Oracle® Identity Manager

Connector Guide for Generic Scripting

11.1.1
E69380-04
August 2020

Oracle Identity Manager Connector Guide for Generic Scripting, 11.1.1

E69380-04

Primary Author: Alankrita Prakash

Contributors: Gowri.G.R

Contents

List of Figures

List of Tables

Preface

This guide describes the Generic Scripting connector, an ICF-based generic connector framework, that enables you to generate a custom connector based on your target system schema and lets you perform connector operations by using your own scripts.

Audience

This guide is intended for resource administrators and system integration teams that want to integrate and manage diverse applications with Oracle Identity Manager. The guide is also intended for users who want to develop, test, and deploy custom connectors. It is imperative that you be familiar with the scripting languages certified for this connector and be able to write your own custom scripts to perform connector operations.

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.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

What's New in the Oracle Identity Manager Connector for Generic Scripting?

This chapter provides an overview of the updates made to the software and documentation for release 11.1.1.5.0 of the Generic Scripting connector.

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

- Software Updates

  This section describes updates made to the connector software. This section also points out the sections of this guide that have been changed in response to each software update.

- Documentation-Specific Updates

  These include major changes made to this guide. For example, the relocation of a section from the second chapter to the third chapter is a documentation-specific update. These changes are not related to software updates.

Software Updates

The following section provides the software updates:

Software Updates in Release 11.1.1.5.0

This is the first release of the Oracle Identity Manager connector for Generic Scripting. Therefore, there are no software updates in this release.

Documentation-Specific Updates

The following section provides the documentation-specific updates:

Documentation-Specific Updates in Release 11.1.1.5.0

The following are documentation-specific updates in Revision "04" of release 11.1.1.5.0:

- The "Oracle Identity Governance or Oracle Identity Manager" row of Table 1-1 has been updated to include support for Oracle Identity Governance 12*c* (12.2.1.4.0).
- Sample Schema File for Database Creation has been added.
- Sample groovy scripts and stored procedures present in Create Script, Update Script, Delete Script, Add Child Data Script, Remove Child Data Script, Lookup Field Synchronization Script, Full and Filtered Reconciliation Script and Incremental Reconciliation Script have been updated.

The following is a documentation-specific update in Revision "03" of release 11.1.1.5.0:

Oracle Identity Governance 19*c* (19.1.0.0.0) has been removed from the "Oracle Identity Governance or Oracle Identity Manager" row of Table 1-1.

The following are the documentation-specific updates in Revision "02" of release 11.1.1.5.0:

- Chapter 5, "Known Issues and Workaround" has been removed from this guide as there are no known issues associated with this release of the connector.
- The following updates have been made in Table 1-1:
  - "Oracle Identity Manager" row has been renamed to "Oracle Identity Governance or Oracle Identity Manager".
  - Oracle Identity Governance 19c (19.1.0.0.0) certification has been added.
  - Oracle Identity Governance 12*c* (12.2.1.3.0) certification has been added.

The following is a documentation-specific update in revision "01" of release 11.1.1.5.0:

This is the first release of the Oracle Identity Manager connector for Generic Scripting on ICF architecture. Therefore, there are no documentation-specific updates in this release.

1

About the Generic Scripting Connector

This chapter introduces the Generic Scripting connector.
Oracle Identity Manager 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.

This chapter discusses the following topics that introduce the Generic Scripting connector:

1.1 Introduction to the Generic Scripting Connector

The Generic Scripting connector is a solution to integrate OIM with target systems that do not have predefined connectors.

This connector enables you to perform connector operations between OIM and your target system by providing your own scripts. The connector generates a custom connector and the required metadata (such as process forms, lookup definitions, scheduled tasks, and so on) based on the target system schema you initially define. Subsequently, the generated connector will invoke your custom scripts to perform the actual connector operations.

You can develop your custom scripts using scripting languages BeanShell, Groovy, or JavaScript. The connector guide includes a few sample scripts that you can modify to suit your requirements and perform connector operations.

The Generic Scripting connector has the ability to connect with multiple target systems using the same connector bundle. The following are some of the advantages of using a Generic Scripting connector:

- Eliminates the need to deploy and test a predefined connector for each target system.
- Reduces time and effort required to develop, deploy, and test custom connectors for multiple target systems in your environment.

- Enables you to easily integrate numerous target systems with OIM that do not have predefined connectors.
- Provides flexibility to define custom rules and business logic that can be dynamically modified at run time for complex applications.
- Provides platform independence for the target system. This connector can be used with target systems that belong to enterprise, mobile, cloud, or social environments.

1.2 Usage Recommendation

Depending on the 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 later than release 9.1.0.2 and earlier than Oracle Identity Manager 11*g* Release 1 (11.1.1.5.3), then you must use the 9.1.*x* version of this connector.
- If you are using Oracle Identity Manager 11*g* Release 1 (11.1.1.5.3) and any later BP in this release track, Oracle Identity Manager 11*g* Release 2 (11.1.2.0.4) and any later BP in this release track, or Oracle Identity Manager 11*g* Release 2 PS3 (11.1.2.3.0) and any later BP in this release track, then you must use the latest 11.1.1.*x* version of this connector.
- If you are using Microsoft SQL Server 2000 as the target system, then you must use the 9.1.*x* version of this connector, irrespective of the Oracle Identity Manager release you are using.

1.3 Certified Components for the Generic Scripting Connector

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

Table 1-1 Certified Components

Item	Requirement
Oracle Identity Manager or Oracle Identity Governance	You can use one of the following releases of Oracle Identity Governance or Oracle Identity Manager: • Oracle Identity Governance 12*c* (12.2.1.4.0) • Oracle Identity Governance 12*c* (12.2.1.3.0) • Oracle Identity Manager 11*g* Release 2 PS3 (11.1.2.3.0) • Oracle Identity Manager 11*g* Release 2 PS2 (11.1.2.2.0)
Target System	Any target system that can be connected through BeanShell, Groovy, or JavaScript. The following are examples of the target system: • Any JDBC compliant database • Any resource over SOAP, HTTP, or REST that supports XML or JSON
Connector Server	11.1.2.1.0
Connector Server JDK	JDK 1.6 or later
Scripting Language	BeanShell, Groovy, JavaScript

1.4 Certified Languages for the Generic Scripting Connector

The connector will support the languages that are supported by Oracle Identity Manager.

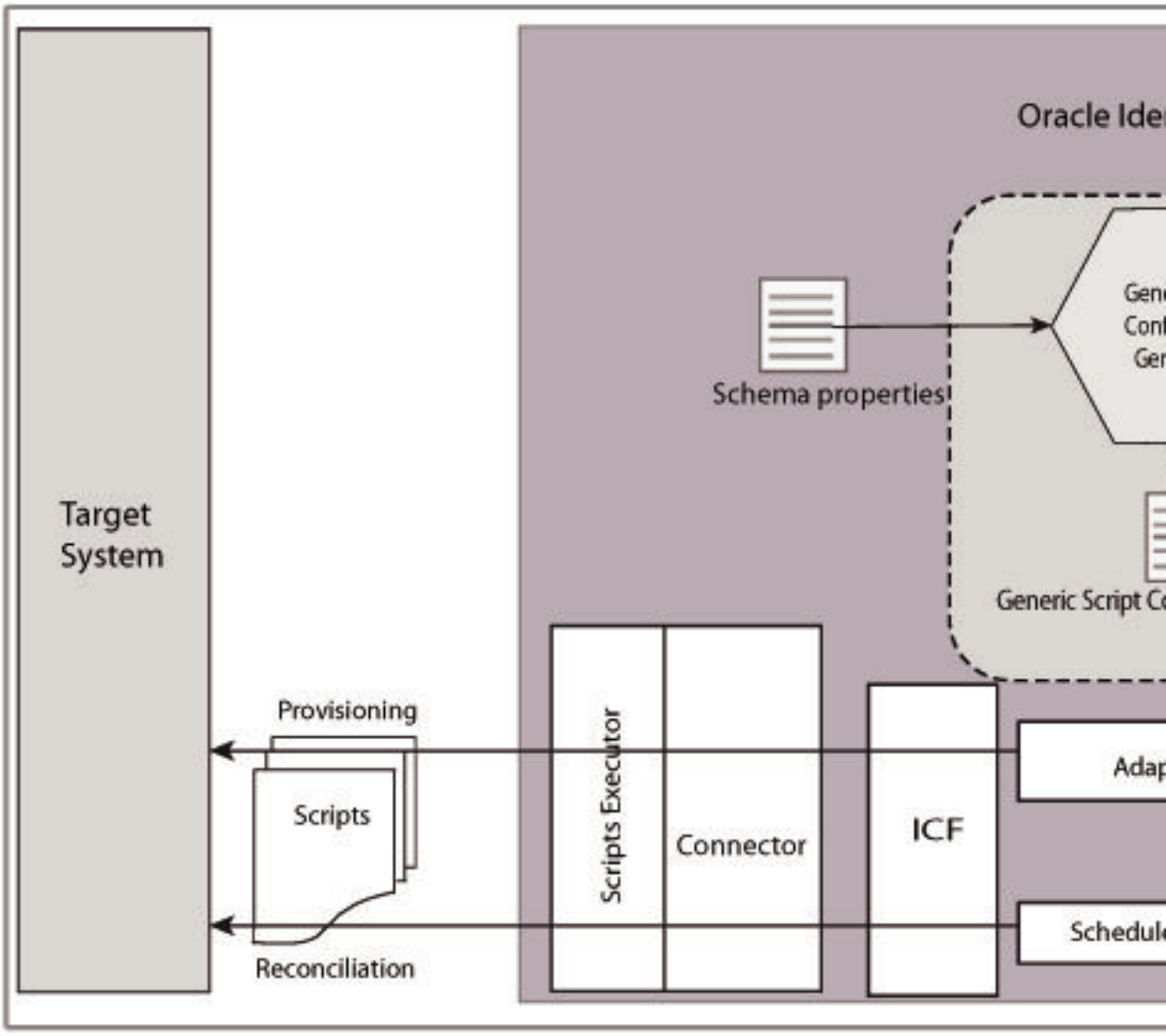
Resource bundles are not part of the connector installation media as the resource bundle entries vary depending on the target system being used.

1.5 Connector Architecture of the Generic Scripting Connector

The Generic Scripting connector is implemented by using the Identity Connector Framework (ICF).

Figure 1-1 shows the architecture of the connector.

Figure 1-1 Connector Architecture



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 Generic Scripting connector is not shipped with any metadata as it is a connector for target system that is not known in advance. Depending on the schema of your target system, 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**

  Understanding the schema of your target system is one of the important aspects in generating the connector. You must create a schema file describing the attributes

of your target system to help the connector know your target system. The Generic Scripting connector includes a groovy file in which you can specify information about your target system. This information is used by the metadata generator, one of the deployment utilities shipped with the connector, to generate the connector based on the target system schema.

In other words, when you run the metadata 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.

1.6 Common Use Cases Supported by the Connector

The Generic Scripting connector can be used with any target system (including custom, home-grown applications) that can be connected with OIM using BeanShell, Groovy, or JavaScript scripting languages.

This section discusses some of the scenarios in which the Generic Scripting connector can be used:

- Integrating SOAP-based target systems

  An organization using a SOAP-based target system wants to integrate with OIM to manage identities. The organization wants to quickly manage its user identities by creating user identities in the target system using OIM and synchronizing user identity changes performed directly in the target system with OIM. In such a scenario, a quick and an easy way is to install the Generic Scripting connector, define the schema file, generate OIM metadata, and then write its own SOAP-based scripts for performing reconciliation and provisioning operations. The connector is ready for use after it is configured with the target system (by providing connection information in the IT resource).

  To create a new user identity in the target system, you must submit the required details in the OIM process form, which triggers a provisioning operation. The connector will execute the corresponding create script against the target system and the user identity will be created on successful execution. Similarly, provisioning operations such as delete and update can be performed.

  To search or retrieve the user identities, you must run a scheduled task from OIM. The connector will run the corresponding search or sync script against the user identities in the target system and fetch all the changes to OIM.

- Integrating heterogenous target systems

  Suppose a web-based product of your organization uses a REST service for customer database updates from clients and a SOAP-based API for backend accounting updates between Mainframe servers. Now suppose you need to

integrate the REST-based and SOAP-based target systems (for customer database updates and accounting updates, respectively), with OIM. One approach would be to deploy and use the predefined Webservices connector for the SOAP-based target system, and develop a custom connector for the REST-based target system. The drawbacks of this approach are as follows:

- Increased time and effort to develop, deploy and test the custom connector for the REST-based target system.
- Deploying and testing the Webservices connector.
- Administering and maintaining two connectors for both systems.

An alternative to this approach is to use a single Generic Scripting connector to interface both the REST and SOAP-based systems to the database instead of deploying a REST connector and also developing and testing a custom connector for SOAP. This enables you to manage both target systems using a single connector.

- Integrating JDBC-based target systems

  Suppose you have multiple databases from different vendors in your organization. For example, you use Oracle Database to store customer and order information and use MS SQL Server to store employee information. Now suppose you must synchronize the information in both the databases with Oracle Identity Manager. One approach would be to deploy and use the Database Application Tables connector for Oracle Database and MS SQL Server. The drawback of this approach is to install one connector each for every database. An alternative to this approach is to use a single Generic Scripting connector to integrate both the databases.

- Integrating cloud-based applications

  A single Generic Scripting connector can be used to integrate one or more cloud-based applications with Oracle Identity Manager. As an alternative to developing, testing, and deploying your custom connector for cloud-based application, you can use the Generic Scripting connector if your cloud-based applications expose their APIs that can be called using any of the certified scripting languages.

1.7 Features of the Connector

The following are the features of the connector:

- Support for Both Trusted Source and Target Resource Reconciliation
- Full and Incremental Reconciliation
- Limited (Filtered) Reconciliation
- Support for Reconciliation of Deleted Records

1.7.1 Support for Both Trusted Source and Target Resource Reconciliation

The Generic Scripting connector includes a groovy file that enables you to configure the connector to run either in the trusted source mode or target resource mode.

See Configuring the ScriptConfiguration.groovy File for more information about configuring the connector for the trusted source and target resource modes.

1.7.2 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.

You can perform a full reconciliation run at any time. See Full Reconciliation and Incremental Reconciliation for more information about performing full and incremental reconciliation.

1.7.3 Limited (Filtered) Reconciliation

You can set a reconciliation filter as the value of the Filter attribute of the scheduled jobs.

This filter specifies the subset of newly added and modified target system records that must be reconciled. This connector does not support complex filters.

See Limited (Filtered) Reconciliation for more information about performing limited reconciliation.

1.7.4 Support for Reconciliation of Deleted 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 target system 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.

1.8 Roadmap for Generating and Using the Connector

The following is the organization of information in the rest of this guide:

- Generating the Generic Scripting Connector describes the procedure that you must perform to configure the groovy file and to run the metadata generator to generate the connector.
- Installing and Configuring the Generic Scripting Connector describes that procedures that you must perform during each stage of connector installation.
- Using the Generic Scripting Connector describes guidelines on using the connector and the procedure to configure reconciliation runs and perform provisioning operations.
- Extending the Functionality of the Generic Scripting Connector describes procedures that you can perform if you want to extend the functionality of the connector.

- Understanding Script Arguments describes the arguments that your custom scripts can use.
- Sample Schema, Scripts, and Connector Generation and Installation Procedure summarizes the connector generation and installation procedure and lists sample schemas and scripts for performing connector operations.
- Files and Directories of the Generic Scripting Connector lists the files and directories that comprise the connector installation media.

2 Generating the Generic Scripting Connector

The procedure to generate the Generic Scripting connector is divided into the following stages:

- Defining the Schema
- Preparing the Resource Properties File
- Configuring the ScriptConfiguration.groovy File
- Generating the Connector

2.1 Defining the Schema

You must define the schema of your target system to let the connector understand the underlying schema of the target system database.

This section discusses the following topics:

- Understanding the Schema File Format
- Creating a Schema File

2.1.1 Understanding the Schema File Format

The schema file is a properties file that is used to represent the structure of your target system. It contains details such as datatypes, mandatory attributes, and the uid attribute that are specific to your target system.

The schema file is used as an input to the metadata generation utility. It is necessary to create a schema.properties file to help the connector understand the target system schema. Before running the metadata generation utility, you must populate the schema file in the specified format.

The schema file is a properties file and consists of name-value pairs. By default, the metadata generation utility generates metadata for an __ACCOUNT__ object class that is used to manage Users, groups, and organizations. If you want to generate metadata for an object class other than __ACCOUNT__, then include the following entry in the schema file:

```
ObjectClass=OBJ_CLASS_NAME
```

Here, *OBJ_CLASS_NAME* is the name of the object class for which you want to generate metadata. The following is a sample value for this entry:

```
ObjectClass=__Test__
```

The following sections discuss the format in which you must specify the value of each property:

- Account Qualifiers

- Field Qualifiers

2.1.1.1 Account Qualifiers

Account qualifiers describe certain attributes of an account in your target system. These qualifiers are common for the target system. You can define the schema of your target system by using the following qualifiers:

- **FieldNames**

  This is a mandatory qualifier. It is a comma-separated list of attributes that the connector must fetch from the target system. All child form names, single-valued and multivalued attributes, including the attribute used for performing incremental reconciliation must be specified here.

  The following is a sample value for the FieldNames qualifier:
  `FieldNames=UserId,UserName,FirstName,LastName,email,Description,Salary , JoiningDate,status,Groups,Roles`

- **UidAttribute**

  This is a mandatory qualifier. It refers to the name of the attribute that corresponds to the unique id of the account.

  For example: `UidAttribute=UserId`

- **NameAttribute**

  This is a mandatory qualifier. This refers to the name of the attribute that corresponds to a descriptive name of the account.

  For example: `NameAttribute=UserName`

- **PasswordAttribute**

  This is an optional qualifier. It refers to the name of the password attribute of the account.

  For example: `PasswordAttribute=accountPwd`

- **StatusAttribute**

  This is an optional qualifier. It refers to the attribute which denotes the status of the account.

  For example: `StatusAttribute=status`

  Oracle Identity Manager requires the status value to be either `True` or `False`. However, if the attribute in the target system contains a value other than true or false, then you must ensure that your script manages the mapping between status values in your target system and Oracle Identity Manager.

2.1.1.2 Field Qualifiers

These qualifiers are specific to each field and are usually specified in one of the following formats:

- The following is the format for parent form fields:

  <FIELDNAME>.<FIELDQUALIFIER>=<VALUE>

  Example: `UserId.Required=true`

- The following is the format for complex child form fields:

<FIELDNAME>.<SUBFIELDNAME>.<FIELDQUALIFIER>=<VALUE>.

Example: `Roles.fromdate.DataType=Long`

The following are the field qualifiers for which values can be specified:

- **Required**

  This field qualifier specifies if the mentioned attribute is mandatory. If the value of this qualifier is set to `true`, the parser will skip processing the records that do not contain this field name.

  For example: `UserId.Required=true`

- **Multivalued**

  This field qualifier specifies if the mentioned attribute is a multivalued field.

  For example: `Roles.Multivalued=true`

- **DataType**

  This field qualifier is used to specify the datatype of the field name. If you do not specify the data type for any field, then it is considered as a String data type by default.

  The following are the possible values for this qualifier:

  – String
  – Long
  – Character
  – Double
  – Float
  – Integer
  – Boolean
  – Byte
  – BigDecimal
  – BigInteger
  – Date

  For example: `startDate.DataType=Date`

- **Subfields**

  This field qualifier specifies the subfields in a multivalued attribute if they are present.

  For example: `Roles.Subfields=roleid,fromdate,todate`

- **EmbeddedObjectClass**

  This field qualifier specifies the object class name of child forms that have more than one subfield. The value of this qualifier is used internally by ICF and is mandatory for all complex child forms.

  For example: `Roles.EmbeddedObjectClass=Roles`

> **See Also:**
>
> Sample Schema File for the Target System for a sample schema file

2.1.2 Creating a Schema File

You must create a schema file describing the structure of your target system as follows:

> **Note:**
>
> You must create the schema.properties file on the computer on which you intend to run the metadata generation utility.

1. Create a .properties file.
2. Add entries in the schema file according to requirements of your environment.

   The following are the mandatory qualifiers that should be defined in the schema file:

   - `FieldNames`
   - `UidAttribute`
   - `NameAttribute`
3. Provide values for each of the entries that you added. See Understanding the Schema File Format for more information about the format in which you must specify these values.
4. Save the .properties file.

2.2 Preparing the Resource Properties File

By default, the connector provides the following parameters in the IT resource to store connection-related information about your target system:

> **Note:**
>
> You must create and place the resource.properties file on the computer that is hosting Oracle Identity Manager.

- host
- port
- user
- password

The connector uses this information to establish a connection from OIM to your target system to perform connector operations. If there are any additional parameters that the connector requires in the scripts being used for connector operations, then you must create a .properties file with these additional parameters. Ensure that the .properties file contains only parameters that are not already available in the default set of IT resource parameters. Including any default IT resource parameters in the .properties file results in creation of duplicate entries and the custom script that you have written for connecting to your target system might fail. For example, host is a parameter that is already available in the IT resource. If you include a host parameter in the .properties file, then your custom connection script fails.

The following is a sample of the resource properties file:

```
applicationName = IDM App
domain = sample.com
proxyHost = www-proxy.example.com
proxyPassword =
proxyPort = 80
scopes = https://www.sample.com/auth/user
```

2.3 Configuring the ScriptConfiguration.groovy File

The Generic Scripting connector is shipped with a groovy file named ScriptConfiguration.groovy.

This section discusses the following topics related to configuring the groovy file:

- About the ScriptConfiguration.groovy File
- Understanding Entries in the Predefined Sections of the Groovy File
- Configuring the Groovy File

2.3.1 About the ScriptConfiguration.groovy File

This ScriptConfiguration.groovy file is located in the genericscript-*RELEASE_NUMBER*/metadata-generator/resources directory of the connector installation ZIP.

You use the ScriptConfiguration.groovy file to specify values for properties that can store basic information about your target system schema. This file is used by the Scripting Generator to perform the following tasks:

- Understand 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 Scripting Generator and directory structure of the generated connector package is discussed later in this chapter.

The ScriptConfiguration.groovy file contains sample configuration (one each for trusted source and target resource) with prepopulated 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 ScriptConfiguration.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.

2.3.2 Understanding Entries in the Predefined Sections of the Groovy File

This section describes the entries in the predefined sections, trusted and target, of the ScriptConfiguration.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 ScriptConfiguration.groovy file, then comment out that entry or attribute by prefixing it with the double-slash symbol (//).

- 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 `GenScriptTrusted` as the value of this entry, then the name of the connector package directory is `GenScriptTrusted.zip`. See Understanding the Generated Connector Package for the directory structure of the connector package.

- 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.

  Default value: `"$itResourceDefName"`

> **Note:**
>
> The value of this entry must be unique for each connector that you create for your target system, if you plan to install or use the connectors in the same OIM environment. 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 metadata generator) such as lookup definitions, resource objects, process forms, and scheduled tasks.
>
> For example, if you specify `GenScriptTrusted` as the value of itResourceName entry, then after you deploy the connector, the configuration lookup definition is created and its name will be Lookup.GenScriptTrusted.Configuration.

- applicationInstanceName

  This is an optional entry and present only in the section for target resource configuration. Enter the name of the application instance for your target system that the connector must generate. If this entry is commented, then the application instance name will be the same as the value of the ITResourceDefName entry.

  Default value: `"$itResourceDefName"`

- connectorDir

  This is an optional entry. This entry is the complete path to the directory that must contain the connector package that is generated when you run the metadata generator. By default, the name of the directory containing the generated connector package is the same as the value of the itResourceDefName entry.

  Sample value: `"/scratch/jdoe/OIMPS3/mw4318/idm7854/server/ConnectorDefaultDirectory/GenScriptTrusted"`

- 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 `GenScriptTrusted` is the value of the itResourceDefName entry, then the name of the XML file that is generated is `GenScriptTrusted-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: `GenScriptTrusted-ConnectorConfig.xml`

- configFileName

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 `GenScriptTrusted` is the value of the itResourceDefName entry, then the name of the XML file that is generated is `GenScriptTrusted-CI.xml`.

- 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 `GenScriptTrusted` is the value of the itResourceDefName entry, then the name of the properties file that is generated is `GenScriptTrusted-generator.properties`.

- version

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

  Sample value: `11.1.1.5.0`

- 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 the connector to run in the trusted source mode.

- bundleJar

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

  Default value: `../lib/org.identityconnectors.genericscript-1.0.11150.jar`

  Do *not* change the value of this entry.

- 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.

  Table 2-1 lists and describes the properties of the Config entry.

Table 2-1 Properties of the config Entry

Property	Mandatory?	Description
schemaFile	Yes	Enter the file URL of the schema file that you want to use. You must enter the file URL in the following format: file:///*URL* Sample value: `file:///home/jdoe/schema.properties` See Defining the Schema for information about the schema file that you created.
resourceProperties	No	Enter the file URL of the properties file containing connection-specific information related to your target system. You must enter the file URL in the following format: file:///*URL* Sample value: `file:///home/jdoe/resource.properties` See Preparing the Resource Properties File for more information about creating this file.
host	Yes	Host name or IP address of the computer hosting the target system.
port	Yes	Port number at which the target system is listening.
user	Yes	User ID or user name of the account in the target system that Oracle Identity Manager must use to connect to and access the target system during reconciliation and provisioning operations. This target system user account must have the necessary permissions to perform all connector operations.
changeLogColumn	No	Optional name of the target system attribute 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). The data type of this target system attribute can be any of the data types supported by the target system. The values in this attribute 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.
createScript	No	This property is present only in the section for target resource configuration. Specify a value for this property if you want the connector to perform Create provisioning operations. Enter the file URL of the script containing the implementation to create objects in your target system. For example, enter the script containing the implementation to perform a create user account provisioning operation. When this script is called, the parent form data is added. You must enter the file URL in the following format: file:///*URL* Sample value: `file:///home/jdoe/scripts/create_user.groovy`

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

Property	Mandatory?	Description
updateScript	No	This property is present only in the section for target resource configuration. Specify a value for this property if you want the connector to perform Update provisioning operations. Enter the file URL of the script containing the implementation to update objects in your target system. For example, enter the script containing the implementation to perform an update user account provisioning operation. This script is called when you update the parent form, or enable or disable the user account. You must enter the file URL in the following format: file:///*URL* Sample value: `file:///home/jdoe/scripts/update_user.groovy`
deleteScript	No	This property is present only in the section for target resource configuration. Specify a value for this property if you want the connector to perform Delete provisioning operations. Enter the file URL of the script containing the implementation to delete objects in your target system. For example, enter the script containing the implementation to perform a delete user account provisioning operation. This script is called when you remove or delete an account. You must enter the file URL in the following format: file:///*URL* Sample value: `file:///home/jdoe/scripts/delete_user.groovy`
executeQueryScript	No	Specify a value for this property if you want to configure the connector to perform reconciliation. Enter the file URL of the script containing the implementation to fetch objects from your target system. This script is called while performing an account search (operations such as full and filtered reconciliation). You must enter the file URL in the following format: file:///*URL* Sample value: `file:///home/jdoe/scripts/recon_user.groovy`
lookupScript	No	This property is present only in the section for target resource configuration. Specify a value for this property if you want the connector to perform lookup field synchronization. Enter the file URL of the script containing the implementation to fetch values of lookup attributes from your target system. You must enter the file URL in the following format: file:///*URL* Sample value: `file:///home/jdoe/scripts/lookup_field_sync.groovy`

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

Property	Mandatory?	Description
syncScript	No	Specify a value for this property if you want the connector to perform incremental reconciliation. Enter the file URL of the script containing the implementation to fetch incremental changes for objects from your target system. You must enter the file URL in the following format: file:///*URL* Sample value: `file:///home/jdoe/scripts/increm_recon_user.groovy`
addMultiValuedAttributeScript	No	This property is present only in the section for target resource configuration. Specify a value for this property if you want the connector to perform provisioning operations on child data. Enter the file URL of the script containing the implementation to add multivalued child data for objects in your target system. This script is called when you add multivalued child attributes. You must enter the file URL in the following format: file:///*URL* Sample value: `file:///home/jdoe/scripts/add_mulval_attr.groovy`
removeMultiValuedAttributeScript	No	This property is present only in the section for target resource configuration. Specify a value for this property if you want the connector to perform provisioning operations on child data. Enter the file URL of the script containing the implementation to remove multivalued child data for objects in your target system. This script is called while removing multivalued child attributes. You must enter the file URL in the following format: file:///*URL* Sample value: `file:///home/jdoe/scripts/remove_mulval_attr.groovy`
connectionScript	No	Enter the file URL of the script containing the implementation to connect to the target system. You must enter the file URL in the following format: file:///*URL* Sample value: `file:///home/jdoe/scripts/connection.groovy`
checkAliveScript	No	Enter the file URL of the script containing the implementation to check whether the connector's physical connection to the target system is alive. This script must do only the minimum that is necessary to check that the connection is still alive You must enter the file URL in the following format: file:///*URL* Sample value: `file:///home/jdoe/scripts/checkalive.groovy`
disposeScript	No	Enter the file URL of the script containing the implementation to dispose any configuration objects. You must enter the file URL in the following format: file:///*URL* Sample value: `file:///home/jdoe/scripts/dispose.groovy`

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

Property	Mandatory?	Description
scriptType	Yes	Enter the language in which the scripts have been written. The possible values are as follows: – GROOVY – BEANSHELL – JAVASCRIPT

> **See Also:**
>
> Understanding Script Arguments for information about the arguments that you can include in the custom scripts that you write to perform connector operations

- lookupAttributeList

  This is an optional entry and is present only in the section for target resource configuration. Enter the list of attributes in your target system that must be handled as lookup fields.

  The connector creates a lookup field for each of the attributes specified in this entry and associates it with the corresponding lookup fields on the OIM User process form.

  If you want to create a lookup field for a single-valued or multivalued field, then enter the value in the following format:

  ['*FIELD_NAME*']

  In this format, replace *FIELD_NAME* with the name of the single or multivalued field.

  If you want create a lookup field for a multivalued field that is embedded, then enter the value in the following format:

  ['*OBJ_CLASS*.*SUB_FIELD_NAME*']

  In this format, replace:

  – *OBJ_CLASS* with the EmbeddedObjectClass name for the child form as specified in the schema file.

  – *SUB_FIELD_NAME* with the subfield name for the child form as specified in the schema file.

  The default value of this entry is:

  ```
  ['ROLES.ROLENAME','FirstName']
  ```

  In this value, `ROLES.ROLENAME` is a multivalued field that is embedded. In other words, `ROLES` is the EmbeddedObjectClass name for roles child form as specified in the schema file (that is, roles.EmbeddedObjectClass=Roles) and `ROLENAME` is one of the subfields for the roles child form as specified in the schema file (that is roles.Subfileds=ROLENAME). `FirstName` is a single-valued field.

  You can modify the default value to meet the requirements in your environment.

  For each of the attributes listed in the lookupAttributeList entry, the connector creates a lookup definition and scheduled job in the following format:

  - Lookup definition format:

    Lookup.${*IT_RES_NAME*}.${*FIELD_NAME*}

    This lookup definition holds the lookup values reconciled from the target system.

  - Scheduled job format:

    IT_RES_NAME Target *FIELD_NAME* Lookup Reconciliation

    This scheduled job is used to load or reconcile lookup values from your target system. See Scheduled Job for Lookup Field Synchronization for more information about the attributes of the scheduled job for lookup reconciliation.

  In both the formats, the connector replaces:

  - *IT_RES_NAME* with the value of the itResourceDefName entry.
  - *FIELD_NAME* with the name of the field for which the lookup field is created.

- entitlementAttributeList

  This is also an optional entry and is present only in the section for target resource configuration. Enter the list of fully qualified attributes in the target system that must be tagged as entitlements.

  The connector creates a lookup field for each of the attributes specified in this entry, assigns the lookup fields to a process form, and adds all the required properties of entitlements.

  If you want to tag entitlements for multivalued fields, then enter the value in the following format:

  ["*MULTIVALUED_FIELD_NAME*"]

  If you want to tag entitlements for a multivalued field that is embedded, then enter the value in the following format:

  ["*OBJ_CLASS*.*SUB_FIELD_NAME*"]

  In this format, replace:

  - *OBJ_CLASS* with the EmbeddedObjectClass name for the child form as specified in the schema file.
  - *SUB_FIELD_NAME* with the subfield name for the child form as specified in the schema file.

  Default value: `["Roles.roleid","__GROUPS__"]`

  You can modify the default value based on your schema.

  In this value, `Roles.RoleId` is an embedded multivalued field and `__GROUPS__` is a multivalued field.

- objectClassAlias

  This is an optional entry. Enter an alias for object class if it is other than ObjectClass.ACCOUNT_NAME or ObjectClass.GROUP_NAME.

  Default value: `['Person']`

- dateAttributeList

  This is an optional entry. Enter the list of attributes that must be handled as date on the process form. Ensure that the data type of the attributes listed here is set to Long in the schema file.

The connector creates a date editor for each of the attributes specified in this entry.

If you want to handle single-valued or multivalued fields as date, then enter the value in the following format:

["*FIELD_NAME*"]

In this format, replace *FIELD_NAME* with the name of the single or multivalued field.

If you want to handle an embedded multivalued field as date, then enter the value in the following format:

["*OBJ_CLASS*.*SUB_FIELD_NAME*"]

In this format, replace:

- *OBJ_CLASS* with the EmbeddedObjectClass name for the child form as specified in the schema file.
- *SUB_FIELD_NAME* with the subfield name for the child form as specified in the schema file.

Default value: `["JoiningDate","Roles.fromdate","Roles.todate"]`

You can modify the default value to meet the requirements in your environment.

The following is a sample value for handling embedded multivalued fields as date:

```
["MyRole.StartDate", "MyRole.EndDate"]
```

- alias

This is a mandatory entry. The metadata generator uses aliases to create relationships between the attributes in the target system and resource object field names in Oracle Identity Manager. In addition, the metadata 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, form name, and form field name levels. Note that the target system attributes 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 attributes to the OIM User form field names. The mappings that you specify here are used to populate entries in the Recon Attribute map 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 attribute 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:

```
['__NAME__':'User Login', 'LastName':'Last
Name','Organization':'Organization Name', 'Employee
Type':'Xellerate Type', '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.

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:

```
['__ENABLE__':'Status', 'FirstName':'First Name', 'email':'Email',
'JoiningDate':'Start Date']
```

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

```
['CONN_ATTR1': 'OIM_FIELD1', 'CONN_ATTR2': 'OIM_FIELD2', . . .
'CONN_ATTRn': 'OIM_FIELDn']
```

In this format:

* *CONN_ATTR* is the connector attribute name.
* *OIM_FIELD* is the name of the field on the OIM User form.

– For target resource configuration

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

* To map connector attributes or target system attributes to fields of the process form. The mappings that you specify here are used to populate entries in the Recon Attribute map and Prov Attribute map lookup definitions for target resource reconciliation.
* To set an alias (a unique and shortened name) for the IT resource name specified in the itResourceName entry.
* To specify a short name for a lengthy process form field name.

  When the number of characters in a process form is more than 11, the metadata generator automatically truncates the process form name to 10 characters and then suffixes it with the digit 0. Subsequently, for every process form that results in the same name after truncating, the suffix is incremented by 1. The metadata generator prevents any two process forms from having the same name by using autonumbering. To gain control over the autogenerated form name and to have meaningful form names, you can use an alias to specify a shortened process form name.

  This is illustrated by the following example:

  Assume that the resource name is GENDB and contains child data that is represented as USER_ROLES in the schema.

  When you run the metadata generator, the process form is created and the form name is UD_GENDB_USER_ROLES. As the number of characters in this process form name is more than 11, the metadata

generator automatically truncates it to UD_GENDB_U0. The truncated form name, UD_GENDB_U0, is not meaningful.

To avoid encountering such issues or forms with autogenerated names, you can use the alias entry to specify short and meaningful process form names.

The following is the default value of the alias entry in the target resource configuration section:

```
['__UID__':'UserId', '__NAME__':'UserId']
```

You cannot delete existing mappings in the default value as they are mandatory. However, you must modify the default value to match the values of the UidAttribute and NameAttribute qualifiers in the schema file. For example, in the schema file, if you have set the values of the UidAttribute and NameAttribute qualifiers to UID and UserId respectively, then you must set the value of the alias entry to the following:

```
['__UID__':'UID', '__NAME__':'UserId']
```

If you want to add mappings for fields other than the ones already present in the alias entry (in other words, optional aliases), 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 in the target resource configuration section:

```
['USERROLERELATIONSHIP':'USRROL', 'comments':'Description', 'Family
Name':'Last Name', 'Visibility':'Status']
```

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

```
['CONN_ATTR1': 'ALIAS_FIELD1', 'CONN_ATTR2': 'ALIAS_FIELD2', . . .
'CONN_ATTRn': 'ALIAS_FIELDn']
```

In this format:

* *CONN_ATTR* is the connector attribute name.
* *ALIAS_FIELD* is the alias corresponding to the connector attribute or target system attribute.

- prepopulate

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

  The default value of this entry is as follows:

  ```
  ['__NAME__':'User Login', 'FIRST_NAME':'First Name', 'LAST_NAME':'Last
  Name', '__PASSWORD__':'Password']
  ```

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

  – User Login
  – First Name
  – Last Name
  – Password

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 TARGET_ATTR1': 'OIM_FIELD1', 'CONN_ATTR2 or TARGET_ATTR2':
'OIM_FIELD2', . . . 'CONN_ATTRn or TARGET_ATTRn': 'OIM_FIELDn']
```

In this format:

- *CONN_ATTR* is the connector attribute name.
- *TARGET_ATTR* is the target system attribute name.
- *OIM_FIELD* is the name of the field on the OIM User form.

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.

2.3.3 Configuring the Groovy File

To configure the ScriptConfiguration.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 on which you intend to run the metadata generation utility. This creates a directory named genericscript-*RELEASE_NUMBER*. See Files and Directories of the Generic Scripting Connector for information about all the files and directories in the connector installation ZIP.
3. In a text editor, open the ScriptConfiguration.groovy file located in the genericscript-*RELEASE_NUMBER*/metadata-generator/resources directory.
4. Specify values for entries in one of the following predefined sections:
   - trusted - for configuring the connector for trusted source mode.
   - target - for configuring the connector for target resource mode.

   See Also:

   Understanding Entries in the Predefined Sections of the Groovy File for information about entries in the predefined sections
5. Save and close the ScriptConfiguration.groovy file.

2.4 Generating the Connector

After configuring the ScriptConfiguration.groovy file, you must run the metadata generator to generate the connector package based on your target system schema.

The metadata generator is the GenericScriptGenerator.cmd or GenericScriptGenerator.sh file that is located in the genericscript-*RELEASE_NUMBER*/metadata-generator/bin directory.

To run the metadata generator, in a command window, change to the genericscript-*RELEASE_NUMBER*/metadata-generator/bin directory (for example,

genericscript-11.1.1.5.0/bin) and run one of the following commands depending on the operating system that you are using:

- **For Microsoft Windows**

  `GenericScriptGenerator.cmd` *`CONFIG_FILE CONFIG_NAME`*

- **For UNIX**

  `GenericScriptGenerator.sh` *`CONFIG_FILE CONFIG_NAME`*

In this command, replace:

- *CONFIG_FILE* with the absolute or relative path name of the ScriptConfiguration.groovy file.
- *CONFIG_NAME* with the name of the configuration within the ScriptConfiguration.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:

```
GenericScriptGenerator.cmd ..\resources\ScriptConfiguration.groovy target
```

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

If you encounter any errors while running the metadata generator, then you must fix it and then resume running the metadata generator.

2.4.1 Understanding the Generated Connector Package

The connector package is a ZIP file that is generated in the GenericScript-*RELEASE_NUMBER*/metadata-generator/ directory.

For example, if you have specified `GenScript` as the value of the itResourceDefName entry in the ScriptConfiguration.groovy file, then the connector package ZIP (GenScript.zip) file is generated in the GenericScript-11.1.1.5.0/metadata-generator/ directory. The directory structure of the connector package is as follows:

```
CONNECTOR_PACKAGE/
        configuration/
                IT_RES_DEF-CI.xml
        resources/
                genericscript-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 ScriptConfiguration.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 ScriptConfiguration.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.

3

Installing and Configuring the Generic Scripting Connector

This chapter describes how to install and configure the Generic Scripting connector. It also describes the prerequisites you must meet before you can successfully install and configure the connector and the steps you must follow after installing the connector. The procedure to install and configure the Generic Scripting connector can be divided into the following stages:

- Preinstallation
- Installing the Connector
- Postinstallation
- Upgrading the Connector

3.1 Preinstallation

Preinstallation involves downloading and copying third-party JAR file on your target system.

Before you install the connector, you must download the third-party JAR files and copy them to a local repository on the computer hosting Oracle Identity Manager. You must place the third-party JAR files in the local repository to let the connector consume it during installation and then communicate with external systems during connector operations.

If you are using BeanShell scripts for performing connector operations, then you must download and copy the BeanShell JAR file. If you are using JavaScript or Groovy scripts, then there is no need to copy any third-party JAR files as they are a part of code files for JDK and OIM, respectively. In addition, you must download and copy any third-party JAR files corresponding to the target system that you are using, if required. For example, if you plan to use BeanShell scripts to perform connector operations and are using Google Apps as your target system, then you must download and copy the BeanShell JAR file and Google Apps third-party JAR files such as google-api-client-1.18.0-rc.jar, google-oauth-client-1.18.0-rc.jar, and so on to the local repository.

You must download and copy the external code files as follows:

1. On the computer hosting Oracle Identity Manager, create a directory named GenericScript-***RELEASE_NUMBER*** under the following directory:

   OIM_HOME/server/ConnectorDefaultDirectory/targetsystems-lib/

   For example, if you are using release 11.1.1.5.0 of this connector, then create a directory named GenericScript-11.1.1.5.0 in the *OIM_HOME*/server/ConnectorDefaultDirectory/targetsystems-lib/ directory.

2. If you plan to use BeanShell scripts for performing connector operations, then download the BeanShell JAR file (Bsh-2.1.8.jar) from the following URL:

   `https://code.google.com/p/beanshell2/wiki/Downloads`

3. Copy the Bsh-2.1.8.jar file to the *OIM_HOME*/server/ConnectorDefaultDirectory/targetsystems-lib/GenericScript-***RELEASE_NUMBER*** directory.

   For example, copy the JAR file to the *OIM_HOME*/server/ConnectorDefaultDirectory/targetsystems-lib/GenericScript-11.1.1.5.0 directory.

4. If your target system requires any third-party JAR files to enable exchange of information from OIM, then download the necessary JAR files and copy them to the following location:

   OIM_HOME/server/ConnectorDefaultDirectory/targetsystems-lib/GenericScript-***RELEASE_NUMBER***

3.2 Installing the Connector

Installation information is divided across the following sections:

- Understanding Installation
- Running the Connector Installer
- Configuring the IT Resource for the Target System

3.2.1 Understanding Installation

The following sections help you understand the installation procedure:

- Summary of Steps to Install the Connector
- About Installing the Connector Locally and Remote

3.2.1.1 Summary of Steps to Install the Connector

Installing this connector requires you to first install the connector bundle that is included in the installation media and then install the connector package (specific to your target system) that you generated while performing the procedure described in Generating the Connector.

The following is a summary of steps to install the Generic Scripting connector

1. Run the connector installer to install the connector bundle included in the installation media. This procedure is described later in this chapter.
2. Run the connector installer to install the connector package (specific to your target system) that you generated while performing the procedure described in Generating the Connector. The procedure to install the connector package is described later in this guide.
3. Configure the IT resource. See Configuring the IT Resource for the Target System for more information.

3.2.1.2 About Installing the Connector Locally and Remote

Depending on where you want to run the connector code (bundle), the connector provides the following installation options:

- Run the connector code locally in Oracle Identity Manager.

  In this scenario, you deploy the connector in Oracle Identity Manager. Deploying the connector in Oracle Identity Manager involves performing the procedures

described in Running the Connector Installer and Configuring the IT Resource for the Target System.

- Run the connector code remotely in a Connector Server.

  In this scenario, you deploy the connector in Oracle Identity Manager, and then, deploy the connector bundle in a Connector Server. See Using an Identity Connector Server in *Oracle Fusion Middleware Developing and Customizing Applications for Oracle Identity Manager* for information about installing, configuring, and running the Connector Server, and then installing the connector in a Connector Server.

3.2.2 Running the Connector Installer

As discussed in one of the earlier sections, you must first install the connector bundle that is included in the installation media and then install the connector bundle that is a part of the connector package that you generated.

The procedure to install both connector bundles is the same except for the following differences:

- Before running the connector installer to install the connector bundle from the installation media, you must copy the contents of the connector installation media to the *OIM_HOME*/server/ConnectorDefaultDirectory directory.
- Before running the connector installer to install the generated connector, you must copy the unzipped connector package (generated in Generating the Connector) to the *OIM_HOME*/server/ConnectorDefaultDirectory directory.

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.

To run the Connector Installer:

1. If you are installing the connector included in the installation media, then copy the contents of the connector installation media to the following directory:

   OIM_HOME/server/ConnectorDefaultDirectory

2. If you are installing the connector from the generated connector package, then copy the unzipped connector package (generated in Generating the Connector) to the following directory:

   OIM_HOME/server/ConnectorDefaultDirectory

3. Log in to Oracle Identity System Administration.
4. In the left pane, under Provisioning Configuration, click **Manage Connector.**
5. In the Manage Connector page, click **Install.**
6. From the Connector List, select one of the following connectors:

- If you are installing the connector included in the connector installation media, then select **GenericScript Connector-*RELEASE_NUMBER.***
- If you are installing the generated connector, then select the name of the connector package (generated by running the metadata 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 relevant connector name depending on whether you are installing the connector included in the connector installation media or the generated connector.

7. Click **Load**.

8. 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.

9. Click **Exit** to close the installation page.

When you run the Connector Installer, it processes the script in the GenericScript-CI.xml file located in the configuration directory. This file is listed in Table C-1.

3.2.3 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 metadata 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 ScriptConfiguration.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 prepopulated with values you have specified for the corresponding properties while performing the procedure described in Configuring the ScriptConfiguration.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
Configuration Lookup	Name of the lookup definition that holds connector configuration entries that are used during connector operations.
Connector Server	Name of the connector server IT resource.
changeLogColumn	Name of the column where the last update-related, non-decreasing, value is stored. Can be a number or a timestamp. 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.
host	Host name or IP address of the computer hosting the target system. Sample value: `myhost`
password	Password of the target system user account that Oracle Identity Manager uses to connect to the target system.
port	Port number at which the target system is listening.
scriptType	Name of the language in which the scripts for performing connector operations have been written. Sample value: `BEANSHELL`
user	User ID of the target system user account that Oracle Identity Manager uses to connect to the target system.

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

3.3 Postinstallation

This section discusses the following postinstallation procedures:

- Configuring Oracle Identity Manager
- Replacing the groovy-all.jar File
- Localizing Field Labels in UI Forms
- Clearing Content Related to Connector Resource Bundles from the Server Cache
- Managing Logging for the Generic Scripting Connector

3.3.1 Configuring Oracle Identity Manager

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:

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

- Creating and Activating a Sandbox
- Creating a New UI Form
- Associating the Form with the Application Instance
- Publishing a Sandbox
- Harvesting Entitlements and Sync Catalog

3.3.1.1 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.

3.3.1.2 Creating a New UI Form

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

3.3.1.3 Associating the Form with the Application Instance

By default, an application instance is automatically created after you install the connector. The name of this application instance is the one that is specified as the value of the applicationInstanceName entry in the ScriptConfiguration.groovy file. If you did not specify a value for the applicationInstanceName entry, then the application instance name will be the same as the value of the ITResourceDefName entry. You must associate this application instance with the form created in Creating a New UI Form.

See Managing Application Instances in *Oracle Fusion Middleware Administering Oracle Identity Manager* for instructions on modifying an application instance to associate it with a form.

After updating the application instance, you must publish it to an organization to make the application instance available for requesting and subsequent provisioning to users. 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.

3.3.1.4 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 Generic Scripting 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.

3.3.1.5 Harvesting Entitlements and Sync Catalog

To harvest entitlements and sync catalog:

1. Run the scheduled jobs for lookup field synchronization discussed 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.

3.3.2 Replacing the groovy-all.jar File

You must replace the groovy-all.jar file located on the computer hosting the connector server with the latest version that is available on the computer hosting Oracle Identity Manager.

Password provisioning operations carried out with this connector do not succeed if you fail to replace the groovy-all.jar file.

To replace the groovy-all.jar file:

1. On the computer hosting Oracle Identity Manager, navigate to the *OIM_HOME*/server/apps/oim.ear/APP-INF/lib directory and copy the groovy-all.jar file.

2. On the computer hosting the connector server, replace the groovy-all.jar located in the *CONNECTOR_SERVER*/lib/framework directory with the groovy-all.jar file copied from Oracle Identity Manager.

3.3.3 Localizing Field Labels in UI Forms

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

1. Create a properties file (for example, GS_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 2PS2 (11.1.2.2.0) and later: *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 Generic Scripting 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_GENSCR_USERNAME__c_description']}">
      <source>USERNAME</source>
      <target/>
      </trans-unit>
      ```

```
<trans-unit
id="sessiondef.oracle.iam.ui.runtime.form.model.ACMEFORM.entity.ACMEFORME
O.UD_GENSCR_USERNAME__c_LABEL">
<source>USERNAME</source>
<target/>
</trans-unit>
```

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

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

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

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

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

   Sample file name: BizEditorBundle_ja.xlf.

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

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

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

3.3.4 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.

3.3.5 Managing Logging for the Generic Scripting Connector

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

The following topics provide detailed information about logging:

- Understanding Log Levels
- Enabling Logging

3.3.5.1 Understanding Log Levels

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

- SEVERE.intValue()+100

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

  This level enables logging of information about errors that might allow Oracle Identity 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
SEVERE.intValue()+100	INCIDENT_ERROR:1
SEVERE	ERROR:1
WARNING	WARNING:1
INFO	NOTIFICATION:1
CONFIG	NOTIFICATION:16
FINE	TRACE:1
FINER	TRACE:16
FINEST	TRACE:32

The configuration file for ODL 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.

3.3.5.2 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='genericscript-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.GENERICSCRIPT" level="[LOG_LEVEL]"
useParentHandlers="false">
     <handler name="genericscript-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='genericscript-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/genericScriptLogs.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.GENERICSCRIPT"
level="NOTIFICATION:1" useParentHandlers="false">
     <handler name="genericscript-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.

3.4 Upgrading the Connector

Upgrading the connector is not applicable as this is the first release for this connector.

4
Using the Generic Scripting Connector

This chapter discusses the following topics:

- Lookup Definitions Used During Connector Operations
- Configuring Reconciliation
- Scheduled Jobs
- Performing Provisioning Operations
- Uninstalling the Connector

4.1 Lookup Definitions Used During Connector Operations

Lookup definitions used during connector operations can be categorized as follows:

- Predefined Lookup Definitions
- Lookup Definitions Synchronized with the Target System

4.1.1 Predefined Lookup Definitions

This section discusses the lookup definitions that are created in Oracle Identity Manager after you generate and 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.*RESOURCE*.Configuration
- 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 guide with the value that you specified for the itResourceName entry in the ScriptConfiguration.groovy file. See Understanding Entries in the Predefined Sections of the Groovy File for more information about entries in the ScriptConfiguration.groovy file.

4.1.1.1 Lookup.*RESOURCE*.Configuration

The Lookup.*RESOURCE*.Configuration 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.*RESOURCE*.Configuration Lookup Definition

Code Key	Decode	Description
Bundle Name	org.identityconnectors.genericscript	This entry holds the name of the connector bundle class. Do not modify this entry.
Bundle Version	1.0.11150	This entry holds the version of the connector bundle class. Do not modify this entry.
Connector Name	org.identityconnectors.genericscript.GenericScriptConnector	This entry holds the name of the connector class. Do not modify this entry.
schema file	file:///*URL*	This entry holds the file URL of the schema file that you want to use.
resource property file	file:///*URL*	This entry holds file URL of the properties file containing connection-specific information related to your target system.
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.
connectionScript[LOADFROMURL]	file:///*URL*	This entry holds the file URL of the script containing the implementation to connect to the target system.
checkAliveScript[LOADFROMURL]	file:///*URL*	This entry holds the file URL of the script containing the implementation to check whether the connector's physical connection to the target system is alive. This script must do only the minimum that is necessary to check that the connection is still alive.
disposeScript[LOADFROMURL]	file:///*URL*	This entry holds the file URL of the script containing the implementation to dispose any configuration objects.
createScript[LOADFROMURL]	file:///*URL*	This entry holds the file URL of the script containing the implementation to create objects in your target system.
updateScript[LOADFROMURL]	file:///*URL*	This entry holds the file URL of the script containing the implementation to update objects in your target system.
deleteScript[LOADFROMURL]	file:///*URL*	This entry holds the file URL of the script containing the implementation to delete objects in your target system.
executeQueryScript[LOADFROMURL]	file:///*URL*	This entry holds the file URL of the script containing the implementation to fetch objects from your target system.
syncScript[LOADFROMURL]	file:///*URL*	This entry holds the file URL of the script containing the implementation to fetch incremental changes for objects from your target system.
lookupScript[LOADFROMURL]	file:///*URL*	This entry holds the file URL of the script containing the implementation to fetch values of lookup attributes from your target system.

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

Code Key	Decode	Description
addMultiValuedAttributeScript[LOADFROMURL]	file:///*URL*	This entry holds the file URL of the script containing the implementation to add multivalued child data for objects in your target system.
removeMultiValuedAttributeScript[LOADFROMURL]	file:///*URL*	This entry holds the file URL of the script containing the implementation to remove multivalued child data for objects in your target system.

4.1.1.2 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 Map	Lookup.*RESOURCE*.UM.ReconAttrMap
Recon Attribute Defaults	Lookup.*RESOURCE*.UM.ReconAttrMap.Defaults

4.1.1.3 Lookup.*RESOURCE*.UM.ReconAttrMap

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

If you have configured the connector for target resource mode:

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 target system attribute 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 connector for trusted source mode:

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 attribute 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 ScriptConfiguration.groovy file. See Understanding Entries in the Predefined Sections of the Groovy File for more information about the alias entry.

4.1.1.4 Lookup.*RESOURCE*.UM.ProvAttrMap

The Lookup.*RESOURCE*.UM.ProvAttrMap lookup definition holds mappings between process form fields and target system attribute 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 attribute 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.

- **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 ScriptConfiguration.groovy file. See Understanding Entries in the Predefined Sections of the Groovy File for more information about the alias entry.

4.1.1.5 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 target system attributes. This lookup definition is created only if you have configured the connector for the trusted source mode.

This lookup definition is used when there is a mandatory field on the OIM User form, but no corresponding attribute 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 attribute 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

4.1.2 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 (displaying a set of roles) to specify the role being assigned to the user.

While configuring the ScriptConfiguration.groovy file, if you specified a value for the lookupAttributeList entry, then the connector creates a lookup definition for every target system attribute specified in this entry and then associates it with the corresponding lookup field on the OIM User process form. The connector creates a lookup definition named in the following format:

Lookup.${*IT_RES_NAME*}.${*FIELD_NAME*}

In this format, the connector replaces:

- *IT_RES_NAME* with the value of the itResourceDefName entry in the ScriptConfiguration.groovy file.
- *FIELD_NAME* with the name of the field for which the lookup field is created.

Lookup field synchronization involves copying additions or changes made to the target system attributes (listed in the lookupAttributeList entry) into corresponding lookup definitions (used as an input source for lookup fields) in Oracle Identity Manager. This is achieved by running scheduled jobs for lookup field synchronization.

The following example illustrates the list of lookup definitions created for a given lookupAttributeList value:

Suppose the value of the itResourceDefName entry is `ACME`. If the value of the lookupAttributeList entry is `['Roles', 'Groups']`, then the connector creates the following lookup definitions:

- Lookup.ACME.Roles
- Lookup.ACME.Groups

After you perform lookup field synchronization, data in the lookup definition 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 target system attribute name specified in the Code Key attribute of the scheduled job for lookup field synchronization.

  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 target system attribute name specified in the Decode attribute of the scheduled job for lookup field synchronization.

  Sample value: `GenScript~SYS_ADMIN`

> **See Also:**
>
> Scheduled Job for Lookup Field Synchronization for information about the attributes of the scheduled job for lookup field synchronization

4.2 Configuring Reconciliation

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:

- Reconciliation Rules
- Full Reconciliation and Incremental Reconciliation
- Limited Reconciliation
- Lookup Field Synchronization

4.2.1 Reconciliation Rules

Reconciliation rules are automatically created when you generate the Generic Scripting connector. The following is the format of the rule element:

```
User Login Equals NameAttribute
```

In this rule element:

- User Login is the User ID field on the OIM User form.
- NameAttribute is the value of the account qualifier in the schema.properties file that you created in Defining the Schema.

For example, if the value of the NameAttribute account qualifier is __NAME__, then the rule element is as follows:

```
User Login Equals__NAME__
```

4.2.2 Full Reconciliation and Incremental Reconciliation

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

In **incremental reconciliation,** only records created or modified after the latest date or timestamp the last reconciliation was run are considered for reconciliation.

After you deploy the connector, you must first perform full reconciliation.

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

- Remove or delete any value currently assigned to the Filter attribute and run the scheduled job for user data reconciliation. See Scheduled Jobs for Reconciliation of User Records for more information about the user reconciliation scheduled job and Filter attribute.
- Remove or delete any value currently assigned to the Sync Token attribute and run the scheduled job for incremental reconciliation. See Scheduled Jobs for Incremental Reconciliation for more information about the scheduled job for incremental reconciliation and Sync Token attribute.

To perform incremental reconciliation, configure and run the scheduled job for incremental reconciliation. The scheduled job for incremental reconciliation is generated only if you specify a value for the changeLogColumn property in the IT resource or ScriptConfiguration.groovy file. The first time you run the scheduled job for incremental reconciliation, a full reconciliation is performed. Subsequently, incremental reconciliation is performed.

At any given point in time, you can switch from incremental reconciliation to full reconciliation. All you need to do is perform a full reconciliation run.

4.2.3 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.

> **Note:**
>
> This connector does not support complex filters.

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 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 attributes of the target system to filter 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*.

4.2.4 Lookup Field Synchronization

As discussed earlier, lookup field synchronization involves obtaining the most current values from specific attributes in the target system to the lookup definitions (used as an input source for lookup fields) in Oracle Identity Manager.

You can perform lookup field synchronization by configuring and running the scheduled jobs for lookup field synchronization.

Scheduled jobs for lookup field synchronization are created only if you have specified a value for the lookupAttributeList entry in the ScriptConfiguration.groovy file. The names of these scheduled jobs are in the following format:

IT_RES_NAME Target *FIELD_NAME* Lookup Reconciliation

For every attribute specified in the lookupAttributeList entry, a corresponding scheduled job for reconciling lookup values from the target system is created. This is illustrated by the following example:

Suppose the value of the itResourceDefName entry is `ACME`. If the value of the lookupAttributeList entry is ['Roles', 'Groups'], then the connector creates the following scheduled jobs:

- ACME Target Roles Lookup Reconciliation
- ACME Target Groups Lookup Reconciliation

> **See Also:**
>
> Scheduled Job for Lookup Field Synchronization for information about the attributes of the scheduled job for lookup field synchronization

4.3 Scheduled Jobs

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

This section discusses the following topics related to scheduled jobs:

- Scheduled Job for Lookup Field Synchronization
- Scheduled Jobs for Reconciliation of User Records
- Scheduled Jobs for Reconciliation of Deleted Users Records
- Scheduled Jobs for Incremental Reconciliation
- Configuring Scheduled Jobs

4.3.1 Scheduled Job for Lookup Field Synchronization

After you generate the connector, scheduled jobs for lookup field synchronization are created only if you have specified a value for the lookupAttributeList entry in the ScriptConfiguration.groovy file.

For every attribute specified in the lookupAttributeList entry, a corresponding scheduled job for reconciling lookup values from the target system is created.

Table 4-5 describes the attributes of the scheduled job for lookup field synchronization. Configuring Scheduled Jobs 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 Scheduled Job for Lookup Field Synchronization

Attribute	Description
Code Key Attribute	Enter the name of the attribute that is used to populate the Code Key column of the lookup definition (specified as the value of the Lookup Name attribute).
Decode Attribute	Enter the name of the attribute that is used to populate the Decode column of the lookup definition (specified as the value of the Lookup Name attribute).
IT Resource Name	Name of the IT resource for the target system installation from which you want to reconcile records. The default value of this attribute is the same as the value of the ITResourceDefName entry in the ScriptConfiguration.groovy file.
Lookup Name	Name of the lookup definition in Oracle Identity Manager that must be populated with values fetched from the target system. The value for this attribute is populated automatically if you have specified a value for the lookupAttributeList entry while configuring the ScriptConfiguration.groovy file. The value of this attribute is in the following format: `Lookup.${IT_RES_NAME}.${FIELD_NAME}` For example, if you have specified Roles as the value of the lookupAttributeList entry, then the value of this attribute is `Lookup.GenScriptTrusted.Roles.`
Object Type	Enter the type of object you want to reconcile. Default value: `OTHER` **Note:** For lookup field synchronization, the object type must be any object other than "User."

4.3.2 Scheduled Jobs for Reconciliation of User Records

After you generate 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.

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 Limited Reconciliation for more information.
Incremental Recon Attribute	Enter the name of the target system attribute that holds the time stamp at which the record was last modified. The value in this attribute is used during incremental reconciliation to determine the newest or latest record reconciled from the target system. Sample value: `ModifiedDate`
ITResource Name	Name of the IT resource for the target system installation from which you want to reconcile user records. Sample value: `GenScriptTrusted`
Latest Token	This attribute holds the value of the attribute that is specified as the value of the Incremental Recon Attribute attribute. The Latest Token attribute is used for internal purposes. By default, this value is empty. **Note:** Do not enter a value for this attribute. The reconciliation engine automatically enters a value in this attribute. Sample value: `1354753427000`
Object Type	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 this attribute.
Resource Object Name	Name of the resource object that is used for reconciliation. Sample value: `GenScriptTrusted User`

Table 4-6 (Cont.) Attributes of the User Reconciliation Scheduled Jobs

Attribute	Description
Scheduled Task Name	Name of the scheduled task that is used for reconciliation. The default value of this attribute in the *RESOURCE* Target Resource User Reconciliation scheduled job is *RESOURCE* `Target Resource User Reconciliation.` The default value of this attribute in the *RESOURCE* Trusted User Reconciliation scheduled job is *RESOURCE*`Trusted Resource User Reconciliation.`

4.3.3 Scheduled Jobs for Reconciliation of Deleted Users Records

After you generate 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 data about deleted users in 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. During a reconciliation run, for each deleted user record on the target system, the target system resource is revoked for the corresponding OIM User.

- *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. During a reconciliation run, for each deleted target system user record, the corresponding OIM User is deleted.

Table 4-7 describes the attributes of both scheduled jobs.

Table 4-7 Attributes of the Delete User Reconciliation Scheduled Jobs

Attribute	Description
IT Resource Name	Name of the IT resource for the target system installation from which you want to reconcile information about deleted user records. Sample value: `GenScript`
Object Type	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 this attribute.
Resource Object Name	Name of the resource object that is used for reconciliation. Sample value: `GenScript User`

4.3.4 Scheduled Jobs for Incremental Reconciliation

After you generate the connector, the scheduled task for performing incremental reconciliation is automatically created in Oracle Identity Manager.

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	Name of the IT resource for the target system installation from which you want to reconcile user records. Sample value: `GenScript`
Object Type	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	Name of the resource object that is used for reconciliation. Sample value: `GenScript User`
Scheduled Task Name	Name of the scheduled task that is used for reconciliation. The default value of this attribute in the *RESOURCE* Target Incremental Resource User Reconciliation scheduled job is *`RESOURCE`* `Target Incremental Resource User Reconciliation.` The default value of this attribute in the *RESOURCE* Trusted Incremental Resource User Reconciliation scheduled job is *`RESOURCE`* `Trusted Incremental Resource User Reconciliation.`
Sync Token	This attribute must be left blank when you run incremental reconciliation for the first time. This ensures that data about all records from the target system are fetched into Oracle Identity Manager. After the first reconciliation run, the connector automatically enters a value for this attribute in an XML serialized format. From the next reconciliation run onward, only data about records that are modified since the last reconciliation run ended are fetched into Oracle Identity Manager." Sample value: `<Long>1452231735775</Long>` **Note:** - *Do not* enter a value for this attribute. The reconciliation engine automatically enters a value in this attribute. - This attribute stores values in an XML serialized format.

4.3.5 Configuring Scheduled Jobs

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.

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 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 Performing Provisioning Operations

To perform provisioning operations in Oracle Identity Manager:

1. Log in to Oracle Identity Administrative and User console.
2. Create a user. See Creating a User 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 Associating the Form with the 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.**

4.5 Uninstalling the Connector

You can uninstall the connector if you wish.

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 Generic Scripting Connector

After you generate and install the connector, you can configure it to meet your requirements. This chapter 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.

5.1 Adding Custom OIM User Fields for Trusted Source Reconciliation

You can add custom OIM User fields for trusted source reconciliation.

While generating the connector by performing the procedures described in Generating the Generic Scripting Connector , you create mappings between OIM User fields and the corresponding target system fields by specifying a value for the alias entry. After generating the connector, if there are additional target system fields that you want to use during trusted source reconciliation, then you can extend the set of fields by creating custom or user-defined fields (UDFs).

To add new fields for trusted source reconciliation:

1. Add the new field on the OIM User process form. See Creating 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

You can add custom fields for target resource reconciliation.

While generating the connector by performing the procedures described in Generating the Generic Scripting Connector , you create mappings between OIM User fields and the corresponding target system fields by specifying a value for the alias entry. After generating the connector, 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). See Creating Custom Attributes in *Oracle Fusion Middleware Administering Oracle Identity Manager* for information about creating custom fields.

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.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

You can add custom fields for provisioning.

While generating the connector, by performing the procedure described in Generating the Generic Scripting 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 provisioning, then you can extend the existing set of fields by creating custom or user-defined fields (UDFs). See Configuring Custom Attributes in *Oracle Fusion Middleware Administering Oracle Identity Manager* for information about creating custom fields.

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 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 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 **Conditional**, **Required for Completion, Allow Cancellation while Pending**, and **Allow Multiple Instances** from the Task Properties section:

> **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, `First Name.`
   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.

A

Understanding Script Arguments

This appendix discusses the arguments that you can include in your custom scripts used for performing connector operations.
The connector executes your custom scripts to perform reconciliation and provisioning operations such as Lookup, Sync, Create, Update, and Delete. For each operation, the connector provides a list of values to your script as an argument. The following is a list of connector operation scripts and their corresponding script arguments that can be directly used in your custom scripts:

- **connectionScript**

  All default IT resource parameters and any additional parameters configured in the resource.properties file are available as script arguments for the connection operation. The following is the list of arguments available for the connection operation:

  - configuration - The list of configuration values. For example, configuration.getHost() is used to read the value of the host configured in the IT resource.
  - timing - When the script must be called. The value of the timing attribute for a connection operation is `connect`.
  - trace - Logger as a script trace bridge to the application.

- **createScript**

  The following is the list of arguments available for a script performing the Create operation:

  - timing - When the script must be called. The value of the timing attribute for a Create operation is `create`.
  - trace - Logger as a script trace bridge to the application.
  - attributes - All attributes.
  - conn - Connection object. For example, JDBC connection.
  - objectclass - Name of the object class.

- **updateScript**

  The following is the list of arguments available for a script performing the Update operation:

  - timing - When the script must be called. The value of the timing attribute for an Update operation is `create`.
  - trace - Logger as a script trace bridge to the application.
  - attributes - List of updated attributes.
  - conn - Connection object. For example, JDBC connection.
  - objectclass - Name of the object class.

- **deleteScript**

The following is the list of arguments available for a script performing the Delete operation:

- timing - When the script must be called. The value of the timing attribute for a Delete operation is `delete`.
- trace - Logger as a script trace bridge to the application.
- attributes - This argument contains the uid attribute.
- conn - Connection object. For example, JDBC connection.
- objectclass - Name of the object class.

- **addMultiValuedAttributeScript**

  The following is the list of arguments available for a script adding multivalued child data for objects in your target system:

  - timing - When the script must be called. The value of the timing attribute for this operation is `addMultiValuedAttribute`.
  - trace - Logger as a script trace bridge to the application.
  - attributes - This argument contains the embedded attribute.
  - conn - Connection object. For example, JDBC connection.
  - objectclass - Name of the object class.

- **removeMultiValuedAttributeScript**

  The following is the list of arguments available for a script removing multivalued child data for objects in your target system:

  - timing - When the script must be called. The value of the timing attribute for this operation is `removeMultiValuedAttribute`.
  - trace - Logger as a script trace bridge to the application.
  - attributes - This argument contains the embedded attribute.
  - conn - Connection object. For example, JDBC connection.
  - objectclass - Name of the object class.

- **lookupScript**

  This script operation is common for all lookup field synchronization jobs. The following is the list of arguments available for this script.

  - timing - When the script must be called. The value of the timing attribute for this operation is in the following format:

    executeQuery:*OBJECT_CLASS*

    In this format, *OBJECT_CLASS* is replaced with the value of the Object Type attribute of the lookup field synchronization scheduled job. For example, if the Object Type attribute of the lookup field synchronization scheduled job contains the value "Role", then the value of the timing argument will be `executeQuery:Role`.
  - trace - Logger as a script trace bridge to the application.
  - conn - Connection object. For example, JDBC connection.
  - handler - resultSetHandler for the connector objects produced by the execute query or null return.

  - ATTRS_TO_GET - Array of attributes to be fetched from the target system.
  - objectclass - Name of the object class.
  - util -Utility to create connector objects.

- **executeQueryScript**

  This script is used during full and limited reconciliation runs. The following is the list of arguments available for this script:

  - timing - When the script must be called. The value of the timing attribute for this operation is `executeQuery`.
  - trace - Logger as a script trace bridge to the application.
  - conn - Connection object. For example, JDBC connection.
  - handler - resultSetHandler for the connector objects produced by the execute query or null return.
  - ATTRS_TO_GET - Array of attributes to be fetched from the target system.
  - objectclass - Name of the object class.
  - util -Utility to create connector objects.
  - filterattribute - Name of the attribute on which the filter criteria is applied. The value of this argument is `null` in case there is no filter.
  - filtervalue - Value of the filter attribute. The value of this argument is `null` in case there is no filter.
  - filteroperator - The filter operation. The value of this argument is `null` in case there is no filter. The following is the list of supported filter operators:

    EQUALS, NOT_EQUALS,

    EQUALS_IGNORE_CASE, NOT_EQUALS_IGNORE_CASE,

    GREATER_THAN, LESS_THAN,

    GREATER_THAN_OR_EQUALS, LESS_THAN_OR_EQUALS,

    STARTS_WITH, NOT_STARTS_WITH,

    STARTS_WITH_IGNORE_CASE, NOT_STARTS_WITH_IGNORE_CASE,

    ENDS_WITH, NOT_ENDS_WITH,

    ENDS_WITH_IGNORE_CASE,NOT_ENDS_WITH_IGNORE_CASE,

    CONTAINS, NOT_CONTAINS,

    CONTAINS_IGNORE_CASE, NOT_CONTAINS_IGNORE_CASE,

    CONTAINS_ALL_VALUES, NOT_CONTAINS_ALL_VALUES

- **syncScript**

  This script is used during incremental reconciliation runs. The following is the list of arguments available for this script:

  - timing - When the script must be called. The value of the timing attribute for this operation is `sync`.
  - trace - Logger as a script trace bridge to the application.
  - conn - Connection object. For example, JDBC connection.

- handler - resultSetHandler for the connector objects produced by the sync query or null return.
- ATTRS_TO_GET - Array of attributes to be fetched from the target system.
- objectclass - Name of the object class.
- util -Utility to create connector objects.
- syncattribute- Name of the attribute configured for incremental reconciliation schedule job.
- synctoken - Value of the sync attribute. This argument contains `null` value during the first execution of the incremental scheduled job.

B

Sample Schema, Scripts, and Connector Generation and Installation Procedure

This appendix provides a complete example of a use case for the Generic Scripting connector, including sample scripts and the connector generation and installation procedures.

This example uses Oracle Database as the target system and the sample scripts for connector operations are written in Groovy.

This appendix discusses the following topics:

B.1 Summary of Steps to Generate and Use the Connector

The following is a summary of steps to generate and use the Generic Scripting connector for a sample environment:

1. Create a properties for your target system schema. See Defining the Schema for detailed information about creating a schema file for your target system.
2. Create a resource properties file if the sample target system requires additional parameters to successfully connect with the target system. See Preparing the Resource Properties File for detailed information about creating a resource properties file.
3. Determine and create scripts for the operations that you want the connector to perform. This involves determining the mode in which you want to run the connector, target resource or trusted source.
4. Configure the ScriptConfiguration.groovy file. See Configuring the ScriptConfiguration.groovy File for detailed information about the entries in the ScriptConfiguration.groovy file.
5. Generate the Generic Scripting Connector based on your target system schema specified in the schema.properties file. See Generating the Connector for detailed information about running the metadata generator to generate the connector.
6. Install the connector included in the connector installation media. See Installing the Connector for detailed information about connector installation.

7. Install the connector that you generated in Step 5. See Installing the Connector for detailed information about connector installation.
8. Configure the IT resource for your target system. See Configuring the IT Resource for the Target System for more information.
9. Create a form for your target system and associate it with an application instance. See the following sections for detailed information:
   - Creating a New UI Form
   - Associating the Form with the Application Instance
10. Replace the groovy-all.jar file with the latest version available on the computer hosting Oracle Identity Manager. See Replacing the groovy-all.jar File for detailed information about replacing the groovy-all.jar file.

B.2 Sample Schema File for Database Creation

The following is a sample schema file for Database creation:

```
CREATE TABLE "GENERIC_PARENT"
  (
    "USERID"   VARCHAR2(20 BYTE) NOT NULL ENABLE,
    "PASSWORD" VARCHAR2(20 BYTE),
    "STATUS"   VARCHAR2(20 BYTE),
    "LAST_UPDATE" TIMESTAMP (6) DEFAULT CURRENT_TIMESTAMP,
    "FIRSTNAME"       VARCHAR2(20 BYTE),
    "LASTNAME"        VARCHAR2(20 BYTE),
    "ORG"             VARCHAR2(20 BYTE),
    "CITY"            VARCHAR2(20 BYTE),
    "EMPLOYEE_NUMBER" NUMBER,
    "STARTDATE" DATE,
    "USERNAME" VARCHAR2(20 BYTE),
    "EMAIL"    VARCHAR2(20 BYTE),
    "ENDDATE" DATE,
    "LONGVALUE" LONG,
    "FLOATVALUE" FLOAT(126),
    "CHARVALUE" CHAR(1 BYTE),
    CONSTRAINT "GENERIC_PARENT_PK" PRIMARY KEY ("USERID")
  )

  CREATE TABLE "GENERIC_GROUP"
  (
    "USERID"    VARCHAR2(20 BYTE),
    "GROUPNAME" VARCHAR2(20 BYTE),
    "STARTDATE" DATE,
    "GROUPID" NUMBER,
    CONSTRAINT "GENERIC_GROUP_GENERIC_PAR_FK1"
FOREIGN KEY ("USERID") REFERENCES "GENERIC_PARENT" ("USERID") ENABLE
  )

  CREATE TABLE "GENERIC_ROLE"
  (
    "USERID"   VARCHAR2(20 BYTE),
    "ROLENAME" VARCHAR2(20 BYTE),
    "ROLEID"   NUMBER,
    "STARTDATE" DATE,
    CONSTRAINT "ROLE_FK" FOREIGN KEY ("USERID") REFERENCES "GENERIC_PARENT"
("USERID") ENABLE
  )
```

```
Trigger for Last_Update

CREATE OR REPLACE TRIGGER UPDATE_LASTUPDATED
BEFORE INSERT OR UPDATE ON GENERIC_PARENT
REFERENCING OLD AS OLDREC NEW AS NEWREC
FOR EACH ROW
BEGIN
  :NEWREC.LAST_UPDATE := sysdate;
END;

Create Procedure

create or replace package types
as type cursorType is ref cursor;
end;
```

B.3 Sample Schema Description

The example target system (an Oracle Database) contains a schema named GENERIC, which in turn contains the GENERIC_PARENT, GENERIC_GROUP, GENERIC_ROLE, and GENERIC_ORGANIZATIONS tables.

The following sections describe the columns of each table in the GENERIC schema:

- GENERIC.GENERIC_PARENT Table Description
- GENERIC.GENERIC_GROUP Table Description
- GENERIC.GENERIC_ROLE Table Description
- GENERIC.ORGANIZATIONS Table Description

B.3.1 GENERIC.GENERIC_PARENT Table Description

Table B-1 lists the columns of the GENERIC.GENERIC_PARENT table.

Table B-1 GENERIC.GENERIC_PARENT Table Description

Column Name	Type	Description
USERID	VARCHAR2(20)	NOT NULL, PRIMARY
PASSWORD	VARCHAR2(20)	
STATUS	VARCHAR2(20)	
LAST_UPDATE	TIMESTAMP (6)	This column is used during incremental reconciliation.
FIRSTNAME	VARCHAR2(20)	
LASTNAME	VARCHAR2(20)	
ORG	VARCHAR2(20)	
CITY	VARCHAR2(20)	
EMPLOYEE_NUMBER	NUMBER	
STARTDATE	DATE	
USERNAME	VARCHAR2(20)	
EMAIL	VARCHAR2(20)	

Table B-1 (Cont.) GENERIC.GENERIC_PARENT Table Description

Column Name	Type	Description
ENDDATE	DATE	
LONGVALUE	LONG	
FLOATVALUE	FLOAT(126)	
CHARVALUE	CHAR(1)	

B.3.2 GENERIC.GENERIC_GROUP Table Description

Table B-2 lists the columns of the GENERIC.GENERIC_GROUP table.

Table B-2 GENERIC.GENERIC_GROUP Table Description

Column Name	Type	Description
USERID	VARCHAR2(20)	FOREIGN KEY
GROUPNAME	VARCHAR2(20)	
STARTDATE	DATE	
GROUPID	NUMBER	

B.3.3 GENERIC.GENERIC_ROLE Table Description

Table B-3 lists the columns of the GENERIC.GENERIC_ROLE table.

Table B-3 GENERIC.GENERIC_ROLE Table Description

Column Name	Type	Description
USERID	VARCHAR2(20)	FOREIGN KEY
ROLENAME	VARCHAR2(20)	
STARTDATE	DATE	
ROLEID	NUMBER	

B.3.4 GENERIC.ORGANIZATIONS Table Description

Table B-4 lists the columns of the GENERIC.ORGANIZATIONS table.

Table B-4 GENERIC.ORGANIZATIONS Table Description

Column Name	Type
ORGNAME	VARCHAR2(20)
ORGID	VARCHAR2(20)

B.4 Sample Schema File for the Target System

Create a schema file representing the structure of your target system. The following is a sample schema file which also includes the LAST_UPDATE attribute that will be used during incremental reconciliation:

```
#Sample Schema file

#List of fields
FieldNames=USERID,PASSWORD,USERNAME,STATUS,EMAIL,FIRSTNAME,LASTNAME,ORGANIZATION,
CITY,EMPLOYEE_NUMBER,STARTDATE,ENDDATE,LONGVALUE,FLOATVALUE,CHARVALUE,GENERIC_GRO
UP,GENERIC_ROLE

#Unique ID Attribute
UidAttribute=USERID

#Account Name attribute
NameAttribute=USERNAME

#Multivalued attributes
GENERIC_GROUP.Multivalued=true
GENERIC_ROLE.Multivalued=true

#Subfields for complex child form
GENERIC_GROUP.Subfields=GROUPNAME,STARTDATE,GROUPID
GENERIC_ROLE.Subfields=ROLENAME,ROLEID,STARTDATE

#Complex child form objectClass
GENERIC_ROLE.EmbeddedObjectClass=MyROLES
GENERIC_GROUP.EmbeddedObjectClass=MyGROUPS

#Datatypes (Default:String)
GENERIC_ROLE.STARTDATE.DataType=Long
GENERIC_GROUP.STARTDATE.DataType=Long
GENERIC_GROUP.GROUPID=Integer
GENERIC_ROLE.ROLEID=Integer

STARTDATE.DataType=Long
ENDDATE.DataType=Long
EMPLOYEE_NUMBER.DataType=Integer
LONGVALUE.DataType=Long
FLOATVALUE.DataType=Double

#Incremental reconciliation attribute with datatype set to Long
LAST_UPDATE.DataType=Long

#Parent and child form mandatory fields
GENERIC_ROLE.ROLENAME.Required=true
GENERIC_GROUP.GROUPNAME.Required=true

#Date format
SystemDateFormat=ddmmyy

#Password Attribute
PasswordAttribute=PASSWORD

#Account Status Attribute and Mapping
StatusAttribute=STATUS
```

```
STATUS.True=Enabled
STATUS.False=Disabled
```

B.5 Sample ScriptConfiguration.groovy File

The following is a sample ScriptConfiguration.groovy file which shows sample entries for both the trusted and target sections:

```
/*
 * Run like:
 * In Windows:
GenericScriptGenerator.cmd ..\resources\ScriptConfiguration.groovy trusted
 * In Linux/Unix: sh GenericScriptGenerator.sh ../resources/
ScriptConfiguration.groovy trusted
 */
trusted {
    /*
     * ITResource name
     */
    itResourceDefName='GenScr' // This will be used as a base name for all
metadata across the connector
    //  itResourceName="$itResourceDefName" //the same as itResourceDefName by
default

    /*
     * Output files
     */
      connectorDir="../$itResourceDefName"                    // output dir of
the connector, is the same as it resource name by default
      xmlFile='GenScr-ConnectorConfig.xml'           // name of the dm xml of
the connector
      configFileName='GenScr-CI.xml'        // name of the config xml
      propertiesFile='GenScr.properties'   // name of the resources/properties
file
      version='11.1.1.5.0'    // connector version

    /*
     * Trusted/Target mode
     * For trusted, we will not create forms, dataobjects and event handlers
     * For target, we will create all above metadata
     */
    trusted=true                                                 // Flag to
denote if the mode is trusted or not

    /*
     * Location of the generic script bundle jar
     */
    bundleJar='../lib/org.identityconnectors.genericscript-1.0.11150.jar'

    /*
     * The Configuration used to run the generic script bundle mentioned above,
and get the schema by calling its SchemaOp, which is required for generating
metadata
     */
    config = [
        'schemaFile' : 'file:///scratch/jdoe/OIMR2PS3/mw3722/idm6004/server/
ConnectorDefaultDirectory/genericscript-11.1.1.5.0/metadata-generator/resources/
trusted/schema_sampletarget.txt', //This property is used during metadata
generator,  provide file:///url of the file
```

```
'resourceProperties' : 'file:///scratch/jdoe/OIMR2PS3/mw3722/idm6004/server/
ConnectorDefaultDirectory/genericscript-11.1.1.5.0/metadata-generator/resources/
trusted/resource.properties', // Location of properties file which defines
target specific resource parameters, provide file:///url of the file
        'host' : 'example.org',
        'port' : '1521',
        'user' : 'generic',
        'changeLogColumn':'LAST_UPDATE',

'executeQueryScript': 'file:///scratch/jdoe/OIMR2PS3/mw3722/idm6004/server/
ConnectorDefaultDirectory/genericscript-11.1.1.5.0/metadata-generator/resources/
trusted/executequeryutil.groovy',  // provide the file:///url of the script
        'syncScript': 'file:///scratch/jdoe/OIMR2PS3/mw3722/idm6004/server/
ConnectorDefaultDirectory/genericscript-11.1.1.5.0/metadata-generator/resources/
trusted/sync.groovy', // provide the file:///url of the script
        'connectionScript': 'file:///scratch/jdoe/OIMR2PS3/mw3722/idm6004/server/
ConnectorDefaultDirectory/genericscript-11.1.1.5.0/metadata-generator/resources/
trusted/connection.groovy', // provide the file:///url of the script
        'checkAliveScript': 'file:///scratch/jdoe/OIMR2PS3/mw3722/idm6004/server/
ConnectorDefaultDirectory/genericscript-11.1.1.5.0/metadata-generator/resources/
trusted/checkAlive.groovy', // provide the file:///url of the script
        'disposeScript': '', // provide the file:///url of the script
        'scriptType': 'GROOVY' // proved script type, valid entries are
GROOVY,BEANSHELL,JAVASCRIPT
    ]

      /**
       *  Provide the attribute list that need to be handled as Date on
process form
       *  Make sure these fields datatype in schema should be long
       *  dateAttributeList is not a mandatory field
       **/

    dateAttributeList = ["STARTDATE","ENDDATE"]

    /**
     * Define alias for object class if it is other than
ObjectClass.ACCOUNT_NAME and ObjectClass.GROUP_NAME
     */

    //objectClassAlias = ['Person']

    /**
     *   Alias are used to map the OIM User Form attributes with the Connector
Attributes.
     *   The Format is of 'Connector Attribute':'OIM User Form Attribute'
     *   Mandatory alias shouldn't be removed. Customer can update these
mandatory attributes but should not be removed
     *   Customer can add other aliases to the OIM User form fields
     **/
    // Mapping is mandatory for attributes User Login, Last Name,Organization,
Xellerate Type and Role. One can modify the required mappings but shouldn't
delete them.
    //UID field is not required in trusted but if customer wanted to add
UID field then one can map it to a valid OIM User Form Label
 alias = ['__NAME__':'User Login', 'LASTNAME':'Last
Name','Organization':'Organization Name', 'Employee Type':'Xellerate Type',
'Role':'Role']
    //Extend the aliases to include more connector attributes for trusted
by mapping
    alias += ['__ENABLE__':'Status', 'FIRSTNAME':'First Name', 'EMAIL':'Email',
```

```
'STARTDATE':'Start Date','ENDDATE':'End Date','EMPLOYEE_NUMBER':'Employee
Number']}

/*
 *
 * Run like:
 * In Windows:
GenericScriptGenerator.cmd ..\resources\ScriptConfiguration.groovy target
 * In Linux/Unix: sh GenericScriptGenerator.sh ../resources/
ScriptConfiguration.groovy target */

target {
    /*
     * ITResource name
     */
    itResourceDefName='GenericScriptTarget' // This will be used as a base
name for all metadata across the connector
    //  itResourceName="$itResourceDefName" //the same as itResourceDefName by
default

    /**
     * Give the name of the Application Instance that need to be generated.
By default Application Instance name is taken as itResourceDefName
     * Application Instance is not a mandatory field
     **/
    //applicationInstanceName="$itResourceDefName"

    /*
     * Output files
     */
    //  connectorDir="../$itResourceDefName"                    // output dir
of the connector, is the same as it resource name by default
    //  xmlFile='GenericScriptTarget-ConnectorConfig.xml'           // name of
the dm xml of the connector
    //  configFileName='GenericScriptTarget-CI.xml'                     // name
of the config xml
    //  propertiesFile='GenericScriptTarget-generator.properties'   // name of
the resources/properties file
    //  version='11.1.1.5.0'        // connector version

    /*
     * Location of the generic script bundle jar
     */
    bundleJar='../lib/org.identityconnectors.genericscript-1.0.11150.jar'

    /*
     * The Configuration used to run the generic script bundle mentioned
above, and get the schema by calling its SchemaOp, which is required for
generating metadata
     */
    config = [
        'schemaFile' : 'file:///scratch/jdoe/OIMR2PS3/mw3722/idm6004/server/
ConnectorDefaultDirectory/genericscript-11.1.1.5.0/metadata-generator/resources/
target/schema_sampletarget.txt', //This property is used during metadata
generator,  provide file:///url of the file

'resourceProperties' : 'file:///scratch/jdoe/OIMR2PS3/mw3722/idm6004/server/
ConnectorDefaultDirectory/genericscript-11.1.1.5.0/metadata-generator/resources/
target/resource.properties', // Location of properties file which defines target
specific resource parameters, provide file:///url of the file
        'host' : 'example.com',
```

```
        'port' : '1521',
        'user' : 'generic',
        'changeLogColumn':'LAST_UPDATE',
        'createScript': 'file:///scratch/jdoe/OIMR2PS3/mw3722/idm6004/server/
ConnectorDefaultDirectory/genericscript-11.1.1.5.0/metadata-generator/resources/
target/create.groovy',   // provide the file:///url of the script
        'updateScript': 'file:///scratch/jode/OIMR2PS3/mw3722/idm6004/server/
ConnectorDefaultDirectory/genericscript-11.1.1.5.0/metadata-generator/resources/
target/update.groovy',  // provide the file:///url of the script
        'deleteScript': 'file:///scratch/jdoe/OIMR2PS3/mw3722/idm6004/server/
ConnectorDefaultDirectory/genericscript-11.1.1.5.0/metadata-generator/resources/
target/update.groovy',  // provide the file:///url of the script

'executeQueryScript': 'file:///scratch/jdoe/OIMR2PS3/mw3722/idm6004/server/
ConnectorDefaultDirectory/genericscript-11.1.1.5.0/metadata-generator/resources/
target/executequeryutil.groovy',  // provide the file:///url of the script
        'lookupScript': 'file:///scratch/jdoe/OIMR2PS3/mw3722/idm6004/server/
ConnectorDefaultDirectory/genericscript-11.1.1.5.0/metadata-generator/resources/
target/lookup.groovy', // provide the file:///url of the script
        'syncScript': 'file:///scratch/jdoe/OIMR2PS3/mw3722/idm6004/server/
ConnectorDefaultDirectory/genericscript-11.1.1.5.0/metadata-generator/resources/
target/syncutil.groovy', // provide the file:///url of the script

'addMultiValuedAttributeScript': 'file:///scratch/jdoe/OIMR2PS3/mw3722/idm6004/
server/ConnectorDefaultDirectory/genericscript-11.1.1.5.0/metadata-generator/
resources/target/addchilddata.groovy', // provide the file:///url of the script

'removeMultiValuedAttributeScript': 'file:///scratch/jdoe/OIMR2PS3/mw3722/
idm6004/server/ConnectorDefaultDirectory/genericscript-11.1.1.5.0/metadata-
generator/resources/target/removechilddata.groovy', // provide the file:///url
of the script
        'connectionScript': 'file:///scratch/jdoe/OIMR2PS3/mw3722/idm6004/server/
ConnectorDefaultDirectory/genericscript-11.1.1.5.0/metadata-generator/resources/
target/connection.groovy', // provide the file:///url of the script
        //'checkAliveScript': 'file:///scratch/jdoe/OIMR2PS3/mw3722/idm6004/
server/ConnectorDefaultDirectory/genericscript-11.1.1.5.0/metadata-generator/
resources/target/checkAlive.groovy', // provide the file:///url of the script
        //'disposeScript': '', // provide the file:///url of the script
        'scriptType': 'GROOVY' // proved script type, valid entries
are GROOVY,BEANSHELL,JAVASCRIPT
    ]

    /**
      *  Lookup List is the list of attributes for which
we need to create lookups and map those fields as lookup in form.
      *  For Main Process Form Fields and a Multivalued field the format is
      *         lookupAttributeList=["FieldName"]
      *  For Embedded Multi Valued field the format is
      *        lookupAttributeList=["ObjectClassName.SubFieldName"]
      *  lookups will be generated with the FieldNames
in format Lookup.${ITResource}.${FieldName}
      *
      *  lookupList is not a mandatory field
      */

    lookupAttributeList=['MyROLES.ROLENAME','MyGROUPS.GROUPNAME','ORGANIZATION']

     /* entitlementAttributeList is the list of fully Qualified
field names to which entitlements need to be tagged.
      * If you require entitlements for a multi valued
attribute which is embedded then the format should be as
```

```
      *        entitlementAttributeList=["${ObjectClass}.SubFieldName"]
      *    If the attribute is just a MultiValued then the format is
      *        entitlementAttributeList=["MultiValuedFieldName"]
      *
      * entitlementAttributeList is not a mandatory attribute
      */

       entitlementAttributeList=["MyROLES.ROLENAME","MyGROUPS.GROUPNAME"]

     /**
     * Define alias for object class if it is other
than ObjectClass.ACCOUNT_NAME and ObjectClass.GROUP_NAME
     */

    //objectClassAlias = ['Person']

    /**
      *  Provide the attribute list that need to be handled
as Date on process form
      *  Make sure these fields datatype in schema should be long
      *  dateAttributeList is not a mandatory field
      **/
    dateAttributeList =
["STARTDATE","ENDDATE","MyROLES.STARTDATE","MyGROUPS.STARTDATE"]
    /*
     * Target attribute to user fields alias
     */
    //Mandatory alias
    alias = ['__UID__':'USERID', '__NAME__':'USERNAME']
    //Optional aliases if any (Can also be used to give
short names for lengthy attribute names)
    alias += ['GENERIC_ROLE':'GR','GENERIC_GROUP':'GG','ENDDATE':'ed',
'EMPLOYEE_NUMBER':'Emp']

    /*
     * Generate prepopulate adapters
     */
    prepopulate = ['__NAME__':'User Login', 'FIRSTNAME':'First Name',
'LASTNAME':'Last Name', '__PASSWORD__':'Password','EMAIL':'Email']

}
```

B.6 Sample Resource Properties File

The following is a sample resource.properties file that contains additional parameters that the connector uses to establish a connection between Oracle Identity Manager and the target system:

```
portNumber=1521
databaseName=orcl
```

B.7 Sample Scripts for Connector Operations

This section lists the scripts used to perform connector operations on the sample target system using the Generic Scripting connector.

All scripts in this section have been written in Groovy scripting language.

- Check Alive Script

- Connection Script
- Dispose Script
- Create Script
- Update Script
- Delete Script
- Add Child Data Script
- Remove Child Data Script
- Lookup Field Synchronization Script
- Full and Filtered Reconciliation Script
- Incremental Reconciliation Script

B.7.1 Check Alive Script

The check alive script periodically verifies whether the physical connection between the connector and target system is alive. The following is a sample check alive script:

```
package groovy;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import oracle.jdbc.driver.OracleDriver;
import
org.identityconnectors.framework.common.exceptions.ConnectionFailedException;
import org.identityconnectors.framework.common.exceptions.ConnectorException;
import groovy.sql.Sql;

ResultSet result = null;
PreparedStatement stmt = null;
trace.info("Checking database connection is valid or not");
try {
    if (conn == null
    || conn.isClosed()) {
        trace.info("Connection is closed")
        throw new ConnectionFailedException("Connection Failed ");
    }
    trace.info("test connection using SELECT 1 FROM DUAL");
    //execute sample query to check database connection is valid or not
    stmt = conn.prepareStatement("SELECT 1 FROM DUAL");
    result = stmt.executeQuery();
    trace.info("connection is valid");
} catch (SQLException ex) {
    //Throw an error if any error while executing the query
    throw new ConnectionFailedException("Connection Failed ");
}
```

B.7.2 Connection Script

The connection script creates the connection between the connector and target system. The following is a sample connection script:

```
package groovy;
import java.sql.Connection;
import java.sql.DriverManager;
import oracle.jdbc.driver.OracleDriver;
import org.identityconnectors.framework.common.exceptions.ConnectorException;
import groovy.sql.Sql;
import oracle.jdbc.pool.OracleDataSource;
import org.identityconnectors.common.security.GuardedString;

Connection ret = null;
try {
    OracleDataSource ds = new OracleDataSource()
    ds.user = configuration.getUser() // Read user from IT resource
    //Password field is encrypted, use GuardedString.Accessor for reading
password
    GuardedString guard= configuration.getPassword()
    guard.access(new GuardedString.Accessor()
            {
                public void access(char[] clearChars)
                {
                    ds.password=new String(clearChars);
                }
            }
            );

    ds.driverType = 'thin'
    ds.serverName = configuration.getHost()
    ds.portNumber = Integer.parseInt(configuration.getPortNumber()) //get Port
number defined in resources.properties file ( portNumber = 19999)
    ds.databaseName = configuration.getDatabaseName() //get database name
defined in resources.properties file ( portNumber = 19999)
    // load the driver class..
    test =  Sql.newInstance(ds)
    ret =test.createConnection()
    ret.setAutoCommit(true);
}//end try
catch (Exception e)
{
    trace.error(e, "Exception while connecting to database");
    throw ConnectorException.wrap(e);
}
//Must return connection object, so that same connection will be available to
all subsequent scripts
return ret;
```

B.7.3 Dispose Script

The following is a sample script to dispose any configuration object:

```
package groovy;
import
org.identityconnectors.framework.common.exceptions.ConnectionFailedException;

trace.info("[Dispose-Groovy] Closing the connection");

try {
    //Close the connection. conn is the script argument
    conn.close();
}
catch(Exception e) {
```

```
    throw new ConnectionFailedException("Could not close connection, Connection
already closed");
}
```

B.7.4 Create Script

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

```
"package groovy
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.text.*;
import java.util.Date.*;
import org.identityconnectors.common.security.GuardedString;
import org.identityconnectors.framework.common.objects.*;

import java.text.*;

// START HERE
System.out.println("[Create-Groovy] Attributes::"+attributes);
//
USERID,PASSWORD,USERNAME,STATUS,EMAIL,FIRSTNAME,LASTNAME,ORGANIZATION,CITY,EMPLOY
EE_NUMBER,STARTDATE,ENDDATE,LONGVALUE,FLOATVALUE,CHARVALUE
//Get all the attributes from script argument

// This shows how to read arrtibures


String uid = attributes.get("__NAME__")!=null?
attributes.get("__NAME__").getValue().get(0):null;
GuardedString pass = attributes.get("__PASSWORD__")!=null?
attributes.get("__PASSWORD__").getValue().get(0):null;
String uname = attributes.get("__NAME__")!=null?
attributes.get("__NAME__").getValue().get(0):null;
enableValue = attributes.get("__ENABLE__")!=null?
attributes.get("__ENABLE__").getValue().get(0):true;
String email=attributes.get("EMAIL")!=null?
attributes.get("EMAIL").getValue().get(0):null;
String first=attributes.get("FIRSTNAME")!=null?
attributes.get("FIRSTNAME").getValue().get(0):null;
String last=attributes.get("LASTNAME")!=null?
attributes.get("LASTNAME").getValue().get(0):null;
String org=attributes.get("ORGANIZATION")!=null?
attributes.get("ORGANIZATION").getValue().get(0):null;
String city=attributes.get("CITY")!=null?
attributes.get("CITY").getValue().get(0):null;
emp=attributes.get("EMPLOYEE_NUMBER")!=null?
attributes.get("EMPLOYEE_NUMBER").getValue().get(0):null;
startdate = attributes.get("STARTDATE")!=null?
attributes.get("STARTDATE").getValue().get(0):null;
enddate = attributes.get("ENDDATE")!=null?
attributes.get("ENDDATE").getValue().get(0):null;
String longval=attributes.get("LONGVALUE")!=null?
attributes.get("LONGVALUE").getValue().get(0):null;
floatval=attributes.get("FLOATVALUE")!=null?
attributes.get("FLOATVALUE").getValue().get(0):null;
charval=attributes.get("CHARVALUE")!=null?
attributes.get("CHARVALUE").getValue().get(0):null;

PreparedStatement createStmt = null;
```

```
String ret =null;
try {

    //Call Target API to create a user

    //createStmt =   conn.prepareStatement(....); or createStmt =
conn.createStatement(....);
    //createStmt =   conn.prepareStatement("INSERT INTO
generic_view(USERID,PASSWORD,USERNAME,STATUS,EMAIL,FIRSTNAME,LASTNAME,ORGANIZATIO
N,CITY,EMPLOYEE_NUMBER,STARTDATE,ENDDATE,LONGVALUE,FLOATVALUE,CHARVALUE)
VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)");
        //Set the input parameters
    createStmt =   conn.prepareStatement("INSERT INTO
GENERIC_PARENT(USERID,PASSWORD,USERNAME,STATUS,EMAIL,FIRSTNAME,LASTNAME,ORG,CITY,
EMPLOYEE_NUMBER,STARTDATE,ENDDATE,LONGVALUE,FLOATVALUE,CHARVALUE)
VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)");
    createStmt.setString(1, uid);
    if(pass!=null)
    {
    pass.access(new GuardedString.Accessor(){
       public void access(char[] clearChars) {  createStmt.setString(2, new
String(clearChars));}
    });
    }
    else
    createStmt.setString(2,null);

    createStmt.setString(3, uname);

    if(enableValue)
        createStmt.setString(4,"Enabled");
    else
        createStmt.setString(4,"Disabled");
    createStmt.setString(5, email);
    createStmt.setString(6, first);
    createStmt.setString(7, last);
    createStmt.setString(8, org);
    createStmt.setString(9, city);

    if (emp!=null)
    createStmt.setInt(10, emp);
    else
    {
    createStmt.setNull(10, java.sql.Types.INTEGER);
    }

    DateFormat formatter = new SimpleDateFormat("dd-MMM-yy");


    if (startdate!=null)
    {
        if (startdate == 0){

        createStmt.setString(11,null);

}
        else
    {
    Date da=new Date(startdate);
    st=formatter.format(da);
    createStmt.setString(11,st);
```

```
    //createStmt.setString(11,null);
    }
    }

    if(enddate!=null)
    {
    if (enddate == 0)
        createStmt.setString(12,null);
        else
    {

    Date eda= new Date(enddate);
    et= formatter.format(eda);
    createStmt.setString(12,et);
       //createStmt.setString(12,null);
    }
    }
    //else createStmt.setObject(12, null);

    createStmt.setString(13, longval);
    if(floatval!=null)
    {
    createStmt.setDouble(14, floatval);}
    else createStmt.setNull(14, java.sql.Types.FLOAT);
    if(charval!=null)
    createStmt.setString(15, charval);
    else createStmt.setObject(15, null);

       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);
"
```

B.7.5 Update Script

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

```
"package groovy
import java.sql.PreparedStatement;
import oracle.jdbc.driver.OraclePreparedStatement;
import java.sql.ResultSet;
import java.text.*;
import java.util.Date.*;
import org.identityconnectors.framework.common.exceptions.*;
import org.identityconnectors.framework.common.objects.*;
import org.identityconnectors.common.security.GuardedString;
import java.text.*;
```

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

//During an Update operation,OIM  sends the UID attribute along with updated
attributes.

//Get all the values of attributes

String uid = attributes.get("__UID__")!=null?
attributes.get("__UID__").getValue().get(0):null;
GuardedString pass = attributes.get("__PASSWORD__")!=null?
attributes.get("__PASSWORD__").getValue().get(0):null;
String uname = attributes.get("__NAME__")!=null?
attributes.get("__NAME__").getValue().get(0):null;
enableValue = attributes.get("__ENABLE__")!=null?
attributes.get("__ENABLE__").getValue().get(0):true;
String email=attributes.get("EMAIL")!=null?
attributes.get("EMAIL").getValue().get(0):null;
String first=attributes.get("FIRSTNAME")!=null?
attributes.get("FIRSTNAME").getValue().get(0):null;
String last=attributes.get("LASTNAME")!=null?
attributes.get("LASTNAME").getValue().get(0):null;
String org=attributes.get("ORG")!=null?
attributes.get("ORG").getValue().get(0):null;
String city=attributes.get("CITY")!=null?
attributes.get("CITY").getValue().get(0):null;
emp = attributes.get("EMPLOYEE_NUMBER")!=null?
attributes.get("EMPLOYEE_NUMBER").getValue().get(0):null;
startdate = attributes.get("STARTDATE")!=null?
attributes.get("STARTDATE").getValue().get(0):null;
enddate = attributes.get("ENDDATE")!=null?
attributes.get("ENDDATE").getValue().get(0):null;
longval = attributes.get("LONGVALUE")!=null?
attributes.get("LONGVALUE").getValue().get(0):null;
floatval = attributes.get("FLOATVALUE")!=null?
attributes.get("FLOATVALUE").getValue().get(0):null;
charval=attributes.get("CHARVALUE")!=null?
attributes.get("CHARVALUE").getValue().get(0):null;


//Throw exception if uid is null
if(uid==null) throw new ConnectorException("UID Cannot be Null");

PreparedStatement upstmt = null;
//upstmt=null;
try {

    //Call Target APIS to update the record

    //stmt =   conn.prepareStatement(....); or stmt =
conn.createStatement(....);
    //upstmt = conn.prepareStatement("UPDATE GENERIC_PARENT SET
PASSWORD=COALESCE(?,PASSWORD),USERNAME=COALESCE(?,USERNAME),STATUS=COALESCE(?,
STATUS), EMAIL= COALESCE(?, EMAIL),FIRSTNAME=COALESCE(?,
FIRSTNAME),LASTNAME =COALESCE(?, LASTNAME), ORGANIZATION=
COALESCE(?, ORGANIZATION), CITY= COALESCE(?, CITY),
EMPLOYEE_NUMBER= COALESCE(?, EMPLOYEE_NUMBER), STARTDATE=COALESCE(to_date(?,'dd-
Mon-yy'), STARTDATE),ENDDATE=COALESCE(to_date(?,'dd-Mon-yy'),
ENDDATE),LONGVALUE=COALESCE(?, LONGVALUE),FLOATVALUE=COALESCE(?,
FLOATVALUE),CHARVALUE=COALESCE(?, CHARVALUE) WHERE USERID =?");
    upstmt = conn.prepareStatement("UPDATE GENERIC_PARENT SET
```

```
PASSWORD=COALESCE(?,PASSWORD),USERNAME=COALESCE(?,USERNAME),STATUS=COALESCE(?,
STATUS), EMAIL= COALESCE(?, EMAIL),FIRSTNAME=COALESCE(?, FIRSTNAME),LASTNAME
=COALESCE(?, LASTNAME), ORG= COALESCE(?, ORG), CITY= COALESCE(?, CITY),
EMPLOYEE_NUMBER= COALESCE(?, EMPLOYEE_NUMBER),STARTDATE=COALESCE(to_date(?,'dd-
Mon-yy'), STARTDATE),ENDDATE=COALESCE(to_date(?,'dd-Mon-yy'),
ENDDATE),FLOATVALUE=COALESCE(?, FLOATVALUE),CHARVALUE=COALESCE(?, CHARVALUE)
WHERE USERID =?");


    //Set the input parameters
    if(pass!=null)
    {
    pass.access(new GuardedString.Accessor(){
   public void access(char[] clearChars) {  upstmt.setString(1, new
String(clearChars));
    System.out.println("password is "+ new String(clearChars)); }
    });
    }
    else
    upstmt.setString(1,null);

    upstmt.setString(2, uname);
    if(enableValue)
        upstmt.setString(3,"Enabled");
    else
        upstmt.setString(3,"Disabled");
    upstmt.setString(4, email);
    upstmt.setString(5, first);
    upstmt.setString(6, last);
    upstmt.setString(7, org);
    upstmt.setString(8, city);
    println("before employee");
    if (emp!=null)
upstmt.setInt(9, emp);
else
{
upstmt.setNull(9, java.sql.Types.INTEGER);
}

            println("before startdate");
DateFormat formatter = new SimpleDateFormat("dd-MMM-yy");
    if (startdate!=null)
    {
    Date da=new Date(startdate);
    st=formatter.format(da);
    upstmt.setString(10,st);
    }
    else
    upstmt.setString(10,null);

    if(enddate!=null)
    {
    Date eda= new Date(enddate);
    et= formatter.format(eda);
    upstmt.setString(11,et);
    }
    else
    upstmt.setString(11,null);


println("before long");
```

```
    //if(longval!=null){upstmt.setLong(12, longval);}
    //else upstmt.setNull(12, java.sql.Types.BIGINT);
println("before float");
    if(floatval!=null){upstmt.setDouble(12, floatval);}
    else upstmt.setNull(12, java.sql.Types.FLOAT);
    if(charval!=null)
    upstmt.setString(13, charval);
    else upstmt.setObject(13, null);
    upstmt.setString(14, uid);

    upstmt.executeUpdate();


} finally {
    if (upstmt!= null)
        upstmt.close();
};
System.out.println("[Update] Updated user::"+ uid);
return new Uid(uid);"
```

B.7.6 Delete Script

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

```
"package groovy;
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 GENERIC_ROLE WHERE USERID=?");
    st.setString(1, uid);
    st.executeUpdate();
//    st.close();

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

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

B.7.7 Add Child Data Script

The following is a sample groovy script for adding multivalued child data:

```
"package groovy
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.

    // Logic for handling simple multi valued attributes

    if(attributes.get("GENERIC_GROUP")!=null) // Here "Groups" is object class
of simple multi-valued attribute
    {

        childDataEOSet=attributes.get("GENERIC_GROUP").getValue();
        childst=conn.prepareStatement("INSERT INTO GENERIC_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);
                //grpattr=  eo;
                if(grpattr!=null){
                    // You are iterating simple multi valued attributes here,
Call target APIs here
                    //conn object is available here
                    groupid=grpattr.getValue().get(0);

groupname=AttributeUtil.find("GROUPNAME",attrsSet).getValue().get(0);

groupstart=AttributeUtil.find("STARTDATE",attrsSet).getValue().get(0);

                    childst.setString(1, id);
                    childst.setString(2, groupname);
                    Date da=new Date(groupstart);

                    SimpleDateFormat formatter = new SimpleDateFormat("dd-MMM-
yy");
                    String st=formatter.format(da);
```

```
                    childst.setString(3, st);
                    childst.setString(4, groupid);

                    childst.executeUpdate();
                    childst.clearParameters();
                }
            };
        }
    }
} finally {
        if (childst != null)
    childst.close();

};

try {
    childDataEOSet = null;
// Logic for handling Complex multi valued attributes
    if(attributes.get("GENERIC_ROLE")!=null)// Here "Roles" is object class of
simple multi-valued attribute
    {

        childDataEOSet=attributes.get("GENERIC_ROLE").getValue();
        childst=conn.prepareStatement("INSERT INTO GENERIC_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(); // Get all the attributes of
child object
                roleattr=AttributeUtil.find("ROLEID",attrsSet);


                // You are iterating complex multi valued attributes here, Call
target APIs here
                //conn object is available here
                if(roleattr!=null){
                    // You are iterating simple multi valued attributes here,
Call target APIs here
                    //conn object is available here
                    roleid=roleattr.getValue().get(0);

rolename=AttributeUtil.find("ROLENAME",attrsSet).getValue().get(0);

rolestart=AttributeUtil.find("STARTDATE",attrsSet).getValue().get(0);

                    childst.setString(1, id);
                    childst.setString(2, rolename);
                    childst.setString(3, roleid);
                    Date da=new Date(rolestart);

                    SimpleDateFormat formatter = new SimpleDateFormat("dd-MMM-
yy");
```

```
                    String st=formatter.format(da);


                    childst.setString(4, st);


                    childst.executeUpdate();
                    childst.clearParameters();
                }

            };
        }
    }
} finally {
if (childst != null)
        childst.close();

};
```

B.7.8 Remove Child Data Script

The section lists a sample groovy script for for removing multivalued child data. The script calls the DELETE_USERGROUP and DELETE_USERROLE stored procedures that have been created on the target system.

The procedure for DELETE_USERGROUP is as follows:

```
"create or replace PROCEDURE DELETE_USERGROUP
(user_id GENERIC_GROUP.USERID%TYPE , group_id GENERIC_GROUP.GROUPID%TYPE )
 AS
BEGIN
DELETE from GENERIC_GROUP where groupid=group_id and userid=user_id;
END DELETE_USERGROUP;"
```

The procedure for DELETE_USERROLE is as follows:

```
"create or replace PROCEDURE DELETE_USERROLE
(user_id GENERIC_ROLE.USERID%TYPE , role_id GENERIC_ROLE.ROLEID%TYPE )
 AS
BEGIN
DELETE FROM GENERIC_ROLE where userid=user_id and roleid=role_id  ;
END DELETE_USERROLE;"
```

The following is a sample groovy script for removing multivalued child data:

```
"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("GENERIC_GROUP")!=null)
    {
        childDataEOSet=attributes.get("GENERIC_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("GENERIC_ROLE")!=null)
    {
        childDataEOSet=attributes.get("GENERIC_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();
};
```

B.7.9 Lookup Field Synchronization Script

The section lists a sample groovy script for performing lookup field synchronization on the Role Name, Group Name, and Organization lookup fields. This script calls the GET_ROLES, GET_GROUPS, and GET_ORGANIZATIONS stored procedures that have been created on the target system.

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 GENERIC_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 GENERIC_GROUPS;
END GET_GROUPS;"
```

The procedure for GET_ORGANIZATIONS is as follows:

```
create or replace PROCEDURE GET_ORGANIZATIONS
( user_cursor OUT TYPES.cursorType
) AS
BEGIN
OPEN user_cursor FOR
SELECT ORGNAME,ORGID from GENERIC.ORGANIZATIONS;
END GET_ORGANIZATIONS;
```

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

```
"package groovy;
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>, objectclass is the
value of object type in schedule job

    // ATTRS_TO_GET and timing are the script arguments

    String codekey =  ATTRS_TO_GET[0];
    String decodekey = ATTRS_TO_GET[1];
    System.out.println("0"+ ATTRS_TO_GET[0]);
    System.out.println("1"+ ATTRS_TO_GET[1]);

    if( timing.equals("executeQuery:ROLENAME"))
```

```
    {
        System.out.println("[Lookup] Getting Roles.");
        st = conn.prepareCall("{call GET_ROLES(?)}");
    }
    else if ( timing.equals("executeQuery:GROUPNAME"))
    {
        System.out.println("[Lookup] Getting Groups.");
        st = conn.prepareCall("{call GET_GROUPS(?)}");

    }
    else if ( timing.equals("executeQuery:ORGANIZATION"))
    {
        System.out.println("[Lookup] Getting Organizations.");
        st = conn.prepareCall("{call GET_ORGANIZATIONS(?)}");

    }
    st.registerOutParameter(1, oracle.jdbc.driver.OracleTypes.CURSOR);
    st.execute();
    rs = st.getObject(1);
    while (rs.next()) {
        cob = new ConnectorObjectBuilder();

    util.addAttribute(codekey,rs.getString(codekey));
    util.addAttribute(decodekey,rs.getString(decodekey));


    if (!util.build()) return;
    }

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

}
"
```

B.7.10 Full and Filtered Reconciliation Script

This section lists a sample groovy script for performing full or filtered reconciliation. This script calls the GET_GENERICGROUP, GET_GENERICROLE, and GENERIC_EXECUTE_QUERY stored procedures for full and filtered reconciliation.

The following stored procedures have been created on the target system for full or filtered reconciliation:

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

"create or replace PROCEDURE GET_GENERICROLE
( user_cursor OUT TYPES.cursorType, userin IN VARCHAR2
) AS
BEGIN
```

```
OPEN user_cursor FOR
SELECT ROLEID,ROLENAME,STARTDATE from GENERIC_ROLE where USERID=userin;
END GET_GENERICROLE;"

"create or replace
PROCEDURE GENERIC_EXECUTE_QUERY
( user_cursor OUT TYPES.cursorType
) AS
BEGIN
OPEN user_cursor FOR
SELECT GENERIC_PARENT.USERID,GENERIC_PARENT.PASSWORD, GENERIC_PARENT.FIRSTNAME ,
GENERIC_PARENT.LASTNAME,
GENERIC_PARENT.EMAIL ,GENERIC_PARENT.ORG,GENERIC_PARENT.CITY,GENERIC_PARENT.START
DATE ,GENERIC_PARENT.EMPLOYEE_NUMBER,GENERIC_PARENT.STATUS,GENERIC_PARENT.USERNAM
E FROM GENERIC_PARENT;
END GENERIC_EXECUTE_QUERY;

To execute procedure from sqldeveloper
var rc refcursor
EXECUTE GENERIC_EXECUTE_QUERY(:rc)
print rc"
```

The script for full and filtered reconciliation is as follows:

```
"package groovy
import org.identityconnectors.framework.common.objects.*;
import java.lang.reflect.*;
import java.lang.String;
import org.identityconnectors.common.security.GuardedString;
import java.text.*;
import java.util.Set;
import java.util.Date.*;
import java.sql.Date.*;
rs = null;
st = null;
try {
    if( !filterMap.isEmpty())
    {
        System.out.println("[Execute Query] Performing Recon with Filter. Filter
is::"+ filterMap+" And Filer Params are::"+filterMap);
        //String[] filter = filterParams.get(0).split(":");
           //   st = conn.prepareCall("{call EXECUTE_QUERY_WITH_FILTER(?,?,?)}");
        //st.setString(2, filter[0]);
        //st.setString(3, filter[1]);
         filterAttr = filterMap.get("filterattribute");
        filterOp = filterMap.get("filteroperator");
    //            if("EQUALto".equalsIgnoreCase(filterOp))
    //    {
    //        dbOp="=";
    //    }
        filterVal = filterMap.get("filtervalue");


    st = conn.prepareCall("{call EXECUTE_QUERY_WITH_FILTER(?,?,?)}");
        st.setString(2, filterAttr);
        st.setString(3, filterVal);
        //st.setString(4, filterOp);

    }
    else // Full recon
    {
```

```
        // // Call target apis to get all records
        System.out.println("[Execute Query] Performing Full Recon.");
        //SELECT GENERIC_PARENT.USERID,GENERIC_PARENT.PASSWORD,
GENERIC_PARENT.FIRSTNAME, GENERIC_PARENT.LASTNAME,
GENERIC_PARENT.EMAIL ,GENERIC_PARENT.ORG,GENERIC_PARENT.CITY,GENERIC_PARENT.START
DATE ,GENERIC_PARENT.EMPLOYEE_NUMBER,GENERIC_PARENT.STATUS,GENERIC_PARENT.USERNAM
E,GENERIC_PARENT.ENDDATE,GENERIC_PARENT.LONGVALUE,GENERIC_PARENT.FLOATVALUE,GENER
IC_PARENT.CHARVALUE FROM GENERIC_PARENT;
        st = conn.prepareCall("{call GENERIC_EXECUTE_QUERY(?)}");
    }
    st.registerOutParameter(1, oracle.jdbc.driver.OracleTypes.CURSOR);
    st.execute();
    rs = st.getObject(1);


    while (rs.next()) {
        cob = new ConnectorObjectBuilder();
        cob.setObjectClass(ObjectClass.ACCOUNT);
        Attribute uid= AttributeBuilder.build(new
String("__UID__"),rs.getString(1));
        Attribute fname= AttributeBuilder.build(new
String("FIRSTNAME"),rs.getString(3));
        Attribute lname= AttributeBuilder.build(new
String("LASTNAME"),rs.getString(4));
        Attribute email= AttributeBuilder.build(new
String("EMAIL"),rs.getString(5));
        Attribute org= AttributeBuilder.build(new String("ORG"),rs.getString(6));
        Attribute city= AttributeBuilder.build(new
String("CITY"),rs.getString(7));
        //dbDate = rs.getDate(8);
        //Attribute startdate = AttributeBuilder.build(new
String("STARTDATE"),rs.getString(8));

        Attribute emp= AttributeBuilder.build(new
String("EMPLOYEE_NUMBER"),rs.getString(9));
        Attribute status= AttributeBuilder.build(new
String("STATUS"),rs.getString(10));
        Attribute name= AttributeBuilder.build(new
String("__NAME__"),rs.getString(11));


        //dbDate = rs.getDate(12);

        //Attribute enddate = AttributeBuilder.build(new
String("ENDDATE"),dbDate.getTime());
        //Attribute longval= AttributeBuilder.build(new
String("LONGVALUE"),rs.getString(13));
        //Attribute floatval= AttributeBuilder.build(new
String("FLOATVALUE"),rs.getString(14));
        //Attribute charval= AttributeBuilder.build(new
String("CHARVALUE"),rs.getString(15));


        cob.addAttribute(fname);
        cob.addAttribute(lname);
        cob.addAttribute(uid);
        cob.addAttribute(name);
        cob.addAttribute(email);
        //cob.addAttribute(startdate);
        //cob.addAttribute(enddate);
        cob.addAttribute(status);
```

```
            cob.addAttribute(org);
            cob.addAttribute(city);
            //cob.addAttribute(longval);
            //cob.addAttribute(floatval);
            //cob.addAttribute(charval);
            cob.addAttribute(emp);

        //    util.addAttribute(Uid.NAME,rs.getString(1));
        //    util.addAttribute(Name.NAME,rs.getString(11)); // Name must be provided
        //    util.addAttribute('FIRSTNAME',rs.getString(3));
        //    util.addAttribute('LASTNAME',rs.getString(4));
        //    util.addAttribute('EMAIL',rs.getString(5));
        //    util.addAttribute('ORGANIZATION',rs.getString(6));
        //    util.addAttribute('CITY',rs.getString(7));
        //    util.addAttribute('STARTDATE',rs.getDate(8));
        //    util.addAttribute('EMPLOYEE_NUMBER',new Integer(rs.getString(9)));
        //
        //    util.addAttribute('STATUS',rs.getString(10));
        //    util.addAttribute('ENDDATE',rs.getDate(12));
        //    util.addAttribute('LONGVALUE',new Long(rs.getString(13)));
        //    util.addAttribute('FLOATVALUE',new Double(rs.getString(14)));
        //    util.addAttribute('CHARVALUE',rs.getString(15));

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

            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));
            Attribute roleName= AttributeBuilder.build(new
String("ROLENAME"),roleResultSet.getString(2));
            //    dbDate = roleResultSet.getDate(3);
            //Attribute startdater = AttributeBuilder.build(new
String("STARTDATE"),dbDate.getTime());

                EmbeddedObjectBuilder roleEA = new EmbeddedObjectBuilder();
                roleEA.addAttribute(roleId);
                roleEA.addAttribute(roleName);
                //roleEA.addAttribute(startdater);
                roleEA.setObjectClass(new ObjectClass("GENERIC_ROLE"));
                eoList.add(roleEA.build());
            }
            roleResultSet.close();
            EmbeddedObject[] roleEm = eoList.toArray(new
EmbeddedObject[eoList.size()]);
            cob.addAttribute(AttributeBuilder.build("GENERIC_ROLE", (Object[])
roleEm));

                    groupStmt = conn.prepareCall("{call GET_GENERICGROUP(?,?)}");
            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>();
            //System.out.println("groupresutset display"+groupResultSet);

            while (groupResultSet.next()) {
```

```
        Attribute groupId= AttributeBuilder.build(new
String("GROUPID"),groupResultSet.getString(1));

        Attribute groupName= AttributeBuilder.build(new
String("GROUPNAME"),groupResultSet.getString(2));
        //    dbDate = groupResultSet.getDate(3);
        //Attribute startdateg = AttributeBuilder.build(new
String("STARTDATE"),dbDate.getTime());
            EmbeddedObjectBuilder groupEA = new EmbeddedObjectBuilder();
            groupEA.addAttribute(groupId);
            groupEA.addAttribute(groupName);
            //groupEA.addAttribute(startdateg);
            groupEA.setObjectClass(new ObjectClass("GENERIC_GROUP"));
            geoList.add(groupEA.build());

        }
        groupStmt.close();


        EmbeddedObject[] groupEm = geoList.toArray(new
EmbeddedObject[geoList.size()]);




    cob.addAttribute(AttributeBuilder.build("GENERIC_GROUP", (Object[])
groupEm));
        if(!handler.handle(cob.build())) return;
    }
} finally {
if( null != rs)
    rs.close();
    if( null != st)
    st.close();

}
"
```

B.7.11 Incremental Reconciliation Script

This section lists a sample groovy script for performing incremental reconciliation. This script calls a stored procedure that has been created on the target system 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
GENERIC_PARENT.USERID,GENERIC_PARENT.PASSWORD,GENERIC_PARENT.FIRSTNAME,GENERIC_PA
RENT.LASTNAME,GENERIC_PARENT.EMAIL,GENERIC_PARENT.ORG,GENERIC_PARENT.CITY,GENERIC
_PARENT.STARTDATE,GENERIC_PARENT.EMPLOYEE_NUMBER,GENERIC_PARENT.STATUS,GENERIC_PA
RENT.USERNAME,GENERIC_PARENT.ENDDATE,GENERIC_PARENT.LAST_UPDATE,to_char(GENERIC_P
ARENT.LAST_UPDATE) FROM GENERIC_PARENT';
else
open user_cursor for 'SELECT
GENERIC_PARENT.USERID,GENERIC_PARENT.PASSWORD,GENERIC_PARENT.FIRSTNAME,GENERIC_PA
RENT.LASTNAME,GENERIC_PARENT.EMAIL,GENERIC_PARENT.ORG,GENERIC_PARENT.CITY,GENERIC
_PARENT.STARTDATE,GENERIC_PARENT.EMPLOYEE_NUMBER,GENERIC_PARENT.STATUS,GENERIC_PA
```

```
RENT.USERNAME,GENERIC_PARENT.ENDDATE,GENERIC_PARENT.LAST_UPDATE,
to_char(GENERIC_PARENT.LAST_UPDATE) FROM GENERIC_PARENT where '|| columnName ||'
>  to_timestamp ('''||columnValue||''')';
end if;
END EXECUTE_QUERY_INCREMENTAL;"
```

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

```
"package groovy;
import org.identityconnectors.framework.common.objects.*;

import java.lang.reflect.*;

import org.identityconnectors.common.security.GuardedString;

import java.text.*;
import java.util.Date.*;
import java.sql.Date.*;

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);

// You have syncattribute and synctoken as script arguments for sync operation
// Call target APIs to get information

st = conn.prepareCall("{call EXECUTE_QUERY_INCREMENTAL(?,?,?)}");
//GENERIC_PARENT.USERID,GENERIC_PARENT.PASSWORD, GENERIC_PARENT.FIRSTNAME,
GENERIC_PARENT.LASTNAME,
GENERIC_PARENT.EMAIL ,GENERIC_PARENT.ORGANIZATION,GENERIC_PARENT.CITY,GENERIC_PAR
ENT.STARTDATE
//,GENERIC_PARENT.EMPLOYEE_NUMBER,GENERIC_PARENT.STATUS,GENERIC_PARENT.USERNAME,G
ENERIC_PARENT.ENDDATE,GENERIC_PARENT.LONGVALUE,GENERIC_PARENT.FLOATVALUE,GENERIC_
PARENT.CHARVALUE
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 uid= AttributeBuilder.build(new
String("__UID__"),rs.getString(1));
        Attribute fname= AttributeBuilder.build(new
String("FIRSTNAME"),rs.getString(3));
        Attribute lname= AttributeBuilder.build(new
String("LASTNAME"),rs.getString(4));
        Attribute email= AttributeBuilder.build(new
String("EMAIL"),rs.getString(5));
        Attribute org= AttributeBuilder.build(new String("ORG"),rs.getString(6));
        Attribute city= AttributeBuilder.build(new
```

```
String("CITY"),rs.getString(7));

        //dbDate = rs.getDate(8);
        //Attribute startdate = AttributeBuilder.build(new
String("STARTDATE"),dbDate.getTime());

        Attribute emp= AttributeBuilder.build(new
String("EMPLOYEE_NUMBER"),rs.getString(9));

        Attribute status= AttributeBuilder.build(new
String("STATUS"),rs.getString(10));
        Attribute name= AttributeBuilder.build(new
String("__NAME__"),rs.getString(11));


        //dbDate = rs.getDate(12);
        //Attribute enddate = AttributeBuilder.build(new
String("ENDDATE"),dbDate.getTime());
        //Attribute longval= AttributeBuilder.build(new
String("LONGVALUE"),rs.getString(13));
        //Attribute floatval= AttributeBuilder.build(new
String("FLOATVALUE"),rs.getString(14));
        //Attribute charval= AttributeBuilder.build(new
String("CHARVALUE"),rs.getString(15));


        cob.addAttribute(fname);
        cob.addAttribute(lname);
        cob.addAttribute(uid);
        cob.addAttribute(name);
        cob.addAttribute(email);
        //cob.addAttribute(startdate);
        //cob.addAttribute(enddate);
        cob.addAttribute(status);
        cob.addAttribute(org);
        cob.addAttribute(city);
        //cob.addAttribute(longval);
        //cob.addAttribute(floatval);
        //cob.addAttribute(charval);
        //cob.addAttribute(emp);


        roleStmt = conn.prepareCall("{call GET_GENERICROLE(?,?)}");
        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));
            Attribute roleName= AttributeBuilder.build(new
String("ROLENAME"),roleResultSet.getString(2));

        //    dbDate = roleResultSet.getDate(3);
        //Attribute startdater = AttributeBuilder.build(new
String("STARTDATE"),dbDate.getTime());

            EmbeddedObjectBuilder roleEA = new EmbeddedObjectBuilder();
            roleEA.addAttribute(roleId);
```

```
            roleEA.addAttribute(roleName);
            //roleEA.addAttribute(startdater);
            roleEA.setObjectClass(new ObjectClass("GENERIC_ROLE"));
            eoList.add(roleEA.build());

        }
        roleResultSet.close();
        EmbeddedObject[] roleEm = eoList.toArray(new
EmbeddedObject[eoList.size()]);
        cob.addAttribute(AttributeBuilder.build("GENERIC_ROLE", (Object[])
roleEm));

        groupStmt = conn.prepareCall("{call GET_GENERICGROUP(?,?)}");
        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 groupName= AttributeBuilder.build(new
String("GROUPNAME"),groupResultSet.getString(2));

            //dbDate = groupResultSet.getDate(3);
        //Attribute startdateg = AttributeBuilder.build(new
String("STARTDATE"),dbDate.getTime());
            Attribute groupId= AttributeBuilder.build(new
String("GROUPID"),groupResultSet.getString(1));

            EmbeddedObjectBuilder groupEA = new EmbeddedObjectBuilder();
            groupEA.addAttribute(groupName);
            //groupEA.addAttribute(startdateg);
            groupEA.addAttribute(groupId);

            groupEA.setObjectClass(new ObjectClass("GENERIC_GROUP"));

            geoList.add(groupEA.build());
        }
        groupStmt.close();
        EmbeddedObject[] groupEm = geoList.toArray(new
EmbeddedObject[geoList.size()]);
        cob.addAttribute(AttributeBuilder.build("GENERIC_GROUP", (Object[])
groupEm));


        Attribute timestamp= AttributeBuilder.build(new
java.lang.String("LAST_UPDATE"),rs.getString(13));

        System.out.println(timestamp)
        token = AttributeUtil.getSingleValue(timestamp);
        System.out.println("token is"+token)
        SyncToken syncToken  = new SyncToken(token);

        SyncDeltaBuilder bld = new SyncDeltaBuilder();

        bld.setObject(cob.build());
        bld.setToken(syncToken);
        bld.setDeltaType(SyncDeltaType.CREATE_OR_UPDATE);
        println bld.build()
        handler.handle(bld.build());
    }
}
```

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

C

Files and Directories of the Generic Scripting Connector

This appendix lists the tables that describe the files and directories corresponding to the Generic Scripting connector.
Table C-1 describes the files and directories on the installation media.

Table C-1 Files and Directories on the Installation Media

File in the Installation Media Directory	Description
bundle/ org.identityconnectors.genericscript-1.0. 11150.jar	This JAR file is the ICF connector bundle.
configuration/GenericScript-CI.xml	This XML file contains configuration information. The Connector Installer uses this XML file to create connector components.
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 location. **Note**: A resource bundle is a file containing localized versions of the text strings that include GUI element labels and messages.
metadata-generator/bin/ GenericScriptGenerator.cmd metadata-generator/bin/ GenericScriptGenerator.sh	This file contains commands to run the metadata generator. Note that the .cmd file is the Microsoft Windows version of the metadata generator. Similarly, the .sh file is the UNIX version of the metadata generator.
metadata-generator/bin/classpath.cmd metadata-generator/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.
metadata-generator/bin/ logging.properties	This file contains the default logging configurations of the metadata generation utility.
metadata-generator/lib/connector-framework-internal.jar	This JAR files contains class files that implement ICF.
metadata-generator/lib/connector-framework.jar	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.
metadata-generator/lib/genericscript-oim-integration.jar	This JAR file contains the class files of the metadata generation utility.
metadata-generator/lib/groovy-all.jar	This JAR file contains the groovy libraries required for running the metadata generator.
metadata-generator/lib/ org.identityconnectors.genericscript-1.0. 11150.jar	This JAR file is the ICF connector bundle. This file is used during metadata generation.

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

File in the Installation Media Directory	Description
metadata-generator/resources/ ScriptConfiguration.groovy	This file contains properties that store basic information about the target system schema, which is used to configure the mode (trusted source or target resource) in which you want to run the connector. In addition, it stores information about the manner in which the connector must connect to the target system. See Configuring the ScriptConfiguration.groovy File for more information about entries in the ScriptConfiguration.groovy file.

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
configuration/*IT_RES_DEF*-CI.xml	This XML file contains configuration information that is used by the Connector Installer during the connector installation process.
resources/genericscript-generator.properties	This property file contains locale-specific properties. You can use this file as a template to add or update locale-related properties.
xml/*IT_RES_DEF*-ConnectorConfig.xml file	This XML file contains definitions for connector components such as IT resource, lookup definitions, scheduled tasks, process forms, and resource objects. This file is also referred to as the connector configuration file.

See Also:

Understanding the Generated Connector Package for information about the structure of the generated connector package

Index