` is the name of the target system database.

The following is a sample value for the `jdbcUrlTemplate` property:

```
jdbc:sybase:Tds:172.21.109.62:9050/master
```

2.2 Discover the Schema and Generate the Connector

After configuring the `DBATConfiguration.groovy` file, you must run the DBAT Generator to discover the schema and generate the connector package.

Topics:

- [Running the DBAT Generator](#)
- [Understanding the Generated Connector Package](#)

2.2.1 Running the DBAT Generator

The DBAT Generator is the `DBATGenerator.cmd` or `DBATGenerator.sh` file that is located in the `dbat-generator-RELEASE_NUMBER/bin` directory (created in Step 3 of [Overview of Configuring the Groovy File](#)).

To run the DBAT Generator:

1. Copy the JDBC driver corresponding to your target system to the `dbat-generator-RELEASE_NUMBER/lib/` directory. The JDBC drivers are listed in [Table 1-1](#).
2. In a command window, change to the `dbat-generator-RELEASE_NUMBER/bin` directory (for example, `dbat-generator-11.1.1.6.0/bin`) and run one of the following commands depending on the operating system that you are using:

- **For Microsoft Windows**

```
DBATGenerator.cmd CONFIG_FILE CONFIG_NAME
```

- **For UNIX**

```
DBATGenerator.sh CONFIG_FILE CONFIG_NAME
```

In this command, replace:

- `CONFIG_FILE` with the absolute or relative path name of the `DBATConfiguration.groovy` file.
- `CONFIG_NAME` with the name of the configuration within the `DBATConfiguration.groovy` file, being used for the target system. The predefined configurations within this file are trusted and target. You can create additional custom configurations with different names depending on your requirements.

The following is a sample command:

```
DBATGenerator.cmd ..\resources\DBATConfiguration.groovy target
```

In this command, "target" denotes the name of the section in the `DBATConfiguration.groovy` file for which values have been specified. In other words, the connector is being configured as a target resource.

3. When prompted, enter a value for User Password, which is the password of the database user account that Oracle Identity Manager must use to connect to the target system.

If you encounter any errors while running the DBAT Generator, then you must fix it and then resume running the DBAT Generator. Missing JDBC drivers in the lib directory can lead to one of the most common errors encountered.

2.2.2 Understanding the Generated Connector Package

The connector package is a ZIP file that is generated in the `/dbat-generator-RELEASE_NUMBER` directory. For example, if you have specified ACME as the value of the `itResourceDefName` entry in the `DBATConfiguration.groovy` file, then the

connector package ZIP (ACME.zip) file is generated in the /dbat-generator-11.1.1.6.0/ directory. The directory structure of the connector package is as follows:

```
CONNECTOR_PACKAGE/  
  bundle/  
    org.identityconnectors.databasetable-1.2.2.jar  
  configuration/  
    IT_RES_DEF-CI.xml  
  dataset/  
  resources/  
    dbat-generator.properties  
  xml/  
    IT_RES_DEF-ConnectorConfig.xml
```

In this directory structure:

- *CONNECTOR_PACKAGE* is replaced with the name of the IT resource definition specified as the value of the `itResourceDefName` entry in the `DBATConfiguration.groovy` file.
- *IT_RES_DEF* is replaced with the name of the IT resource definition specified as the value of the `itResourceDefName` entry in the `DBATConfiguration.groovy` file.

The following behavior is observed after generation of the connector configuration XML file:

- The length of a field (column) from the target system is not fetched into the process form. Therefore, except for the Unique ID and Password fields, the length of all other data fields (of the String data type) on the process form is always set to 255 characters. The length of the Unique ID and Password fields is set to 40 characters.
- All columns in a database table that are not null are displayed as mandatory process form fields in Oracle Identity Manager.

3

Installing and Configuring the Database Application Tables Connector

The procedure to deploy the DBAT connector is divided across three stages namely preinstallation, installation, and postinstallation.

The procedure to install and configure the connector can be divided into the following stages:

- [Prerequisites for Installing the DBAT Connector](#)
- [Overview of Installing DBAT Connector](#)
- [Postinstallation](#)
- [Upgrading the DBAT Connector](#)

3.1 Prerequisites for Installing the DBAT Connector

Prerequisite for the DBAT connector involves creating a target system user account and configuring the database.

Perform the following preinstallation procedures on your target system:

- [About Creating the Target System User Account for Connector Operations](#)
- [About Configuring IBM DB2 Running on IBM z/OS](#)
- [About Configuring Oracle Database](#)

3.1.1 About Creating the Target System User Account for Connector Operations

Oracle Identity Manager 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.

3.1.2 About Configuring IBM DB2 Running on IBM z/OS

During a provisioning operation, the connector runs Java stored procedures to perform the required action on the target system. If your IBM DB2 installation is running on IBM z/OS, then you must configure the WLM to enable the running of these stored procedures. See IBM z/OS documentation for detailed information about configuring the WLM.

3.1.3 About Configuring Oracle Database

Note:

This is an optional procedure. Perform this procedure on an Oracle database table only if you want an autoincrementing primary key.

At any time after creating the Oracle database table, you can set up an autoincrementing primary key column for that database table. To set the autoincrementing primary key, create a sequence, and then create a trigger that inserts a unique autogenerated number in the primary key field while inserting a new record into the parent table. The following is a trigger that you can use:

```
CREATE OR REPLACE TRIGGER trigger_name
BEFORE INSERT ON table_name FOR EACH ROW
BEGIN
SELECT sequence_name.nextval INTO :new.primaty_Key_column_name FROM DUAL;
END;
```

3.2 Overview of Installing DBAT Connector

You can run the connector code locally in Oracle Identity Manager or remotely in a Connector Server.

Depending on where you want to run the generated connector, the connector provides the following installation options:

- To run the connector code locally in Oracle Identity Manager, perform the procedure described in [Installing the Connector in Oracle Identity Manager](#).
- To run the connector code remotely in a Connector Server, perform the procedures described in [Installing the Connector in Oracle Identity Manager](#) and [About Deploying the Connector Bundle in a Connector Server](#).

3.2.1 Installing the Connector in Oracle Identity Manager

In this scenario, you install the connector in Oracle Identity Manager using the Connector Installer.

 **Note:**

- In this guide, the term **Connector Installer** has been used to refer to the Connector Installer feature of the Oracle Identity Manager Administrative and User Console.
- If this version of the connector has been installed earlier and you want to install it for another target system whose libraries need to be added, then:
 1. Download the connector bundle JAR file from the Oracle Identity Manager database by using the Download JARs utility.
 2. Extract the contents of the JAR file and copy the JDBC drivers for your target system to the lib directory. See the "JDBC drivers" row of [Table 1-1](#) to determine the JDBC drivers for your target system.
 3. Run the Upload JARs utility to upload the JAR file to the Oracle Identity Manager database.
 4. Run the PurgeCache utility to clear content related to connector bundle JARs from the server cache. See [Clearing Content Related to Connector Resource Bundles from the Server Cache](#) for information about running the PurgeCache utility.

You must install the connector package (generated after running the DBAT Generator) by running the connector installer. To do so:

1. Copy the unzipped connector package (generated in [Discover the Schema and Generate the Connector](#)) to the following directory:

OIM_HOME/server/ConnectorDefaultDirectory

2. Create a directory in *OIM_HOME*/ConnectorDefaultDirectory/targetsystems-lib with the same name as the installer package. For example:

OIM_HOME/server/ConnectorDefaultDirectory/targetsystems-lib/dbat-11.1.1.6.0

Copy the JDBC driver to this directory. See [Table 1-1](#) to know the JDBC driver corresponding to your target system.

 **Note:**

- If you are using Oracle Database or Oracle RAC as the target system, then no JDBC driver is needed. You can skip this step and proceed to the next.
- During connector installation, all JDBC drivers located in the *OIM_HOME/server/ConnectorDefaultDirectory/targetsystems-lib/dbat-11.1.1.6.0* directory are loaded into the Oracle Identity Manager database and then registered with the Database Application Tables connector.

If this step is skipped for any reason, then you must manually copy the JDBC drivers into the lib directory of the connector JAR file, and then run the Oracle Identity Manager Upload JARs utility to post the connector JAR file to the Oracle Identity Manager database. See *Migrating JARs and Resource Bundle* in *Oracle Fusion Middleware Developing and Customizing Applications for Oracle Identity Manager* for information about running the Upload JARs utility.

3. Log in to Oracle Identity System Administration.
4. In the Manage Connector page, click **Install**.
5. From the Connector List, select the name of the connector package (generated by running the DBAT Generator). This list displays the names and release numbers of connectors whose installation files you copy into the default connector installation directory in Step 1.

If you have copied the installation files into a different directory, then:

- a. In the **Alternative Directory** field, enter the full path and name of that directory.
 - b. To repopulate the list of connectors in the Connector List list, click **Refresh**.
 - c. From the Connector List list, select the name of the connector package.
6. Click **Load**.
 7. To start the installation process, click **Continue**.

The following tasks are performed in sequence:

- a. Configuration of connector libraries
- b. Import of the connector XML files (Using Deployment Manager).
- c. Compilation of Adapter Definitions

On successful completion of a task, a check mark is displayed for the task. If a task fails, then an X mark and a message stating the reason for failure are displayed. Depending on the reason for the failure, make the required correction and then perform one of the following steps:

- Retry the installation by clicking **Retry**.
- Cancel the installation and begin again from Step 1.

If all three tasks of the connector installation process are successful, then a message indicating successful installation is displayed.

When you run the Connector Installer, it processes the script in the DBAT-CI.xml file located in the configuration directory. This file is listed in [Table C-1](#).

3.2.2 About Deploying the Connector Bundle in a Connector Server

You can deploy the Database Application Tables connector either locally in Oracle Identity Manager or remotely in the Connector Server. A **connector server** is a separate Oracle application that enables remote execution of an Identity Connector, such as the Database Application Tables connector. Running a connector on the connector server allows you to pass provisioning and reconciliation requests through the firewalls in a manner defined by the connector server.

The procedure to deploy the connector bundle in a connector server is divided into the following stages:

- [Installing and Configuring the Connector Server](#)
- [Running the Connector Server](#)
- [Installing the Connector on the Connector Server](#)

3.2.2.1 Installing and Configuring the Connector Server

Connector servers are available in two implementations:

- As a .Net implementation that is used by Identity Connectors implemented in .Net framework
- As a Java Connector Server implementation that is used by Java-based Identity Connectors

The Database Application Tables connector is implemented in Java, so you can deploy this connector to a Java Connector Server.

Use the following steps to install and configure the Java Connector Server:

Note:

Before you deploy the Java Connector Server, ensure that you install the JDK or JRE on the same computer where you are installing the Java Connector Server and that your `JAVA_HOME` or `JRE_HOME` environment variable points to this installation.

1. Create a new directory on the computer where you want to install the Java Connector Server.

Note:

In this section, `CONNECTOR_SERVER_HOME` represents this directory.

2. Unzip the Java Connector Server package in the new directory created in Step 1. You can download the Java Connector Server package from Oracle Technology Network.

- Open the ConnectorServer.properties file located in the `conf` directory. In the ConnectorServer.properties file, set the following properties, as required by your deployment.

Property	Description
<code>connectorserver.port</code>	Port on which the Java Connector Server listens for requests. Default is: 8759.
<code>connectorserver.bundleDir</code>	Directory where the connector bundles are deployed. Default is: <code>bundles</code> .
<code>connectorserver.libDir</code>	Directory in which to place dependent libraries. Default is: <code>lib</code> .
<code>connectorserver.usessl</code>	If set to <code>true</code> , the Java Connector Server uses SSL for secure communication. Default is: <code>false</code> . If you specify <code>true</code> , use the following options on the command line when you start the Java Connector Server: <ul style="list-style-type: none"> <code>-Djavax.net.ssl.keyStore</code> <code>-Djavax.net.ssl.keyStoreType</code> (<i>optional</i>) <code>-Djavax.net.ssl.keyStorePassword</code>
<code>connectorserver.ifaddress</code>	Bind address. To set this property, uncomment it in the file (if necessary). The bind address can be useful if there are more NICs installed on the computer.
<code>connectorserver.key</code>	Java Connector Server key.

- Set the properties in the ConnectorServer.properties file, as follows:
 - To set the `connectorserver.key`, run the Java Connector Server with the `/setKey` option.

 **Note:**

For more information, on running the Java Connector Server, see [Running the Connector Server](#).

- For all other properties, edit the ConnectorServer.properties file manually.
- The `conf` directory also contains the `logging.properties` file, which you can edit if required by your deployment.

 **Note:**

Oracle Identity Manager has no built-in support for connector servers, so you cannot test your configuration.

- Copy the connector bundle (the `bundle/org.identityconnectors.databasetable-1.2.2` file of the installation media) to the `/bundle` directory of the connector server. See [Table C-1](#) for information about the connector bundle and other files in the installation media.

7. Copy the JDBC driver for the target system that you are using to the /lib directory of the connector server. See [Table 1-1](#) for information about the relevant JDBC driver for your target system.

3.2.2.2 Running the Connector Server

To run the Java Connector Server on Microsoft Windows, Solaris, Linux, see Using the Java Connector Server in *Oracle Fusion Middleware Developing and Customizing Applications for Oracle Identity Manager*.

3.2.2.3 Installing the Connector on the Connector Server

To install the connector into the Java connector server, see Installing the Connector in the Connector Server in *Oracle Fusion Middleware Developing and Customizing Applications for Oracle Identity Manager*.

3.3 Postinstallation

These are the tasks that you must perform after installing the DBAT connector.

This section discusses the following postinstallation procedures:

- [Configuring the IT Resource for the Target System](#)
- [Configuring the Connector for Date Format](#)
- [Configuring the Connector for a Target System with an Autoincrement Primary Key](#)
- [About Configuring Oracle Identity Manager](#)
- [Localizing Field Labels in UI Forms](#)
- [Clearing Content Related to Connector Resource Bundles from the Server Cache](#)
- [Managing Logging](#)
- [Using Lookup Definitions](#)
- [Setting Up Process Form Fields as Entitlements](#)
- [Configuring Process Form Fields as Date Fields](#)
- [About Configuring Secure Communication Between the Target System and Oracle Identity Manager](#)
- [Configuring Secure Communication Between the Connector Server and Oracle Identity Manager](#)
- [Configuring the Connector for Stored Procedures and Groovy Scripts](#)

3.3.1 Configuring the IT Resource for the Target System

The IT resource for the target system contains connection information about the target system. Oracle Identity Manager uses this information during provisioning and reconciliation.

When you run the DBAT Generator, the IT resource corresponding to this connector is automatically created in Oracle Identity Manager. You must specify values for the parameters of this IT resource as follows:

1. Log in to Oracle Identity System Administration.

2. In the left pane, under Configuration, click **IT Resource**.
3. In the IT Resource Name field on the Manage IT Resource page, enter the name of the IT resource, and then click **Search**. The name of the IT resource is the value of the `itResourceName` property in the `DBATConfiguration.groovy` file.
4. Click the edit icon for the IT resource.
5. From the list at the top of the page, select **Details and Parameters**.
6. Specify values for the parameters of the IT resource. [Table 3-1](#) describes each parameter.

 **Note:**

The IT resource parameters (except for Password) described in [Table 3-1](#) are pre-populated with values you have specified for the corresponding properties while performing the procedure described in [Overview of Configuring the Groovy File](#). You must specify a value for the Password IT resource parameter. For the rest of the IT resource parameters, you can verify the existing values and make changes if required.

Table 3-1 IT Resource Parameters

Parameter	Description
<code>jdbcDriver</code>	JDBC driver class name.
<code>jdbcUrlTemplate</code>	JDBC URL template of the target database. The value that you specify depends on the database product that you are using. See Determining the Value for the jdbcUrlTemplate Property for more information.
<code>host</code>	Host name or IP address of the computer hosting the target system. Sample value: <code>myhost</code>
<code>port</code>	Enter the number of the port at which the target system database is listening.
<code>database</code>	Name of the target database.
<code>user</code>	User ID of the database user account that Oracle Identity Manager uses to connect to the target system.
<code>password</code>	Password of the database user account that Oracle Identity Manager uses to connect to the target system.
Configuration Lookup	Name of the lookup definition that holds connector configuration entries that are used during connector operations.
<code>datasource</code>	Data source name for the data source naming properties. Sample Value: <code>jdbc/operationsDB</code>
<code>jndiProperties</code>	Properties used to establish a connection with the target system by using JDBC drivers, enable additional connection properties, or look up a <code>DataSource</code> using JNDI. Sample value: <code>"java.naming.factory.initial=weblogic.jndi.WLInitialContextFactory", "java.naming.provider.url=t3://acme.com:15000", "java.naming.security.principal=weblogic", "java.naming.security.credentials=weblogic1"</code>

Table 3-1 (Cont.) IT Resource Parameters

Parameter	Description
table	Name of the parent table or view that contains user records.
keyColumn	Name of the column that uniquely identifies each row in the parent table.
passwordColumn	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. See Table 2-1 for more information about the passwordColumn property.
changeLogColumn	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 About 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 most youngest record reconciled from the target system. Note: You must specify a value for this property if you want to perform incremental reconciliation.
quoting	Column quoting property (such as <code>None</code> , <code>Single</code> , <code>Double</code> , <code>Back</code> , or <code>Brackets</code>) 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.
enableEmptyString	Set to <code>true</code> if you want to enable support for writing an empty string instead of a NULL value. Set to <code>false</code> , if empty strings must be written as NULL values. Note: This property can be applied only to mandatory String attributes.
rethrowAllSQLExceptions	Set to <code>false</code> , 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 <code>true</code> .
nativeTimestamps	If the value of this property is set to <code>false</code> , 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 <code>true</code> , then timestamp data is retrieved as <code>java.sql.Timestamp</code> type, and then the connector performs the conversion.
allNative	If value of this property is <code>false</code> , then attribute data is converted to Strings by using the JDBC driver. Set the value of this property to <code>true</code> 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.
validConnectionQuery	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: <pre>SELECT 1 FROM DUMMY</pre>
relationTables	A comma-separated list of child table names when user data is spread across parent and child tables.

Table 3-1 (Cont.) IT Resource Parameters

Parameter	Description
statusColumn	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: <ul style="list-style-type: none"> - 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.
customizedQuery	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.
Connector Server	Name of the connector server IT resource.
dateFormat	Allows the user to format how date data is converted to strings. <ul style="list-style-type: none"> • If you want to handle date data as a date editor, then do <i>not</i> 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. See Configuring the Connector for Date Format for configuration procedure for handling date data as date editor or text.
enableValue	Value used on the target system that depicts that a user record is in the enabled status.
disable value	Value used on the target system that depicts that a user record is in the disabled status.
timestampFormat	Allows the user to format how timestamp data is converted to strings. Specifying this property invalidates the nativeTimestamps and allNative properties.

7. To save the values, click **Update**.

3.3.2 Configuring the Connector for Date Format

The connector enables you to configure the manner in which date data must be handled. You can handle date data either as a date editor or text.

To handle date data as a date editor:

1. Ensure that no value is entered for the dateFormat parameter of the IT resource.
2. Log in to the Design Console.
3. In the process form, change all date parameters to denote date editor.
4. Change the process form field data type from string to date.
5. In the Lookup.RESOURCE.UM.ProvAttrMap and Lookup.RESOURCE.UM.ReconAttrMap lookup definitions, add the [DATE] tag for all lookup entries corresponding to date fields.
6. In the resource object, update the date fields by setting the reconciliation data field type to **Date**.
7. On the Reconciliation Field Mappings tab of the process definition, add field mappings for date fields.

8. On the Object Reconciliation tab of the resource object, click **Create Reconciliation Profile** to copy changes made to the resource object into the MDS.

To handle date data as text:

1. Ensure that you enter a value for the dateFormat parameter of the IT resource.
2. Log in to the Design Console.
3. Ensure that the Lookup.RESOURCE.UM.ProvAttrMap and Lookup.RESOURCE.UM.ReconAttrMap lookup definitions do *not* include the [DATE] tag for lookup entries corresponding to date fields.
4. In the resource object, verify that the date fields are of String data type. If they are not, then set the reconciliation data field type of the date fields to **String**.
5. On the Reconciliation Field Mappings tab of the process definition, verify that the field mappings for date fields contain String data type and not Date.

3.3.3 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.

3.3.4 About Configuring Oracle Identity Manager

 **Note:**

Perform the procedures described in this section *only* if you are using the connector in the target resource configuration mode.

You must create a UI form and an application instance for the resource against which you want to perform reconciliation and provisioning operations. In addition, you must run entitlement and catalog synchronization jobs. These procedures are described in the following sections:

- [Creating and Activating a Sandbox](#)
- [Creating a New UI Form](#)
- [Creating an Application Instance](#)
- [Publishing a Sandbox](#)
- [Harvesting Entitlements and Sync Catalog](#)

3.3.4.1 Creating and Activating a Sandbox

See *Managing Sandboxes in Oracle Fusion Middleware Developing and Customizing Applications for Oracle Identity Manager* for instructions on creating and activating a sandbox.

3.3.4.2 Creating a New UI Form

See *Managing Forms in Oracle Fusion Middleware Administering Oracle Identity Manager* for instructions on creating a new UI form. While creating the UI form, ensure that you select the resource object corresponding to the DBAT connector that you want to associate the form with. In addition, select the **Generate Entitlement Forms** check box.

3.3.4.3 Creating an Application Instance

See *Managing Application Instances in Oracle Fusion Middleware Administering Oracle Identity Manager* for instructions on creating an application instance. While creating the application instance, ensure that you select the form created in [Creating a New UI Form](#).

After creating the application instance, you must publish it to an organization to make the application instance available for requesting and subsequent provisioning to users. However, as a best practice, perform the following procedure before publishing the application instance:

1. In the System Administration console, deactivate the sandbox.
2. Log out of the System Administration console.
3. Log in to the Self Service console and activate the sandbox that you deactivated in Step 1.
4. In the Catalog, check for the Application Instance UI (form fields) and ensure that it appears correctly.
5. Publish the application instance only if everything appears correctly. Otherwise, fix the issues and then publish the application instance.

See *Managing Organizations Associated With Application Instances in Oracle Fusion Middleware Administering Oracle Identity Manager* for instructions on publishing an application instance to an organization.

3.3.4.4 Publishing a Sandbox

Before you publish a sandbox, perform the following procedure as a best practice to validate all sandbox changes made till this stage as it is hard to revert changes once a sandbox is published:

1. In the System Administration console, deactivate the sandbox.
2. Log out of the System Administration console.
3. Log in to the Self Service console using the xelsysadm user credentials and then activate the sandbox that you deactivated in Step 1.
4. In the Catalog, ensure that the DBAT application instance form appears with correct fields.
5. Publish the sandbox. See *Publishing a Sandbox in Oracle Fusion Middleware Developing and Customizing Applications for Oracle Identity Manager* for instructions on publishing a sandbox.

3.3.4.5 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 Manager* 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 Manager* for more information about this scheduled job.

3.3.5 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 one of the following files in a text editor:
 - For Oracle Identity Manager 11g Release 2 PS2 (11.1.2.2.0) or later releases:
SAVED_LOCATION\xliffBundles\oracle\iam\ui\runtime\BizEditorBundle_en.xlf
 - For releases prior to Oracle Identity Manager 11g Release 2 PS2 (11.1.2.2.0):
SAVED_LOCATION\xliffBundles\oracle\iam\ui\runtime\BizEditorBundle.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 7.7.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 7.7.a through 7.7.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 Manager* for more information about exporting and importing metadata files

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

3.3.6 Clearing Content Related to Connector Resource Bundles from the Server Cache

When you deploy the connector, the resource bundles are copied from the resources directory on the installation media into the Oracle Identity Manager database. Whenever you add a new resource bundle to the connectorResources directory or make a change in an existing resource bundle, you must clear content related to connector resource bundles from the server cache.

To clear content related to connector resource bundles from the server cache you can either restart Oracle Identity Manager or run the PurgeCache utility. The following is the procedure to clear the server cache by running the PurgeCache utility:

1. In a command window, switch to the *OIM_HOME/server/bin* directory.
2. Enter one of the following commands:
 - On Microsoft Windows: `PurgeCache.bat All`
 - On UNIX: `PurgeCache.sh All`

When prompted, enter the user name and password of an account belonging to the SYSTEM ADMINISTRATORS group. In addition, you are prompted to enter the service URL in the following format:

```
t3://OIM_HOST_NAME:OIM_PORT_NUMBER
```

In this format:

- Replace *OIM_HOST_NAME* with the host name or IP address of the Oracle Identity Manager host computer.
- Replace *OIM_PORT_NUMBER* with the port on which Oracle Identity Manager is listening.

You can use the PurgeCache utility to purge the cache for any content category.

3.3.7 Managing Logging

Managing logging is discussed in the following sections:

- [Understanding Log Levels](#)
- [Enabling Logging](#)

3.3.7.1 Understanding Log Levels

Oracle Identity Manager release uses Oracle Java Diagnostic Logging (OJDL) for logging. OJDL 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 Manager 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 log levels are mapped to ODL message type and level combinations as shown in [Table 3-2](#).

Table 3-2 Log Levels and ODL Message Type:Level Combinations

Log Level	ODL Message Type:Level
<code>SEVERE.intValue()+100</code>	<code>INCIDENT_ERROR:1</code>
<code>SEVERE</code>	<code>ERROR:1</code>
<code>WARNING</code>	<code>WARNING:1</code>
<code>INFO</code>	<code>NOTIFICATION:1</code>
<code>CONFIG</code>	<code>NOTIFICATION:16</code>
<code>FINE</code>	<code>TRACE:1</code>
<code>FINER</code>	<code>TRACE:16</code>
<code>FINEST</code>	<code>TRACE:32</code>

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 Manager.

3.3.7.2 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 3-2](#) 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 Manager, 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.

3.3.8 Using Lookup Definitions

Note:

This is an optional procedure. Perform this procedure only if you want to use lookup definitions as the input source for some of the fields on the process form during provisioning operations.


If you are configuring the connector for provisioning, then you may want to create lookup fields on the process form. For example, during provisioning operations, you may want to select the Country Code value from a lookup field. After deploying the connector, you can set up this field as a lookup field by specifying an input source for the field.

You can use a lookup definition as the input source. For example, you can create a lookup definition containing country codes and then set up the lookup definition as the input source for the Country field. If you want to use a lookup definition as the input source, then perform the following steps:

See Also:

Adding or Editing Fields in Data Sets in *Oracle Fusion Middleware Administering Oracle Identity Manager* for detailed information about each of the following steps

1. Log in to the Design Console.
2. Create an empty lookup definition that must be used as an input source. Skip this step if you want to use the Lookup.DBAT.Example lookup definition, which is created after you install the connector.
3. Update the process form to change the data field from text to lookup field. To do so:
 - a. Expand **Development Tools** and double-click **Form Designer**.

- b. Search for and open the process form that contains the required information to provision resources to a target system user account. The name of the process form is in the following format:
`UD_[RESOURCE]_[TABLE_ALIAS]`
 - c. Click **Create New Version**.
A new version of the form is created for editing.
 - d. On the Additional Columns tab, in the row for the data field that you want to set up as a lookup field, change the value of the Field Type column from TextField to **LookupField**.
 - e. On the Properties tab, select the data field (whose Field Type was changed from TextField to Lookup Field in the preceding step), and then click **Add Property**.
 - f. In the Add Property dialog box:
 - From the Property Name list, select **Lookup Code**.
 - In the Property Value field, enter the name of the lookup definition to be used as the input source (the one used in Step 2).
-  **Note:**

If you want to set up the lookup field for entitlement, then from the Property Name list, select **Entitlement** and set its property value to `true`.
- g. Click the Save icon and then close the dialog box.
 - h. Click the Save icon on the process form.
 - i. Click **Make Version Active**.
 - j. You must create and activate a new version of parent form as well to include the latest version of the child form.
4. Update the lookup entry to include information that specifies the field is a lookup field. To do so:
 - a. Expand **Administration** and double-click **Lookup Definition**.
 - b. Search for and open the **Lookup.RESOURCE.UM.ProvAttrMap** lookup definition.
 - c. In the lookup entries, search for the field name in the Code Key column that has been changed to a lookup field, and then suffix the field name with `[LOOKUP]`. This denotes that the process form field is a lookup field.
 - d. Click the Save icon.
 - e. Repeat Steps 4.4.b through 4.4.d to suffix the corresponding Code Key entry with `[LOOKUP]` in the lookup definition that holds reconciliation attribute mappings:

While performing Step 4.4.c, search for and open the **Lookup.RESOURCE.UM.ReconAttrMap** lookup definition.

This completes the procedure for setting up a lookup definition as an input source.

3.3.9 Setting Up Process Form Fields as Entitlements

If a child process form field has to be configured as a entitlement, then you must associate it with the corresponding lookup definition, and set the entitlement property to true. If the corresponding lookup definition does not exist, then you must create it manually. See *Marking Entitlement Attributes on Child Process Forms* in *Oracle Fusion Middleware Administering Oracle Identity Manager* for more information.

3.3.10 Configuring Process Form Fields as Date Fields

By default, after metadata generation, all date fields are displayed as text fields on the process form. If you want to display these fields as a Date picker, then you must perform the following steps:

 **Note:**

See *Step 3: Modify Connector Configuration Page* in *Oracle Fusion Middleware Administering Oracle Identity Manager* for detailed information on performing each of the steps discussed in this procedure.

1. In the Design Console, search for and open the process form containing the fields that must be displayed as Date picker.
2. On the Reconciliation Field Mappings tab, delete the field mappings for the date field that is being displayed as String.
3. In the parent form, delete the entries corresponding to the field that must be displayed as the Date picker.
4. In the Resource Object, change the reconciliation field type from String to **Date**.
5. In the process form, on the Reconciliation Field Mappings tab, add the field mapping for the date field.

3.3.11 About Configuring Secure Communication Between the Target System and Oracle Identity Manager

 **Note:**

It is recommended that you perform the procedure described in this section to secure communication between the target system and Oracle Identity Manager.

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

- [Configuring Secure Communication Between IBM DB2 and Oracle Identity Manager](#)

- [Configuring Secure Communication Between Microsoft SQL Server and Oracle Identity Manager](#)
- [Configuring Secure Communication Between MySQL and Oracle Identity Manager](#)
- [Configuring Secure Communication Between Oracle Database and Oracle Identity Manager](#)
- [Configuring Secure Communication Between Sybase Adaptive Server Enterprise and Oracle Identity Manager](#)

3.3.11.1 Configuring Secure Communication Between IBM DB2 and Oracle Identity Manager

Note:

- IBM DB2 version 9.1 Fix Pack 2 and later support secure communication over SSL.
- Before configuring secure communication between IBM DB2 and Oracle Identity Manager, 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 Manager.

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

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

For IBM DB2 version 9.5:

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

For versions other than IBM DB2 9.5:

```
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 welcome1 -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 welcome1 -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 welcome1 -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. If you are using IBM DB2 Version 9.5, then create the `SSLconfig.ini` file, and then add and set values for the SSL parameters. See the IBM DB2 Version 9.5 documentation for more information.
7. If you are using IBM DB2 Version 9.7 then, set the SSL configuration parameters and the `DB2COMM` registry. See the IBM DB2 Version 9.7 documentation for more information.
8. Verify your SSL settings by running the `db2 GET DATABASE MANAGER CONFIGURATION` command.
9. Import the certificate into the Java keystore of the application server on which Oracle Identity Manager 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 changeit -
keystore C:\Users\example_user\.keystore
```

The certificate is imported into the keystore.

3.3.11.2 Configuring Secure Communication Between Microsoft SQL Server and Oracle Identity Manager

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

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 Manager.
Export the certificate on the Microsoft SQL Server host computer.
2. Copy the certificate to the Oracle Identity Manager host computer.
3. Import the certificate into the JVM truststore of the application server on which Oracle Identity Manager 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`

3.3.11.3 Configuring Secure Communication Between MySQL and Oracle Identity Manager

To configure secure communication between MySQL and Oracle Identity Manager:

1. See MySQL documentation for information about enabling SSL communication between MySQL and a client system. In this context, the client is Oracle Identity Manager.
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 Manager host computer.
5. Import the certificates into the JVM truststore of the application server on which Oracle Identity Manager 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 Manager cluster, you must import the file into the truststore on each node of the cluster.

3.3.11.4 Configuring Secure Communication Between Oracle Database and Oracle Identity Manager

To secure communication between Oracle Database and Oracle Identity Manager, 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](#)

3.3.11.4.1 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.

3.3.11.4.2 Configuring SSL Communication in Oracle Database

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

 **Note:**

See [Enabling Secure Sockets Layer in *Oracle Database Security Guide*](#) for detailed information about enabling SSL communication between Oracle Database and Oracle Identity Manager.

1. Export the certificate on the Oracle Database host computer.
2. Copy the certificate to Oracle Identity Manager.
3. Import the certificate into the JVM truststore of the application server on which Oracle Identity Manager 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 Manager cluster, you must import the file into the truststore on each node of the cluster.

3.3.11.5 Configuring Secure Communication Between Sybase Adaptive Server Enterprise and Oracle Identity Manager

To configure secure communication between Sybase Adaptive Server Enterprise and Oracle Identity Manager:

1. Refer to Sybase Adaptive Server Enterprise documentation for information about enabling SSL communication between Sybase Adaptive Server Enterprise and a client system. In this context, the client is Oracle Identity Manager.

Export the certificate on the Sybase Adaptive Server Enterprise host computer.

2. Copy the certificate to the Oracle Identity Manager host computer.
3. Import the certificate into the JVM truststore of the application server on which Oracle Identity Manager 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

3.3.12 Configuring Secure Communication Between the Connector Server and Oracle Identity Manager

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 Manager. The procedure to configure secure communication is the same as the procedure described in section [About Configuring Secure Communication Between the Target System and Oracle Identity Manager](#). 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.

3.3.13 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

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 Manager host computer, create Groovy scripts that call the relevant stored procedures on the target system to perform provisioning operations. 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.

The following is a sample Groovy script that calls the `DELETE_USERGROUP` and `DELETE_USERROLE` stored procedure created in Step 1:

```
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();
};
};

```

3. Update the configuration lookup definition to include information about the Groovy scripts as follows:

 **Note:**

Perform the procedure described in this step only if you want to configure the connector for stored procedures and you have not entered values for script-related properties such as `createScript`, `executeQueryScript`, `lookupScript`, and so on in the `DBATConfiguration.groovy` file.

- a. In the Design Console, expand **Administration**, and double-click **Lookup Definition**.
- b. Search for and open the **Lookup.Configuration.RESOURCE** lookup definition.
- c. Click **Add**.
- d. In the newly added row, depending on the reconciliation or provisioning operation you want to perform, add one or all of the following lookup entries:

Table 3-3 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.

 **Note:**

Instead of the file URL of the Groovy script, you can directly enter the Groovy script in the Decode column. In such a case, ensure that the corresponding Code Key value does not contain [LOADFROMURL]. For example, if you directly enter the Groovy script for the create user account provisioning operation, then the corresponding code key entry 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
```

- e. Click the Save icon.
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();
```

This completes the procedure for configuring your connector to use stored procedure for provisioning operations.

3.4 Upgrading the DBAT Connector

You can upgrade the connector from release 11.1.1.5.0 to the current release.

If you want to upgrade the connector from release 11.1.1.5.0 to this release of the connector, then you must update the connector bundle JAR file. No other configuration procedures are required.

Already installed connectors need not be upgraded as the artifacts are generated dynamically based on schema. However, if you want to upgrade to the latest functionality provided by the connector, then you need to update the JAR file.

Note:

Before you perform the upgrade procedure:

- It is strongly recommended that you create a backup of the Oracle Identity Manager database. Refer to the database documentation for information about creating a backup.
- As a best practice, perform the upgrade procedure in a test environment initially.

To update the connector bundle JAR:

1. Download the latest version of this connector from Oracle Technology Network and extract its contents to any directory on the computer hosting Oracle Identity Manager.
2. Run the Delete JARs utility to remove the `org.identityconnectors.databasetable-1.2.2` file from the Oracle Identity Manager database. This utility is copied into the following location when you install Oracle Identity Manager:

For Microsoft Windows:

```
OIM_HOME/server/bin/DeleteJars.bat
```

For UNIX:

OIM_HOME/server/bin/DeleteJars.sh

When you run the utility, you are prompted to enter the login credentials of the Oracle Identity Manager administrator, URL of the Oracle Identity Manager host computer, context factory value, type of JAR file being deleted, and the name of the JAR file to be deleted.

3. Run the Upload JARs utility to post the latest version of the `org.identityconnectors.databasetable-1.2.2` file from the `/bundle` directory of the installation media to the Oracle Identity Manager database.

For Microsoft Windows:

OIM_HOME/server/bin/UploadJars.bat

For UNIX:

OIM_HOME/server/bin/UploadJars.sh

When you run the utility, you are prompted to enter the login credentials of the Oracle Identity Manager administrator, URL of the Oracle Identity Manager host computer, context factory value, type of JAR file being uploaded, and the location from which the JAR file is to be uploaded.

4. Run the PurgeCache utility to clear content related to connector bundle JARs from the server cache. See [Clearing Content Related to Connector Resource Bundles from the Server Cache](#) for information about running the PurgeCache utility.

4

Using the Database Application Tables Connector

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

Topics:

- [Guidelines to Apply While Using the DBAT Connector](#)
- [Overview of Lookup Definitions Used During Connector Operations](#)
- [Understanding Reconciliation Scheduled Jobs](#)
- [About Configuring Reconciliation for DBAT Connector](#)
- [Performing Provisioning Operations](#)
- [Configuring Action Scripts](#)
- [About Uninstalling the DBAT Connector](#)

4.1 Guidelines to Apply While Using the DBAT Connector

Before using the DBAT connector, ensure that the lookup definitions is synchronized and for a trusted source reconciliation, the date formats set in both the target system and Oracle Identity Manager are the same.

Apply the following guidelines while using the connector:

- Before a target resource reconciliation run is performed, lookup definitions must be synchronized with the child tables of the target system. In other words, scheduled tasks for lookup field synchronization must be run before user reconciliation runs.
- If you have configured the connector for trusted source reconciliation, then ensure that the date formats set in both the target system and Oracle Identity Manager are the same. To ensure that the date formats match:
 1. Check the date format set on Oracle Identity Manager. To do so:
 - a. Log in to the Administrative and User Console.
 - b. In the Welcome page of Oracle Identity Manager Administration, under System Management, click **System Configuration**. Alternatively, you can click the System Management tab, and then click **System Configuration**.
 - c. Search for and open the **Default Date Format** system property.
 - d. On the System Property Detail page, note the date format displayed in the Value field.
 2. In the `DBATConfiguration.groovy` file, ensure that the value of the `timestampFormat` property is the same as the date format in Step 1.1.d.

4.2 Overview of Lookup Definitions Used During Connector Operations

Know more about the lookup definitions used during connector operations

It can be categorized as follows:

- [About Predefined Lookup Definitions](#)
- [Understanding Custom Lookup Definitions Synchronized with the Target System](#)

4.2.1 About Predefined Lookup Definitions

This section discusses the lookup definitions that are created in Oracle Identity Manager after you deploy the connector. These lookup definitions are either prepopulated with values or values must be manually entered in them after the connector is deployed. In addition, you can customize entries in the lookup definitions to suit your requirements. This section discusses the following lookup definitions:

- [Lookup.Configuration.RESOURCE](#)
- [Lookup.RESOURCE.UM.Configuration](#)
- [Lookup.RESOURCE.UM.ReconAttrMap](#)
- [Lookup.RESOURCE.UM.ProvAttrMap](#)
- [Lookup.RESOURCE.UM.ReconAttrMap.Defaults](#)



Note:

RESOURCE has been used as a place holder text for IT resource name. Therefore, replace all instances of *RESOURCE* in this guides with the value that you specified for the `itResourceName` entry in the `DBATConfiguration.groovy` file. See [Entries in the Predefined Sections](#) for more information about entries in the `DBATConfiguration.groovy` file.

4.2.1.1 Lookup.Configuration.RESOURCE

The `Lookup.Configuration.RESOURCE` lookup definition holds connector configuration entries that are used during reconciliation (both trusted source and target resource) and provisioning operations.

[Table 4-1](#) lists the entries in this lookup definition.

Table 4-1 Entries in the Lookup.Configuration.RESOURCE Lookup Definition

Code Key	Decode	Description
Bundle Name	org.identityconnectors.data.basetable	This entry holds the name of the connector bundle class. Do not modify this entry.

Table 4-1 (Cont.) Entries in the Lookup.Configuration.RESOURCE Lookup Definition

Code Key	Decode	Description
Bundle Version	1.2.2	This entry holds the version of the connector bundle class. Do not modify this entry.
Connector Name	org.identityconnectors.data.basetable.DatabasetableConnector	This entry holds the name of the connector class. Do not modify this entry.
Pool Max Idle	10	This entry holds the maximum number of idle objects in a pool.
Pool Max Size	10	This entry holds the maximum number of connections that the pool can create.
Pool Max Wait	150000	This entry holds the 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	120000	This entry holds the minimum time, in milliseconds, the connector must wait before evicting an idle object.
Pool Min Idle	1	This entry holds the minimum number of idle objects in a pool.
User Configuration Lookup	Lookup.RESOURCE.UM.Configuration	This entry holds the name of the lookup definition that contains configuration information specific to the user object type. See Lookup.RESOURCE.UM.Configuration for more information about this lookup definition.

4.2.1.2 Lookup.RESOURCE.UM.Configuration

The Lookup.RESOURCE.UM.Configuration lookup definition contains entries specific to the user object type. This lookup definition is preconfigured.

[Table 4-2](#) lists the default entries in this lookup definition when you have configured your target system as a target resource.

Table 4-2 Entries in the Lookup.RESOURCE.UM.Configuration Lookup Definition for a Target Resource Configuration

Code Key	Decode
Provisioning Attribute Map	Lookup.RESOURCE.UM.ProvAttrMap
Recon Attribute Map	Lookup.RESOURCE.UM.ReconAttrMap

[Table 4-3](#) lists the default entries in this lookup definition when you have configured your target system as a trusted source.

Table 4-3 Entries in the Lookup.RESOURCE.UM.Configuration Lookup Definition for a Trusted Source Configuration

Code Key	Decode
Recon Attribute Defaults	Lookup.RESOURCE.UM.ReconAttrMap.Defaults
Recon Attribute Map	Lookup.RESOURCE.UM.ReconAttrMap

You can add or modify entries in this lookup definition. For example, you can add an entry in this lookup definition if you want to use the connector for configuring validation of data during reconciliation and provisioning. See [Extending the Functionality of the Database Application Tables Connector](#) for more information on using this lookup definition for transformation and validation.

4.2.1.3 Lookup.*RESOURCE.UM.ReconAttrMap*

The Lookup.*RESOURCE.UM.ReconAttrMap* lookup definition holds mappings between resource object fields and target system attributes. In this connector, the target system attributes correspond to the target system column names. Depending on whether you have configured your target system as a trusted source or target resource, this lookup definition is used during target resource or trusted source user reconciliation runs, respectively.

If you have configured your target system as a target resource:

The following is the format of the Code Key and Decode values in this lookup definition:

- **For single-valued attributes**
 - **Code Key:** Reconciliation attribute of the resource object against which target resource user reconciliation runs must be performed
 - **Decode:** Corresponding connector attribute name or the target system column name

- **For multivalued attributes**

- **Code Key:** *RO_ATTR_NAME~ATTR_NAME[LOOKUP]*

In this format:

- * *RO_ATTR_NAME* specifies the reconciliation field for the child table.
- * *ATTR_NAME* is the name of the multivalued attribute.
- * *[LOOKUP]* is a keyword that is appended to the code key value if the child data is picked from a lookup or declared as an entitlement.

- **Decode:** Combination of the following elements separated by the tilde (~) character:

EMBED_OBJ_NAME~RELATION_TABLE_NAME~ATTR_NAME

In this format:

- * *EMBED_OBJ_NAME* is the name of the object (for example, an account's address) on the target system that is embedded in another object.
- * *RELATION_TABLE_NAME* is the name of child table in the target system.
- * *ATTR_NAME* is the name of the column in the child table corresponding to the multivalued attribute in the Code Key column.

If you have configured your target system as a trusted source:

The following is the format of the Code Key and Decode values in this lookup definition:

- **Code Key:** Reconciliation attribute of the resource object against which trusted source user reconciliation runs must be performed

- **Decode:** Corresponding target system column name

The entries in this lookup definition depend on the data available in the target system. The entries of this lookup definition are populated based on the values specified for the alias entry in the DBATConfiguration.groovy file. See [Entries in the Predefined Sections](#) for more information about the alias entry.

4.2.1.4 Lookup.RESOURCE.UM.ProvAttrMap

The Lookup.RESOURCE.UM.ProvAttrMap lookup definition holds mappings between process form fields and target system column names. This lookup definition is used for performing provisioning operations.

The following is the format of the Code Key and Decode values in this lookup definition:

- **Code Key:** Name of the label on the process form
- **Decode:** Corresponding target system column name

For entries corresponding to child form fields, the following is the format of the Code Key and Decode values:

- **Code Key:** *CHILD_FORM_NAME~FIELD_NAME*

In this format:

- *CHILD_FORM_NAME* specifies the name of the child form.
- *FIELD_NAME* specifies the name of the label on the child form in the Administrative and User Console.

- **Decode:** Combination of the following elements separated by the tilde (~) character:

EMBED_OBJ_NAME~RELATION_TABLE_NAME~COL_NAME

In this format:

- *EMBED_OBJ_NAME* is the name of the object (for example, an account's address) on the target system that is embedded in another object.
- *COL_NAME* is the name of the column in the child table corresponding to the child form specified in the Code Key column.
- *RELATION_TABLE_NAME* is the name of child table in the target system.

The entries in this lookup definition depend on the data available in the target system. The values in the lookup definition are populated based on the value specified for the alias entry in the DBATConfiguration.groovy file. See [Entries in the Predefined Sections](#) for more information about the alias entry.

4.2.1.5 Lookup.RESOURCE.UM.ReconAttrMap.Defaults

The Lookup.RESOURCE.UM.ReconAttrMap.Defaults lookup definition holds default values of the mandatory fields on the OIM User form that are not mapped with the connector attributes. This lookup definition is created only if you have configured your target system as a trusted source.

This lookup definition is used when there is a mandatory field on the OIM User form, but no corresponding column in the target system from which values can be fetched during trusted source reconciliation runs. In addition, this lookup definition is used if the mandatory field on the OIM User form has a corresponding column that is empty or contains null values.

The following is the format of the Code Key and Decode values in this lookup definition:

- **Code Key:** Name of the user field on the Administrative and User Console.
- **Decode:** Corresponding default value to be displayed.

For example, the Role field is a mandatory field on the OIM User form. Suppose the target system contains no column that stores information about the role for a user account. During reconciliation, no value for the Role field is fetched from the target system. However, as the Role field cannot be left empty, you must specify a value for this field. Therefore, the Decode value of the Role Code Key has been set to `Full-Time`. This implies that the value of the Role field on the OIM User form displays Full-Time for all user accounts reconciled from the target system.

[Table 4-4](#) lists the default entries in this lookup definition.

Table 4-4 Entries in the Lookup.RESOURCE.UM.ReconAttrMap.Defaults Lookup Definition

Code Key	Decode
Role	Full-Time
Organization Name	Xellerate Users
Xellerate Type	End-User

4.2.2 Understanding Custom Lookup Definitions Synchronized with the Target System

During a provisioning operation, you use a lookup field on the process form to specify a single value from a set of values. For example, you may want to select a role from a lookup field to specify the role being assigned to the user.

When you deploy the connector, an empty lookup definition (Lookup.RESOURCE.Example) is created. The Lookup.RESOURCE.Example lookup definition is used to store values from a child table that must be displayed in a lookup field during provisioning. Depending upon your environment, you can customize the Lookup.RESOURCE.Example lookup definition to suit your requirement. Alternatively, you can create your own lookup definition for storing values to be displayed in a lookup field. See [Using Lookup Definitions](#) for information about setting up lookup fields.

Lookup field synchronization involves obtaining the most current values from specific tables in the target system to the lookup definitions (used as an input source for lookup fields, for example Lookup.RESOURCE.Example) in Oracle Identity Manager.

The *RESOURCETarget* Lookup Reconciliation scheduled job is used to synchronize values of these lookup definitions with the tables in the target system. While configuring the *RESOURCETarget* Lookup Reconciliation scheduled job, you specify the name of the lookup definition that you want to synchronize as the value of the Lookup Name attribute. See [Scheduled Job for Lookup Field Synchronization](#) for more information about this scheduled task.

After lookup definition synchronization, data is stored in the following format:

- Code Key value: `IT_RESOURCE_KEY~LOOKUP_FIELD_ID`

In this format:

- *IT_RESOURCE_KEY* is the numeric code assigned to each IT resource in Oracle Identity Manager.
- *LOOKUP_FIELD_ID* is the target system code assigned to each lookup field entry. This value is populated based on the column name specified in the Code Key attribute of the *RESOURCE* Lookup Reconciliation scheduled job.

Sample value: 1~SA

- Decode value: *IT_RESOURCE_NAME~LOOKUP_FIELD_ID*

In this format:

- *IT_RESOURCE_NAME* is the name of the IT resource in Oracle Identity Manager.
- *LOOKUP_FIELD_ID* is the target system code assigned to each lookup field entry. This value is populated based on the column name specified in the Decode attribute of the *RESOURCE* Lookup Reconciliation scheduled job.

Sample value: DBAT Lookup~SYS_ADMIN

4.3 Understanding Reconciliation Scheduled Jobs

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

This section discusses the following topics:

- [Scheduled Job for Lookup Field Synchronization](#)
- [About Attributes of the Scheduled Jobs](#)
- [About Configuring Scheduled Jobs for DBAT Connector](#)

4.3.1 Scheduled Job for Lookup Field Synchronization

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

[Table 4-5](#) describes the attributes of the *RESOURCE* Lookup Reconciliation scheduled job. [About Configuring Scheduled Jobs for DBAT Connector](#) describes the procedure to configure scheduled jobs.

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 4-5 Attributes of the RESOURCE Lookup Reconciliation Scheduled Job

Attribute	Description
Code Key Attribute	<p>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:</p> <ul style="list-style-type: none"> When scripts are not being used: <i>TABLE_NAME.COLUMN_NAME</i> Sample value: <code>ROLES.ROLE_ID</code> When scripts are being used, it would be according to the script mentioned in groovy file. Sample value: <code>Code Key Attribute-roleId</code> Where, roleId is the columns in the table on which lookup is being run.
Decode Attribute	<p>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:</p> <ul style="list-style-type: none"> When scripts are not being used: <i>TABLE_NAME.COLUMN_NAME</i> Sample value: <code>ROLES.ROLE_NAME</code> When scripts are being used, it would be according to the script mentioned in groovy file. Sample value: <code>Decode Attribute-roleName</code> Where, roleName is the columns in the table on which lookup is being run.
IT Resource Name	<p>Enter the name of the IT resource for the target system installation from which you want to reconcile records. Default value: <code>DBAT Lookup</code></p>
Lookup Name	<p>Enter the name of the lookup definition in Oracle Identity Manager that must be populated with values fetched from the target system. Default value: <code>Lookup.DBAT.Example</code> Note: Before you perform lookup field synchronization, the lookup definition name that you specify must exist in Oracle Identity Manager.</p>
Object Type	<p>Enter the type of object you want to reconcile. Default value: <code>Other</code> Note: For lookup field synchronization, the object type must be any object other than "User."</p>

4.3.2 About 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](#)

4.3.2.1 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 Manager. 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 4-6](#) describes the attributes of both scheduled jobs.

Table 4-6 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 About 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 <i>RESOURCE</i> Target Resource User Reconciliation scheduled job is <code>RESOURCE Target Resource User Reconciliation</code> . The default value of this attribute in the <i>RESOURCE</i> Trusted User Reconciliation scheduled job is <code>RESOURCETrusted Resource User Reconciliation</code> .

4.3.2.2 Scheduled Jobs for Reconciliation of Deleted Users Records

After you create the connector, the scheduled task for reconciling data about deleted users records is automatically created in Oracle Identity Manager. 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 4-7](#) describes the attributes of both scheduled jobs.

Table 4-7 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

4.3.2.3 Scheduled Jobs for Incremental Reconciliation

While configuring the DBATConfiguration.groovy file, if you have specified a value for the changeLogColumn property, then the scheduled job for incremental reconciliation is automatically created in Oracle Identity Manager when you install the connector. If you did not specify a value for the changeLogColumn property before connector installation, then perform the procedure described in [Configuring the Connector for Incremental Reconciliation](#) to create the scheduled job for incremental reconciliation.

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 4-6](#) describes the attributes of both scheduled jobs.

Table 4-8 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 4-8 (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: <code>RESOURCE Target Incremental Resource User Reconciliation</code>
Sync Token	Depending on the value specified for the <code>changeLogColumn</code> property in the Config entry of the <code>DBATConfiguration.groovy</code> file, this attribute holds one of the following values: <ul style="list-style-type: none"> 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 <i>not</i> date or time stamp based (for example, numeric or strings): This attribute holds the newest or the most recent value of the <code>changeLog</code> column of the record that was last reconciled. Sample value: <code><String>3</String></code> Note: <ul style="list-style-type: none"> - Do <i>not</i> 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.

4.3.2.3.1 Configuring the Connector for Incremental Reconciliation

As discussed earlier, the scheduled job for incremental reconciliation is automatically created in Oracle Identity Manager during connector installation, if you have specified a value for the `changeLogColumn` property while configuring the `DBATConfiguration.groovy` file. If you did not specify a value for the `changeLogColumn` property before installing the connector, you can still configure the connector to create the scheduled job for incremental reconciliation. To do so:

1. In a text editor, open the `DBATConfiguration.groovy` file for editing. This file is located in the `dbat-RELEASE_NUMBER/generator/dbat-generator-RELEASE_NUMBER` directory of the connector installation ZIP.
2. Set a value for the `changeLogColumn` property. See the "changeLogColumn" row of [Table 2-1](#) for information about that values that you can specify for this property.
3. Run the DBAT Generator. See [Discover the Schema and Generate the Connector](#) for information on running the DBAT Generator. The connector package is generated that contains the `IT_RES_DEF-ConnectorConfig.xml` file. This file contains definitions for connector components such as IT resource, lookup definitions, scheduled tasks, process forms, and resource objects.
4. Import the scheduled job and task corresponding to incremental reconciliation from the `IT_RES_DEF-ConnectorConfig.xml` file. To do so:

 **Note:**

See Importing Deployments in *Oracle Fusion Middleware Developing and Customizing Applications for Oracle Identity Manager* for detailed instructions on performing each of the steps discussed in this procedure.

- a. Log in to the System Administration console.
 - b. Add the *IT_RES_DEF-ConnectorConfig.xml* file to the Deployment Manager for import.
 - c. Except for the incremental reconciliation scheduled job and task, remove all other artifacts from the *IT_RES_DEF-ConnectorConfig.xml* file.
 - d. Import the *IT_RES_DEF-ConnectorConfig.xml* file.
5. Update the IT resource by setting the value of the `changeLogColumn` parameter to the value entered in Step 2.

This completes the procedure for importing the scheduled job for incremental reconciliation into Oracle Identity Manager.

4.3.3 About Configuring Scheduled Jobs for DBAT Connector

This section describes the procedure to configure scheduled jobs. You can apply this procedure to configure the scheduled jobs for lookup field synchronization and reconciliation.

- [Scheduled Jobs for Lookup Field Synchronization and Reconciliation](#)
- [Configuring Scheduled Jobs](#)

4.3.3.1 Scheduled Jobs for Lookup Field Synchronization and Reconciliation

[Table 4-9](#) lists the scheduled jobs that you can configure.

Table 4-9 Scheduled Jobs for Lookup Field Synchronization and Reconciliation

Scheduled Task	Description
<i>RESOURCE</i> Lookup Reconciliation	This scheduled job is used for lookup field synchronization. See Scheduled Job for Lookup Field Synchronization for information about this scheduled job.
<i>RESOURCE</i> Target Resource User Reconciliation	This scheduled job is used for user reconciliation when the target system is configured as a target resource. See Scheduled Jobs for Reconciliation of User Records for more information.
<i>RESOURCE</i> Trusted Resource User Reconciliation	This scheduled job is used for user reconciliation when the target system is configured as a trusted source. See Scheduled Jobs for Reconciliation of User Records for more information.
<i>RESOURCE</i> Target Resource User Delete Reconciliation	This scheduled job is used for reconciliation of deleted user records when the target system is configured as a target resource. See Scheduled Jobs for Reconciliation of Deleted Users Records for more information.
<i>RESOURCE</i> Trusted Resource User Delete Reconciliation	This scheduled job is used for reconciliation of deleted user records when the target system is configured as a trusted source. See Scheduled Jobs for Reconciliation of Deleted Users Records for more information.

Table 4-9 (Cont.) Scheduled Jobs for Lookup Field Synchronization and Reconciliation

Scheduled Task	Description
<i>RESOURCE</i> Target Incremental Resource User Reconciliation	This scheduled job is used to perform incremental reconciliation when the target system is configured as a target resource. See Scheduled Jobs for Incremental Reconciliation for more information.
<i>RESOURCE</i> Trusted Incremental Resource User Reconciliation	This scheduled job is used to perform incremental reconciliation when the target system is configured as a trusted resource. See Scheduled Jobs for Incremental Reconciliation for more information.

4.3.3.2 Configuring Scheduled Jobs

To configure a scheduled job:

1. Log in to Oracle Identity System Administration.
2. In the left pane, under System Management, click **Scheduler**.
3. Search for and open the scheduled task as follows:
 - a. On the left pane, 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 following parameters:
 - **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.

 **Note:**

See *Creating Jobs in Oracle Fusion Middleware Administering Oracle Identity Manager* for detailed information about schedule types.

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:**

- 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 is left empty, then reconciliation is not performed.
- Attributes of the scheduled task are discussed in [About Attributes of the Scheduled Jobs](#).

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

 **Note:**

The Stop Execution option is available in the Administrative and User Console. You can use the Scheduler Status page to either start, stop, or reinitialize the scheduler.

4.4 About Configuring Reconciliation for DBAT Connector

Reconciliation involves duplicating in Oracle Identity Manager the creation of and modifications to user accounts on the target system.

This section discusses the following topics related to configuring reconciliation:

- [Connector Objects Used During Target Resource Reconciliation and Provisioning](#)
- [Overview of Connector Objects Used During Trusted Source Reconciliation](#)
- [About Performing Full Reconciliation and Incremental Reconciliation](#)
- [About Performing Limited Reconciliation](#)

4.4.1 Connector Objects Used During Target Resource Reconciliation and Provisioning

As mentioned earlier, target resource reconciliation involves fetching data about newly created or modified users on the target system and using this data to add or modify resources assigned to OIM Users. Provisioning involves creating or modifying account data on the target system through Oracle Identity Manager.

The scheduled job that you use to start a target resource reconciliation run is automatically created when you create the connector.

**See Also:**

Managing Reconciliation in *Oracle Fusion Middleware Administering Oracle Identity Manager* for generic information about connector reconciliation

This section discusses the following topics:

- [User Attributes for Target Resource Reconciliation and Provisioning](#)
- [Supported Target Resource Reconciliation Functions](#)
- [Understanding Reconciliation Rule for Target Resource Reconciliation](#)
- [About Reconciliation Action Rules for Target Resource Reconciliation](#)
- [Understanding Provisioning Functions](#)

4.4.1.1 User Attributes for Target Resource Reconciliation and Provisioning

[Table 4-10](#) provides information about the mandatory user attribute mappings for target resource reconciliation and provisioning. The rest of the user attributes mapping for provisioning and reconciliation is created based on the alias mapping specified in the `DBATConfiguration.groovy` file. In other words, all other attributes that are taken dynamically from the columns in your target system must be mapped with their corresponding fields in Oracle Identity Manager. This mapping is achieved by specifying a value for the alias entry in the `DBATConfiguration.groovy` file. See [Entries in the Predefined Sections](#) for more information about the alias element in the section for configuring the target system as a target resource.

Table 4-10 User Attributes for Target Resource Reconciliation and Provisioning

Process Form Field	Connector Attribute	Description	Mandatory?
User ID	__NAME__	Unique ID of a user account	Yes
Unique Id	__UID__	Unique ID of a user account This is a connector attribute. Note: This is a hidden field. The value in this field is used by the connector to update the user ID.	Yes
Password	__PASSWORD__	Password of the user account	Yes, when the corresponding target system column is mandatory.
Status	__ENABLE__	This field stores the status of the user account.	Yes, when the target system contains a column that stores the status of a user account.

4.4.1.2 Supported Target Resource Reconciliation Functions

The connector supports any of the following actions during a target resource reconciliation run:

- For each account created on the target system, a reconciliation event is generated. Depending on the reconciliation matching rule, a resource is assigned to the corresponding OIM User.
- Updates made to each account on the target system generates update reconciliation events. These updates are propagated to the corresponding resource.
- Deletion of child data from accounts on the target system results in deletion of the same data from the resource. For example, if user John Doe is removed from the Leave Approvers group on the target system, then the same action is performed on the resource assigned to the OIM User John Doe.

4.4.1.3 Understanding Reconciliation Rule for Target Resource Reconciliation



See Also:

Reconciliation Metadata in *Oracle Fusion Middleware Developing and Customizing Applications for Oracle Identity Manager* for generic information about reconciliation matching and action rules

The following sections provide information about the reconciliation rules for this connector:

- [Reconciliation Rule for Target Resource Reconciliation](#)
- [Viewing Reconciliation Rules for Target Resource Reconciliation](#)

4.4.1.3.1 Reconciliation Rule for Target Resource Reconciliation

Reconciliation rules are automatically created when you create the Database Applications Table connector. The following is the process-matching rule:

Rule name: *RESOURCE* User

Rule element: User Login Equals User ID

In the rule name, *RESOURCE* is the name of the IT resource (for example, DB1) that you specify for the `itResourceName` entry in the `DBATConfiguration.groovy` file.

In the rule element:

- User Login is the User ID field on the OIM User form.
- User ID is the `__NAME__` attribute of the connector.

4.4.1.3.2 Viewing Reconciliation Rules for Target Resource Reconciliation

After you create the connector, you can view the reconciliation rule for target resource reconciliation by performing the following steps:



Note:

Perform the following procedure only after the connector is deployed.

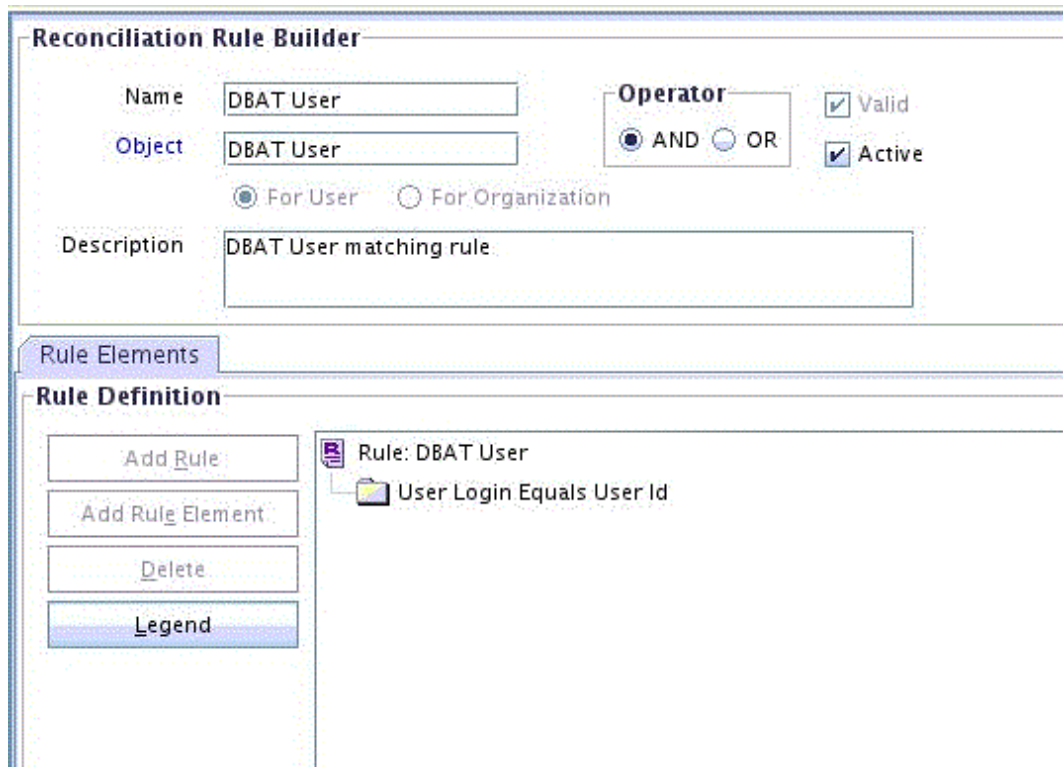
1. Log in to the Oracle Identity Manager Design Console.
2. Expand **Development Tools**.
3. Double-click **Reconciliation Rules**.
4. Search for the rule name for target system reconciliation. The rule name is in the following format:

RESOURCE User

Here, *RESOURCE* is the name of the IT resource (for example, DBAT) that you specify for the `itResourceName` entry in the `DBATConfiguration.groovy` file.

Figure 4-1 shows the reconciliation rule for target resource reconciliation.

Figure 4-1 Reconciliation Rule for Target Resource Reconciliation



4.4.1.4 About Reconciliation Action Rules for Target Resource Reconciliation

The following sections provide information about the reconciliation rules for this connector:

- [Reconciliation Action Rules for Target Resource Reconciliation](#)
- [Viewing Reconciliation Action Rules for Target Resource Reconciliation in the Design Console](#)

4.4.1.4.1 Reconciliation Action Rules for Target Resource Reconciliation

Table 4-11 lists the action rules for target resource reconciliation.

Table 4-11 Action Rules for Target Resource Reconciliation

Rule Condition	Action
No Matches Found	Assign to Authorizer With Least Load
One Entity Match Found	Establish Link
One Process Match Found	Establish Link

 **Note:**

No action is performed for rule conditions that are not predefined for this connector. You can define your own action rule for such rule conditions. See *Setting a Reconciliation Action Rule in Oracle Fusion Middleware Developing and Customizing Applications for Oracle Identity Manager* for information about modifying or creating reconciliation action rules.

4.4.1.4.2 Viewing Reconciliation Action Rules for Target Resource Reconciliation in the Design Console

After you create the connector, you can view the reconciliation action rules for target resource reconciliation by performing the following steps:

1. Log in to the Oracle Identity Manager Design Console.
2. Expand **Resource Management**.
3. Double-click **Resource Objects**.
4. Search for and open the resource object corresponding to your target system. The resource object name is in the following format:
RESOURCE_NAME User
Here, *RESOURCE_NAME* is the name of the resource (for example, DB1) that is specified in the DBATConfiguration.groovy file.
5. Click the **Object Reconciliation** tab, and then click the **Reconciliation Action Rules** tab. The Reconciliation Action Rules tab displays the action rules defined for this connector. [Figure 4-2](#) shows the reconciliation action rule for target resource reconciliation.

Figure 4-2 Reconciliation Action Rules for Target Resource Reconciliation

	Rule Condition	Action	User
1	One Process Match Found	Establish Link	
2	No Matches Found	Assign To Authorizer With Lea...	
3	One Entity Match Found	Establish Link	

4.4.1.5 Understanding Provisioning Functions

Provisioning involves creating or modifying a user's data on the target system through Oracle Identity Manager.

The connector supports the following provisioning functions:

- Create User
- Update User
- Enable User
- Disable User
- Revoke User
- Grant Entitlement
- Revoke Entitlement

Note:

The Enable User or Disable User provisioning operations are supported only if there is a column in the target system that stores user account status and values for the Status Column, Enable Value, and Disable Value columns are set.

4.4.2 Overview of Connector Objects Used During Trusted Source Reconciliation

Trusted source reconciliation involves fetching data about newly created or modified users directly on the target system and using this data to create or update OIM Users.

See Trusted Source Reconciliation in *Oracle Fusion Middleware Administering Oracle Identity Manager* for conceptual information about trusted source reconciliation.

This section discusses the following topics:

- [User Attributes for Trusted Source Reconciliation](#)

- [Viewing Reconciliation Rule for Trusted Source Reconciliation](#)
- [Viewing the Reconciliation Action Rules for Trusted Source Reconciliation](#)

4.4.2.1 User Attributes for Trusted Source Reconciliation

Table 4-12 provides information about the mandatory user attribute mappings for trusted source reconciliation. The rest of the user attributes mapping for reconciliation must be created. In other words, all other attributes that are taken dynamically from the columns in your target system must be mapped with their corresponding fields in Oracle Identity Manager. This mapping is achieved by specifying a value for the alias entry in the DBATConfiguration.groovy file. See [Entries in the Predefined Sections](#) for more information about the alias element in the section for configuring the target system as a trusted source.

Table 4-12 lists user attributes for trusted source reconciliation.

Table 4-12 User Attributes for Trusted Source Reconciliation

OIM User Form Field	Connector or Target System Attribute	Description
User Login	__UID__	User login of a user account. This is a connector attribute. Note: This is a hidden field. The value in this field is used by the connector to update the user ID.
Last Name	__NAME__	Unique ID of a user account
Status	__ENABLE__	Status of the user account This is a connector attribute. This attribute is mandatory if the target system contains a column for storing statuses of user accounts.

4.4.2.2 Viewing Reconciliation Rule for Trusted Source Reconciliation



See Also:

Reconciliation Metadata in *Oracle Fusion Middleware Developing and Customizing Applications for Oracle Identity Manager* for generic information about reconciliation matching and action rules

Reconciliation rules are automatically created when you create the Database Applications Table connector. The following is the process-matching rule:

Rule name: *RESOURCE* Trusted User

Rule element: User Login Equals User ID

In the rule name, *RESOURCE* is the name of the IT resource (for example, DBAT) that you specify for the *itResourceName* entry in the DBATConfiguration.groovy file.

In the rule element:

- User Login is the User ID field on the OIM User form.

- User ID is the `__NAME__` attribute of the connector.

After you deploy the connector, you can view the reconciliation rule for target resource reconciliation by performing the following steps:



Note:

Perform the following procedure only after the connector is deployed.

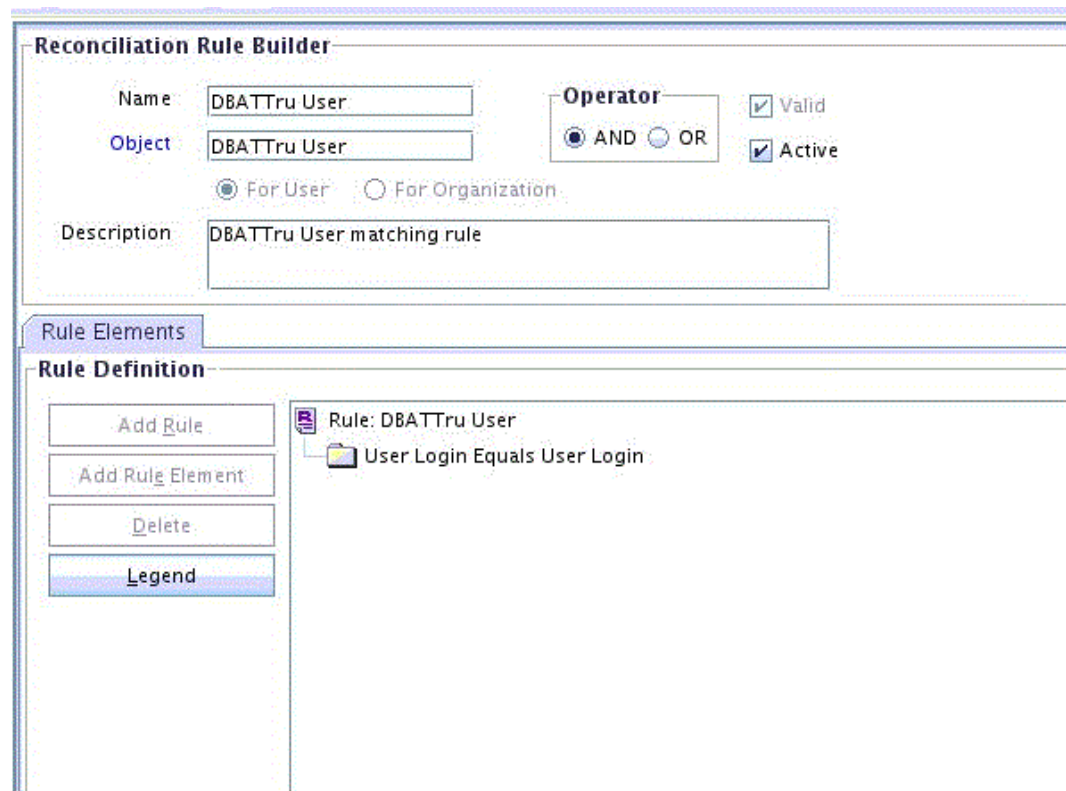
1. Log in to the Oracle Identity Manager Design Console.
2. Expand **Development Tools**.
3. Double-click **Reconciliation Rules**.
4. Search for the rule name for trusted source reconciliation. The rule name is in the following format:

RESOURCE Trusted User

Here, *RESOURCE* is the name of the IT resource (for example, DBATTru) that you specify for the `itResourceName` entry in the `DBATConfiguration.groovy` file.

Figure 4-3 shows the reconciliation rule for trusted source reconciliation.

Figure 4-3 Reconciliation Rule for Trusted Source Reconciliation



4.4.2.3 Viewing the Reconciliation Action Rules for Trusted Source Reconciliation

The following sections provide information about the reconciliation action rules for this connector:

- [Action Rules for Trusted Source Reconciliation](#)
- [View Reconciliation Action Rules for Trusted Source Reconciliation](#)

4.4.2.3.1 Action Rules for Trusted Source Reconciliation

[Table 4-13](#) lists the action rules for trusted source reconciliation.

Table 4-13 Action Rules for Trusted Source Reconciliation

Rule Condition	Action
No Matches Found	Create User
One Entity Match Found	Establish Link

Note:

No action is performed for rule conditions that are not predefined for this connector. You can define your own action rule for such rule conditions. See *Setting a Reconciliation Action Rule in Oracle Fusion Middleware Developing and Customizing Applications for Oracle Identity Manager* for information about modifying or creating reconciliation action rules.

4.4.2.3.2 View Reconciliation Action Rules for Trusted Source Reconciliation

After you deploy the connector, you can view the reconciliation action rules for target resource reconciliation by performing the following steps:

1. Log in to the Oracle Identity Manager Design Console.
2. Expand **Resource Management**.
3. Double-click **Resource Objects**.
4. Search for and open the resource object corresponding your target system. The resource object name is in the following format:
RESOURCE Trusted User
RESOURCE is the name of the IT resource (for example, DBAT) that you specify for the `itResourceName` entry in the `DBATConfiguration.groovy` file.
5. Click the **Object Reconciliation** tab, and then click the **Reconciliation Action Rules** tab. The Reconciliation Action Rules tab displays the action rules defined for this connector. [Figure 4-4](#) shows the reconciliation action rule for trusted source reconciliation.

Figure 4-4 Reconciliation Action Rules for Trusted Source Reconciliation

	Rule Condition	Action	User
1	One Entity Match Found	Establish Link	
2	One Process Match Found	Establish Link	
3	No Matches Found	Create User	

4.4.3 About Performing Full Reconciliation and Incremental Reconciliation

Full reconciliation involves reconciling all existing user records from the target system into Oracle Identity Manager. 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 Manager.

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.

4.4.4 About 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](#)

4.4.4.1 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 Manager. If you do not specify a value for the Filter attribute, then all the records in the target system are reconciled into Oracle Identity Manager.

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 Manager*.

4.4.4.2 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.
- At any given point in time, you can change the WHERE clause by modifying the value of the customizedQuery parameter of the IT resource. There is no need to change the value in the DBATConfiguration.groovy file and regenerate the connector.

4.5 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 Manager:

1. Log in to Oracle Identity Administrative and User console.
2. Create a user. See *Managing Users in Oracle Fusion Middleware Performing Self Service Tasks with Oracle Identity Manager* for more information about creating a user.
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 IT resource (in [Creating an Application Instance](#)), and then click **Checkout**.
5. Specify values for fields in the application form. In addition to specifying values for the parent form, if you want to add child values, then you can specify values for fields on the child form.

 **Note:**

Ensure to select proper values for lookup type fields as there are a few dependent fields. Selecting a wrong value for such fields may result in provisioning failure.

6. Click **Ready to Submit**.
7. Click **Submit**.
8. If you want to provision entitlements, then:
 - a. On the Entitlements tab, click **Request Entitlements**.
 - b. In the Catalog page, search for and add to cart the entitlement, and then click **Checkout**.
 - c. Click **Submit**.

4.6 Configuring Action Scripts

You can configure action scripts to run before or after the create, update, or delete an account provisioning operations.

This section describes action scripts in the following topics:

- [About Action Scripts](#)
- [Lookup Entries for Running Action Scripts](#)
- [Running a CMD Script Before a Create Operation](#)

4.6.1 About Action Scripts

Actions are scripts that you can configure to run before or after the create, update, or delete an account provisioning operations. For example, you could configure a script to run before every user creation.

Every connector should specify the scripting language and target it supports. The Database Application Tables connector supports the following scripts:

- **CMD:** Windows batch script
- **GROOVY:** Groovy script

The target means the location where the script is executed. If the target is **Connector**, then the script is executed on the same computer (JVM or .Net Runtime) where the connector is deployed. For example, if you deploy the connector on the connector server, the script will be executed on that computer.

That is, if you have deployed the connector in OIM, the script runs in your JVM. If you have deployed the connector remotely in the connector server, then the script runs in the remote JVM or .Net Runtime.

Note:

This connector supports only the **Connector** target. This means that the connector supports execution of action scripts on the computer on which the connector is deployed. However, action scripts on the target system can be [Configuring the Connector for Stored Procedures and Groovy Scripts](#) handled by using custom Groovy scripts or procedures. See for more information.

4.6.2 Lookup Entries for Running Action Scripts

[Table 4-14](#) describes the entries to be added to the Lookup.*RESOURCE*.UM.Configuration lookup definition for running actions scripts.

Table 4-14 Lookup Entries for Running Action Scripts

Code Key	Decode
<i>SCHEDULE</i> Action Language	Scripting language of the script you want to run. Enter <code>cmd</code> or <code>GROOVY</code> as the decode value.
<i>SCHEDULE</i> Action File	Full path and name to the file containing the script to be run. Note that the file containing the script must be located on the computer on which Oracle Identity Manager is running.
<i>SCHEDULE</i> Action Target	Context in which the script must be run. Enter <code>Connector</code> as the decode value.

In the preceding table, *SCHEDULE* defines when an action must be performed. An action can be invoked either before or after a create, update, or delete provisioning operation. Therefore, *SCHEDULE* can be replaced with any of the following values:

- Before Create
- Before Update
- Before Delete
- After Create
- After Update
- After Delete

4.6.3 Running a CMD Script Before a Create Operation

All the entries in [Table 4-14](#) define an action together. Therefore, to configure action scripts, all the entries must be defined. Otherwise, no action is performed.

As an example, the following procedure describes the steps to run a cmd script before a create operation:

1. Log in to the Design Console.
2. Search for and open the **Lookup.RESOURCE.UM.Configuration** lookup definition.
3. Add the following new values:
 - **Code Key:** *SCHEDULE* Action Language
Sample value: Before Create Action Language
 - **Decode:** Enter the scripting language of the script you want to execute
For example, `cmd` or `GROOVY`.
4. Add these new values:
 - **Code Key:** *SCHEDULE* Action File
Sample value: Before Create Action File
 - **Decode:** Enter the full path of the batch file that invokes the script. (Oracle Identity Manager must be able to access this file.)
Sample value: `/home/Scripts/InvokeCustomScript.bat`
5. Add these new values:
 - **Code Key:** *SCHEDULE* Action Target
Sample value: Before Create Action Target
 - **Decode:** Connector
As previously stated, the Database Application Tables connector supports the CMD script for a Connector target.
6. Save the lookup definition.

Now, this action will be executed every time you create a user. You must configure these three values for each action you want to execute.

4.7 About Uninstalling the DBAT 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, see *Uninstalling Connectors* in *Oracle Fusion Middleware Administering Oracle Identity Manager*.

5

Extending the Functionality of the Database Application Tables Connector

After you deploy the connector, you can configure it to meet your requirements.

This topic discusses the following optional configuration procedures:

Note:

From Oracle Identity Manager Release 11.1.2 onward, lookup queries are not supported. See *Managing Lookups* in *Oracle Fusion Middleware Administering Oracle Identity Manager* for information about managing lookups by using the Form Designer in the Oracle Identity Manager System Administration console.

- [Adding Custom OIM User Fields for Trusted Source Reconciliation](#)
- [Adding Custom Fields for Target Resource Reconciliation](#)
- [Adding Custom Fields for Provisioning](#)
- [Configuring Transformation of Data During User Reconciliation.](#)
- [Configuring Validation of Data During Reconciliation and Provisioning.](#)

5.1 Adding Custom OIM User Fields for Trusted Source Reconciliation

By default, the mandatory attributes listed in [Table 4-12](#) are mapped for trusted source reconciliation between Oracle Identity Manager and the target system. To add new fields for trusted source reconciliation:

1. Add the new field on the OIM User process form. See *Configuring Custom Attributes* in *Oracle Fusion Middleware Administering Oracle Identity Manager* for information on creating UDFs.

Note:

If the new field that you want to add is already present on the OIM User field, then skip this step and proceed to the next step.

2. Log in to the Design Console.
3. In the resource object definition, add the reconciliation field corresponding to the attribute as follows:

- a. Expand the **Resource Management** folder, and then double-click **Resource Objects**.
 - b. Search for and open the resource object corresponding to your target system.
 - c. On the Object Reconciliation tab, click **Add Field** to open the Add Reconciliation Field dialog box.
 - d. Specify a value for the field name. For example, *Building*.
 - e. From the Field Type list, select a data type for the field. In addition, if you want to designate the attribute as a mandatory attribute, then select the check box.
 - f. Click the Save icon, and then close the dialog box.
 - g. Click the Save icon.
4. Create a reconciliation field mapping in the process definition as follows:
 - a. Expand the **Process Management** folder, and then double-click **Process Definition**.
 - b. Search for and open the process definition for your target system.
 - c. On the Reconciliation Field Mapping tab, click **Add Field Map**.
 - d. From the Field Name list in the Add Reconciliation Field Mapping dialog box, select the name that you have assigned to the attribute created in the resource object.
 - e. Select a value from the **User Attribute** menu and click **OK**.
 - f. If the field mapping is a key field for matching the process data, check the key Field for Reconciliation matching check box.
 - g. Click the Save icon.
 5. Create a reconciliation profile as follows:
 - a. Expand the **Resource Management** folder, and then double-click **Resource Objects**.
 - b. Search for and open the resource object corresponding to your target system.
 - c. On the Object Reconciliation tab, click **Create Reconciliation Profile**. This copies changes made to the resource object into the MDS.
 - d. Click the Save icon.
 6. Add an entry for the attribute in the lookup definition for reconciliation attribute mapping as follows:
 - a. Expand the **Administration** folder, and then double-click **Lookup Definition**.
 - b. Search for and open the **Lookup.RESOURCE.UM.ReconAttrMap** lookup definition.
 - c. To add a row, click **Add**.
 - d. In the **Code Key** column, enter the name that you have set for the attribute in the resource object. For example, *Building*.
 - e. In the **Decode** column, enter the corresponding name of the target system column. For example, *BUILDING*.
 - f. Click the Save icon.

5.2 Adding Custom Fields for Target Resource Reconciliation

While generating the connector by performing the procedures described in [Generating the Database Application Tables Connector](#), you create mappings between the OIM User fields and the corresponding target system fields (columns) by specifying a value for the alias entry. If there are additional target system fields that you want to use during target resource reconciliation, then you can extend the set of fields by creating custom or user-defined fields (UDFs).

To add a custom field for reconciliation:

1. Log in to the Design Console.
2. In the resource object definition, add the reconciliation field corresponding to the attribute as follows:
 - a. Expand the **Resource Management** folder, and then double-click **Resource Objects**.
 - b. Search for and open the resource object corresponding to your target system.
 - c. On the Object Reconciliation tab, click **Add Field** to open the Add Reconciliation Field dialog box.
 - d. Specify a value for the field name. For example, *Building*.
 - e. From the Field Type list, select a data type for the field. In addition, if you want to designate the attribute as a mandatory attribute, then select the check box.
 - f. Click the Save icon, and then close the dialog box.
 - g. Click the Save icon.
3. Add an entry for the attribute in the lookup definition for reconciliation attribute mapping as follows:
 - a. Expand the **Administration** folder, and then double-click **Lookup Definition**.
 - b. Search for and open the **Lookup.RESOURCE.UM.ReconAttrMap** lookup definition.
 - c. To add a row, click **Add**.
 - d. In the **Code Key** column, enter the name that you have set for the attribute in the resource object. For example, *Building*.
 - e. In the **Decode** column, enter the corresponding name of the target system column. For example, *BUILDING*.
 - f. Click the Save icon.
4. Add the attribute as a field on the process form as follows:
 - a. Expand the **Development Tools** folder, and then double-click **Form Designer**.
 - b. Search for and open the process form for your target system.
 - c. Click **Create New Version** to create a version of the process form. Then, enter a version name and click the Save icon.
 - d. Click **Add**.
 - e. In the newly added row, enter values for the Name, Variant Type, Field Label, and Field Type columns. If required, enter values for the rest of the columns.

 **Note:**

- If the attribute on the target system is of the Time, or Timestamp format, then set the value of the Variant Type column to **String**.
- If you want to handle date attributes of the target system as a date editor, then set the value of the Variant Type column to **Date**. Otherwise, set it to **String**.

- f. Click the Save icon.
 - g. Click **Make Version Active** to activate the new version of the process form.
5. Create a reconciliation field mapping in the process definition as follows:
 - a. Expand the **Process Management** folder, and then double-click **Process Definition**.
 - b. Search for and open the process definition for your target system.
 - c. On the Reconciliation Field Mapping tab, click **Add Field Map**.
 - d. From the Field Name list in the Add Reconciliation Field Mapping dialog box, select the name that you have assigned to the attribute created in the resource object.
 - e. Double-click the Process Data Field, a new pop-up will appear. The entries in the pop-up correspond to the process form fields.
 - f. Select the corresponding newly added field from the pop-up.
 - g. If the field mapping is a key field for matching the process data, check the key Field for Reconciliation matching check box.
 - h. Click the Save icon.
 6. Create a reconciliation profile as follows:
 - a. Expand the **Resource Management** folder, and then double-click **Resource Objects**.
 - b. Search for and open the resource object corresponding to your target system.
 - c. On the Object Reconciliation tab, click **Create Reconciliation Profile**. This copies changes made to the resource object into the MDS.
 - d. Click the Save icon.
 7. Perform all changes made to the Form Designer of the Design Console (in Step 4) in a new UI form as follows:
 - a. Log in to Oracle Identity System Administration.
 - b. Create and active a sandbox. See [Creating and Activating a Sandbox](#) for more information.
 - c. Create a new UI form to view the newly added field along with the rest of the fields. See [Creating a New UI Form](#) for more information about creating a UI form.
 - d. Associate the newly created UI form with the application instance of your target system. To do so, open the existing application instance for your

resource, from the Form field, select the form (created in Step 7.7.c), and then save the application instance.

- e. Publish the sandbox. See [Publishing a Sandbox](#) for more information.
8. Add the attribute for provisioning. [Adding Custom Fields for Provisioning](#) for detailed information about the procedure.

5.3 Adding Custom Fields for Provisioning

While generating the connector, you create mappings between the OIM User fields and the corresponding target system fields (columns) by specifying a value for the alias entry. If there are additional target system fields that you want to use during target resource reconciliation, then you can extend the set of fields by creating custom or user-defined fields (UDFs).

To add a new user-defined field for provisioning:

1. Add the attribute as a field on the process form as follows:

 **Note:**

Directly proceed to the next step if you have already added the field to the process form while performing the procedure described in [Adding Custom Fields for Target Resource Reconciliation](#).

- a. Expand **Development Tools**, and then double-click **Form Designer**.
- b. Search for and open the process form for your target system.
- c. Click **Create New Version** to create a version of the form. Then, enter a version name and click the Save icon.
- d. Click **Add**.
- e. In the newly added row, enter values for the Name, Variant Type, Field Label, and Field Type columns. If required, enter values for the rest of the columns.

 **Note:**

- If the attribute on the target system is of the Time, or Timestamp format, then set the value of the Variant Type column to **String**.
- If you want to handle date attributes of the target system as a date editor, then set the value of the Variant Type column to **Date**. Otherwise, set it to **String**.

- f. Click the Save icon.
 - g. Click **Make Version Active** to activate the new version of the process form.
2. Perform all changes made to the Form Designer of the Design Console (in Step 1) in a new UI form as follows:
 - a. Log in to Oracle Identity System Administration.

- b. Create and activate a sandbox. See [Creating and Activating a Sandbox](#) for more information.
 - c. Create a new UI form to view the newly added field along with the rest of the fields. See [Creating a New UI Form](#) for more information about creating a UI form.
 - d. Associate the newly created UI form with the application instance of your target system. To do so, open the existing application instance for your resource, from the Form field, select the form (created in Step 2.2.c), and then save the application instance.
 - e. Publish the sandbox. See [Publishing a Sandbox](#) for more information.
3. Add an entry in the lookup definition for provisioning attribute mappings as follows:
 - a. Expand **Administration**, and then double-click **Lookup Definition**.
 - b. Search for and open the **Lookup.RESOURCE.UM.ProvAttrMap** lookup definition.
 - c. To add a row, click **Add**.
 - d. In the **Code Key** column, enter the field label for the attribute on the process form. See Step 1 for information about this field name.
 - e. In the **Decode** column, enter the corresponding name of the target system column. For example, `BUILDING`.
 - f. Click the Save icon.
 4. To enable updates of the attribute, add an update process task in the process definition as follows:
 - a. Expand **Process Management**, and then double-click **Process Definition**.
 - b. Search for and open the process definition for your target system.
 - c. On the Tasks tab, click **Add**.
 - d. On the General tab of the dialog box that is displayed, enter a name and description for the task, and then select the following fields in the Task Properties section:
 - Conditional
 - Required for Completion
 - Allow Cancellation while Pending
 - Allow Multiple Instances

 **Note:**

The name must be in the `PROCESS_FORM_FIELD_NAME Updated` format.

- e. Click the Save icon.
- f. On the Integration tab, attach the adapter responsible for performing the update account provisioning operations and map the adapter variables as listed in the following table:

Variable Name	Data Type	Map To	Qualifier	Literal Value
processKeyInstance	Long	Process Data	Process Instance	NA
Adapter return value	Object	Response Code	NA	NA
objectType	String	Literal	String	User
attrFieldName	String	Literal	String	Building
itResourceFieldName	String	Literal	String	IT Resource Form Field Name

- g. Click the Save icon.
 - h. On the Response tab, add appropriate responses.
 - i. Click the Save icon.
 - j. Click the Save icon and then close the dialog box.
5. Adding the attribute for reconciliation.

When you add an attribute on the process form, you must also enable reconciliation of values for that attribute from the target system. See [Adding Custom Fields for Target Resource Reconciliation](#) for more information.

5.4 Configuring Transformation of Data During User Reconciliation

You can configure transformation of reconciled single-valued 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 Manager.



Note:

This section describes an optional procedure. Perform this procedure only if you want to configure transformation of data during reconciliation.

To configure transformation of data:

1. Write code that implements the required transformation logic in a Java class.

The following sample transformation class creates a value for the Full Name attribute by using values fetched from the FIRST_NAME and LAST_NAME columns of the target system:

```
package oracle.iam.connectors.common.transform;

import java.util.HashMap;

public class TransformAttribute {

    /*
    Description:Abstract method for transforming the attributes

    param hmUserDetails<String,Object>
    HashMap containing parent data details
```

```

    param hmEntitlementDetails <String,Object>

    HashMap containing child data details

    */
    public Object transform(HashMap hmUserDetails, HashMap
hmEntitlementDetails,String sField) {
    /*
    * You must write code to transform the attributes.
    Parent data attribute values can be fetched by
    using hmUserDetails.get("Field Name").
    *To fetch child data values, loop through the
    * ArrayList/Vector fetched by
    hmEntitlementDetails.get("Child      Table")
    * Return the transformed attribute.
    */
    String sFirstName= (String)hmUserDetails.get("First Name");
    String sLastName= (String)hmUserDetails.get("Last Name");
    String sFullName=sFirstName+"."+sLastName;
    return sFullName;
    }
}

```

2. Create a JAR file to hold the Java class.
3. Run the Oracle Identity Manager Upload JARs utility to post the JAR file to the Oracle Identity Manager database. This utility is copied into the following location when you install Oracle Identity Manager:

 **Note:**

Before you use this utility, verify that the `WL_HOME` environment variable is set to the directory in which Oracle WebLogic Server is installed.

- For Microsoft Windows:
`OIM_HOME/server/bin/UploadJars.bat`
- For UNIX:
`OIM_HOME/server/bin/UploadJars.sh`

When you run the utility, you are prompted to enter the login credentials of the Oracle Identity Manager administrator, URL of the Oracle Identity Manager host computer, context factory value, type of JAR file being uploaded, and the location from which the JAR file is to be uploaded. Specify 1 as the value of the JAR type.

4. Create a lookup definition for transformation and add an entry to it as follows:
 - a. Log in to the Design Console.
 - b. Expand **Administration**, and then double-click **Lookup Definition**.
 - c. In the Code field, enter `Lookup.RESOURCE.UM.ReconTransformation` as the name of the lookup definition.
 - d. Select the **Lookup Type** option.
 - e. On the Lookup Code Information tab, click **Add**.
A new row is added.

- f. In the **Code Key** column, enter the name of the resource object field into which you want to store the transformed value. For example: `FirstName`.
 - g. In the **Decode** column, enter the name of the class that implements the transformation logic. For example, `oracle.iam.connectors.common.transform.TransformAttribute`.
 - h. Save the changes to the lookup definition.
5. Add an entry in the `Lookup.RESOURCE.UM.Configuration` lookup definition to enable transformation as follows:
 - a. Expand **Administration**, and then double-click **Lookup Definition**.
 - b. Search for and open the **Lookup.RESOURCE.UM.Configuration** lookup definition.
 - c. Create an entry that holds the name of the lookup definition used for transformation as follows:

Code Key: `Recon Transformation Lookup`

Decode: `Lookup.RESOURCE.UM.ReconTransformation`
 - d. Save the changes to the lookup definition.

5.5 Configuring Validation of Data During Reconciliation and Provisioning

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` column 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.

For data that fails the validation check, the following message is displayed or recorded in the log file:

```
oracle.iam.connectors.icfcommon.recon.SearchReconTask : handle : Recon event skipped,
validation failed [Validation failed for attribute: [FIELD_NAME]]
```



Note:

This feature cannot be applied to the Locked/Unlocked status attribute of the target system.

To configure validation of data:

1. Write code that implements the required validation logic in a Java class.

The following sample validation class checks if the value in the First Name attribute contains the number sign (#):

```
package com.validate;
import java.util.*;
public class MyValidation {
public boolean validate(HashMap hmUserDetails,
```

```

        HashMap hmEntitlementDetails, String field) {
    /*
    * You must write code to validate attributes. Parent
    * data values can be fetched by using hmUserDetails.get(field)
    * For child data values, loop through the
    * ArrayList/Vector fetched by hmEntitlementDetails.get("Child
Table")
    * Depending on the outcome of the validation operation,
    * the code must return true or false.
    */
    /*
    * In this sample code, the value "false" is returned if the field
    * contains the number sign (#). Otherwise, the value "true" is
    * returned.
    */
    boolean valid=true;
    String sFirstName=(String) hmUserDetails.get(field);
    for(int i=0;i<sFirstName.length();i++){
        if (sFirstName.charAt(i) == '#'){
            valid=false;
            break;
        }
    }
    return valid;
}
}

```

2. Create a JAR file to hold the Java class.
3. Run the Oracle Identity Manager Upload JARs utility to post the JAR file to the Oracle Identity Manager database. This utility is copied into the following location when you install Oracle Identity Manager:

 **Note:**

Before you use this utility, verify that the `WL_HOME` environment variable is set to the directory in which Oracle WebLogic Server is installed.

- For Microsoft Windows:
`OIM_HOME/server/bin/UploadJars.bat`
- For UNIX:
`OIM_HOME/server/bin/UploadJars.sh`

When you run the utility, you are prompted to enter the login credentials of the Oracle Identity Manager administrator, URL of the Oracle Identity Manager host computer, context factory value, type of JAR file being uploaded, and the location from which the JAR file is to be uploaded. Specify 1 as the value of the JAR type.

4. If you created the Java class for validating a process form field for reconciliation, then:
 - a. Log in to the Design Console.
 - b. Expand **Administration**, and then double-click **Lookup Definition**.
 - c. In the Code field, enter `Lookup.RESOURCE.UM.ReconValidation` as the name of the lookup definition.

- d. Select the **Lookup Type** option.
 - e. On the Lookup Code Information tab, click **Add**.
A new row is added.
 - f. In the Code Key column, enter the resource object field name. For example, `FirstName`.
 - g. In the Decode column, enter the class name. For example, `com.validate.MyValidation`.
 - h. Save the changes to the lookup definition.
 - i. Search for and open the **Lookup.RESOURCE.UM.Configuration** lookup definition.
 - j. Create an entry with the following values:
Code Key: `Recon Validation Lookup`
Decode: `Lookup.RESOURCE.UM.ReconValidation`
 - k. Save the changes to the lookup definition.
5. If you created the Java class for validating a process form field for provisioning, then:
- a. Log in to the Design Console.
 - b. Expand **Administration**, and then double-click **Lookup Definition**.
 - c. In the Code field, enter `Lookup.RESOURCE.UM.ProvValidation` as the name of the lookup definition.
 - d. Select the **Lookup Type** option.
 - e. On the Lookup Code Information tab, click **Add**.
A new row is added.
 - f. In the **Code Key** column, enter the process form field name. In the **Decode** column, enter the class name.
 - g. Save the changes to the lookup definition.
 - h. Search for and open the **Lookup.RESOURCE.UM.Configuration** lookup definition.
 - i. Create an entry with the following values:
Code Key: `Provisioning Validation Lookup`
Decode: `Lookup.RESOURCE.UM.ProvValidation`
 - j. Save the changes to the lookup definition.

6

Known Issues and Workarounds

Know more about the known issue and workaround associated with this release of the connector.

Topic:

- [The Custom Schema Feature of IBM DB2 is not Supported](#)

6.1 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.

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:

- [Sample Groovy Script for a Create Provisioning Operation](#)
- [Sample Groovy Script for an Update Provisioning Operation](#)
- [Sample Groovy Script for a Delete Provisioning Operation](#)
- [Sample Groovy Script for an Add Child Data Provisioning Operation](#)
- [Sample Stored Procedure and Groovy Script for a Delete Child Data Provisioning Operation](#)
- [Sample Stored Procedure and Groovy Script for Lookup Field Synchronization](#)
- [Sample Stored Procedure and Groovy Script for Full or Filter Reconciliation](#)
- [Sample Stored Procedure and Groovy Script for Incremental Reconciliation](#)
- [Tables Used for Sample Groovy and Configuration Scripts](#)

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] Attributes::"+ 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.prepareStatement("{call GET_ROLES(?)}");
    }
    else
    {
        System.out.println("[Lookup] Getting Groups.");
        st = conn.prepareStatement("{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.prepareStatement("{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.DESCRPTION,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.DESCRPTION,USERINFO.SALARY,USERINFO.JOININGDATE ,USERINFO.STA
TUS FROM USERINFO USERINFO where '|| columnName ||' = '||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.prepareStatement("{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.prepareStatement("{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("GROUPEID"),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

A sample groovy scripts and configuration scripts uses the lookup table for roles, groups and parent and child table for user accounts

The tables that are used by sample groovy scripts and configuration scripts 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.

It 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

You can perform an incremental trusted source reconciliation run after generating and installing the connector.

The following are the tasks to be performed for an incremental trusted source reconciliation run:

1. The target system (database) must have users.
2. Generate the connector. This involves configuring the `DBATConfiguration.groovy` file and running the DBAT Generator to discover the schema and generate the connector. See [Generating the Database Application Tables Connector](#) for detailed information about connector generation.

 **Note:**

While configuring the `DBATConfiguration.groovy` file, ensure you have specified a value for the `changeLogColumn` property. Otherwise, the scheduled job used to perform an incremental reconciliation run is not created.

3. Install the generated connector. See [Overview of Installing DBAT Connector](#) for detailed information about installing the generated connector.
4. Create the IT resource for your target system. See [About Configuring Secure Communication Between the Target System and Oracle Identity Manager](#) for more information.
5. 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 Manager. See [Scheduled Jobs for Reconciliation of User Records](#) for more information about this scheduled job.
6. Perform some changes to user records in your target system.
7. 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

You can perform an incremental target resource reconciliation run after generating and installing the connector.

The following are the tasks to be performed for an incremental target resource reconciliation run:

1. The target system (database) must have users.
2. Generate the connector. This involves configuring the `DBATConfiguration.groovy` file and running the DBAT Generator to discover the schema and generate the connector. See [Generating the Database Application Tables Connector](#) for detailed information about connector generation.

 **Note:**

While configuring the `DBATConfiguration.groovy` file, ensure you have specified a value for the `changeLogColumn` property. Otherwise, the scheduled job used to perform an incremental reconciliation run is not created.

3. Install the generated connector. See [Overview of Installing DBAT Connector](#) for detailed information about installing the generated connector.
4. Create the IT resource for your target system. See [Configuring the IT Resource for the Target System](#) for more information.
5. Create a form and an application instance for your target system. See the following sections for more information:
 - [Creating a New UI Form](#)
 - [Creating an Application Instance](#)
6. 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 Manager. At the end of this reconciliation run, all the user records in the target system are linked to corresponding OIM Users as resources.

See [Scheduled Jobs for Reconciliation of User Records](#) for more information about the scheduled job.
7. Perform some changes to user records in your target system.
8. 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

You can 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.

Perform the following steps:

1. Generate the connector. This involves configuring the `DBATConfiguration.groovy` file and running the DBAT Generator to discover the schema and generate the connector. See [Generating the Database Application Tables Connector](#) for detailed information about connector generation.

 **Note:**

While configuring the `DBATConfiguration.groovy` file, ensure you have specified a value for the `changeLogColumn` property. Otherwise, the scheduled job used to perform an incremental reconciliation run is not created.

2. Install the generated connector. See [Overview of Installing DBAT Connector](#) for detailed information about installing the generated connector.
3. Create the IT resource for your target system. See [Configuring the IT Resource for the Target System](#) for more information.
4. Ensure you have empty lookup definitions such as `Lookup.RESOURCE.Example` to store values from child tables in your target system.
5. Update the form and lookup definition to include information that specifies the field is a lookup field. See [Using Lookup Definitions](#) for more information.
6. Run the `RESOURCETarget 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

You can provision operations on child data using the Oracle Identity Administrative and User console.

The following are the tasks to performed for perform provisioning operations on child data:

1. Generate the connector. This involves configuring the `DBATConfiguration.groovy` file and running the DBAT Generator to discover the schema and generate the connector. See [Generating the Database Application Tables Connector](#) for detailed information about connector generation.

 **Note:**

While configuring the `DBATConfiguration.groovy` file, ensure you have specified a value for the `changeLogColumn` property. Otherwise, the scheduled job used to perform an incremental reconciliation run is not created.

2. Install the generated connector. See [Overview of Installing DBAT Connector](#) for detailed information about installing the generated connector.
3. Create the IT resource for your target system. See [Configuring the IT Resource for the Target System](#) for more information.
4. Create a form and an application instance for your target system. See the following sections for more information:
 - [Creating a New UI Form](#)
 - [Creating an Application Instance](#)
5. To provision child data, see [Performing Provisioning Operations](#) for more information.

C

Files and Directories in the Database Application Tables Connector

This appendix describes the files and directories corresponding to the DBAT connector.

The appendix includes the following topics:

- [Files and Directories in the DBAT Connector Installation Media](#)
- [Files and Directories in the Generated Connector Package](#)

C.1 Files and Directories in the DBAT Connector Installation Media

These are the files and directories in the connector installation media.

[Table C-1](#) describes the files and directories in the installation media.

Table C-1 Files and Directories on the Installation Media

Files in the Installation Media Directory	Description
Files and directories in the dbat-11.1.1.6.0.zip file	
undle/ org.identityconnectors.databases. etable-1.2.2	This JAR file is the ICF connector bundle.
generator/dbat- generator-11.1.1.6.0.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-2 .
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 Manager database. Note: A resource bundle is a file containing localized versions of the text strings that include GUI element labels and messages.
Files and directories in the dbat-11.1.1.6.0.zip file	
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.
bin/logging.properties	This file contains the default logging configurations of the DBAT generator.

Table C-1 (Cont.) Files and Directories on the Installation Media

Files in the Installation Media Directory	Description
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 Manager 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.databases.etable-1.2.2	This JAR file is the Identity Connector bundle. During connector installation, this file is copied to the Oracle Identity Manager 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. See Overview of Configuring the Groovy File for more information about the entries in the DBATConfiguration.groovy file.

C.2 Files and Directories in the Generated Connector Package

These are the the files and directories in the generated connector package.

[Table C-2](#) describes the files and directories in the generated connector package.

Table C-2 Files and Directories in the Generated Connector Package

File in the Connector Package	Description
bundle/org.identityconnectors.databasetable-1.2.2	This JAR file contains the connector bundle.
onfiguration/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.

Table C-2 (Cont.) Files and Directories in the Generated Connector Package

File in the Connector Package	Description
xml/IT_RES_DEF-ConnectorConfig.xml file	<p>This XML file contains definitions for connector components such as IT resource, lookup definitions, scheduled tasks, process forms, and resource objects.</p> <p>This file is also referred to as the connector configuration file.</p>



See Also:

[Understanding the Generated Connector Package](#) for information about the structure of the generated connector package

Index