

Oracle® Fusion Middleware
Developer's Guide for Oracle Identity Manager
11g Release 1 (11.1.1)
E14309-09

March 2013

Oracle Fusion Middleware Developer's Guide for Oracle Identity Manager, 11g Release 1 (11.1.1)

E14309-09

Copyright © 1991, 2013, Oracle and/or its affiliates. All rights reserved.

Primary Author: Debapriya Datta

Contributing Author:

Contributor:

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

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

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

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

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

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

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

This software or hardware and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Contents

Preface	xxxiii
Audience.....	xxxiii
Documentation Accessibility	xxxiii
Related Documents	xxxiii
Conventions	xxxiv

Part I Concepts

1 Design Console Overview

1.1	Starting the Design Console	1-1
1.2	Navigating Around the Design Console	1-1
1.2.1	Design Console Menu Bar	1-2
1.2.1.1	File Menu	1-2
1.2.1.2	Edit Menu	1-3
1.2.1.3	Toolbar Menu.....	1-3
1.2.1.4	Help Menu.....	1-3
1.2.1.5	Keyboard Shortcuts in the Design Console	1-3
1.2.2	Design Console Toolbar.....	1-4
1.2.3	Design Console Explorer	1-5
1.2.3.1	Starting a Form.....	1-6
1.2.3.2	Refreshing the List of Forms.....	1-6
1.2.4	Design Console Workspace.....	1-6
1.2.4.1	The Form View.....	1-7
1.2.4.2	The Table View	1-7
1.3	Special Field and Form Types.....	1-9
1.3.1	Data Fields	1-9
1.3.2	Lookup Fields.....	1-10
1.3.3	Date and Time Fields.....	1-10
1.3.4	List.....	1-11
1.3.5	Notes Window	1-11
1.3.6	Tabs on Forms	1-12
1.4	Assignment Windows	1-13
1.5	Search Operations	1-14
1.5.1	Starting a Search.....	1-15

1.5.2	Constructing a Search Filter	1-15
1.5.3	Results of a Search	1-15
1.5.4	Working with a Set of Query Results	1-16
1.5.5	Optimizing Query Performance	1-16
1.5.6	Exceeding the Limit for a Result Set	1-17
1.6	Forms Accessible from the Design Console	1-17
1.6.1	User Management.....	1-18
1.6.2	Resource Management.....	1-19
1.6.3	Process Management.....	1-19
1.6.4	Administration.....	1-20
1.6.5	Development Tools.....	1-20
1.6.5.1	Business Rule Definition.....	1-20

2 Developing Adapters

2.1	Introduction to Adapters	2-1
2.2	Types of Adapters.....	2-3
2.3	Adapter Environment and Tools.....	2-7
2.3.1	Configuring the Adapter Environment.....	2-7
2.3.2	Remote Manager	2-8
2.3.3	The Adapter Factory.....	2-8
2.3.4	Compiling Adapters	2-8
2.3.4.1	Automatic Compilation of Adapters	2-8
2.3.4.2	Compiling Adapters Manually.....	2-9
2.4	Defining Adapters	2-10
2.5	Tabs of the Adapter Factory Form	2-11
2.5.1	Adapter Tasks	2-11
2.5.2	Execution Schedule.....	2-11
2.5.3	Resources	2-12
2.5.4	Variable List.....	2-12
2.5.5	Usage Lookup	2-12
2.5.6	Responses.....	2-13
2.6	Disabling and Re-enabling Adapters	2-13
2.7	About Adapter Variables.....	2-13
2.7.1	Creating an Adapter Variable.....	2-13
2.7.2	Modifying an Adapter Variable	2-15
2.7.3	Deleting an Adapter Variable	2-15
2.8	Creating Adapter Tasks	2-16
2.8.1	Types of Adapter Tasks	2-16
2.8.2	Creating a Java Task	2-17
2.8.3	Creating a Remote Task.....	2-21
2.8.4	Creating a Stored Procedure Task.....	2-23
2.8.5	Creating a Utility Task	2-25
2.8.6	To Create an Oracle Identity Manager API Task	2-27
2.8.7	Reassigning the Value of an Adapter Variable	2-28
2.8.8	Adding an Error Handler Task.....	2-30
2.8.9	Creating a Logic Task.....	2-31
2.9	Modifying Adapter Tasks.....	2-33

2.10	Changing the Order and Nesting of Tasks.....	2-34
2.11	Deleting Adapter Tasks.....	2-35
2.12	Working with Responses	2-35
2.12.1	To Create a Response	2-36
2.12.2	To Modify a Response.....	2-36
2.12.3	To Delete a Response.....	2-37
2.13	Scheduling Rule Generators and Entity Adapters.....	2-37
2.13.1	Scheduling Rule Generators and Entity Adapters.....	2-38

3 Using Adapters

3.1	Working with Rule Generator Adapters	3-1
3.1.1	Mapping Rule Generator Adapter Variables.....	3-1
3.1.2	Associating Rule Generators with Processes.....	3-4
3.1.3	Removing Rule Generators from Form Fields.....	3-4
3.2	Working with Entity Adapters	3-5
3.3	Working with Task Assignment Adapters.....	3-5
3.3.1	Attaching Task Assignment Adapters to Process Tasks.....	3-5
3.3.2	Removing Task Assignment Adapters from Process Tasks.....	3-9
3.3.2.1	To Remove a Task Assignment Adapter from a Process Task	3-9
3.4	Working with Prepopulate Adapters	3-9
3.4.1	Attaching Prepopulate Adapters to Form Fields.....	3-9
3.4.2	Removing Prepopulate Adapters from Form Fields	3-13
3.5	Working with Process Task Adapters.....	3-13
3.5.1	Guidelines for Working with a Process Task Adapter	3-13
3.5.2	Attaching Process Task Adapters to Process Tasks.....	3-14
3.5.3	Removing Process Task Adapters from Process Tasks.....	3-17
3.5.3.1	To Remove a Process Task Adapter from a Process Task	3-18
3.6	Adapter Mapping Information	3-18
3.6.1	Adapter Task Mapping Information.....	3-19
3.6.1.1	Adapter Variables.....	3-19
3.6.1.2	Adapter Task.....	3-19
3.6.1.3	Literal	3-20
3.6.1.4	Adapter References	3-20
3.6.1.5	Organization Definition.....	3-20
3.6.1.6	Process Definition.....	3-21
3.6.1.7	User Definition.....	3-21
3.6.2	Adapter Variable Mapping Information.....	3-22
3.6.2.1	From the Variable List Tab.....	3-23
3.6.2.2	Process Task Adapter Variable Mappings.....	3-23
3.6.2.3	Task Assignment Adapter Variable Mappings.....	3-25
3.6.2.4	Rule Generator and Entity Adapter Variable Mappings.....	3-27
3.6.2.5	Prepopulate Adapter Variable Mappings.....	3-28

4 Using the Callback Service

4.1	Introducing the Callback Service.....	4-1
4.1.1	Using Callbacks.....	4-2

4.1.2	Understanding Event Processing	4-3
4.1.3	Retrying Callbacks.....	4-4
4.2	Mapping Oracle Identity Manager Attributes.....	4-4
4.3	Sending Event Callbacks.....	4-6
4.4	Configuring the Callback Service	4-8
4.4.1	Understanding CallbackConfiguration.xml	4-8
4.4.2	Importing CallbackConfiguration.xml	4-12
4.5	Troubleshooting the Callback Service.....	4-13

5 Developing Rules

5.1	Overview of Business Rule Definition.....	5-1
5.2	Event Handler Manager Form	5-1
5.3	Data Object Manager Form	5-4
5.3.1	Tabs of the Data Object Manager Form.....	5-5
5.3.1.1	Attach Handlers Tab	5-5
5.3.1.1.1	Assigning an Event Handler or Adapter to a Data Object.....	5-6
5.3.1.1.2	Organizing the Execution Schedule of Event Handlers or Adapters	5-6
5.3.1.1.3	Removing an Event Handler or Adapter from a Data Object	5-6
5.3.1.2	Map Adapters Tab.....	5-7
5.4	Reconciliation Rules Form	5-7
5.4.1	Defining a Reconciliation Rule	5-8
5.4.2	Adding a Rule Element.....	5-9
5.4.3	Nesting a Rule Within a Rule.....	5-11
5.4.4	Deleting a Rule Element or Rule	5-11

6 Developing Scheduled Tasks

6.1	Overview of Task Creation.....	6-1
6.1.1	Steps in Task Creation.....	6-1
6.1.2	Example of Scheduled Task	6-2
6.2	Define the Metadata for the Scheduled Task.....	6-2
6.3	Configure the Scheduled Task XML File.....	6-2
6.4	Develop the Scheduled Task Class.....	6-3
6.5	Configure the Plug-in XML File.....	6-4
6.6	Create the Directory Structure for the Scheduled Task.....	6-5

7 Developing Plug-ins

7.1	Background of the Plug-in Framework	7-1
7.1.1	About the Plug-in Framework.....	7-1
7.1.2	About Plug-in Stores	7-2
7.1.2.1	File Store.....	7-2
7.1.2.2	Database Store.....	7-2
7.1.3	Steps for Developing Plug-ins	7-3
7.2	Configuring Plug-ins	7-3
7.3	Defining and Using Plug-ins	7-4
7.3.1	Declaring Plug-ins	7-4
7.3.2	Specifying Plug-in Metadata	7-5

7.3.3	Developing Plug-ins	7-5
7.4	Registering Plug-ins.....	7-6
7.4.1	Registering and Unregistering Plug-ins By Using APIs	7-6
7.4.2	Registering and Unregistering Plug-ins By Using the Plugin Registration Utility....	7-7
7.5	About Mapped Values	7-8
7.5.1	Accessing Mapped Values.....	7-9
7.6	Plug-in Points	7-9

8 Developing Event Handlers for Extending User Management Operations

8.1	An Overview of User Management Operations.....	8-1
8.2	Extending User Management Operations with Event Handlers	8-2
8.2.1	Understanding Elements in Event Handlers XML Files	8-2
8.2.2	Writing Custom Event Handlers.....	8-4
8.2.2.1	Implementing Custom Event Handlers	8-5
8.2.2.2	Creating Plug-ins for Custom Event Handlers	8-8
8.2.2.3	Defining Custom Events.....	8-8
8.3	Troubleshooting an Event Handler.....	8-9

9 Configuring LDAP Container Rules

10 Understanding Context

10.1	Child Context.....	10-1
10.2	Context Types.....	10-2

Part II Application-Specific Connectors

11 Developing Resource Objects

11.1	Viewing Resource Details	11-1
11.2	Working with Users and Organizations Associated with Resources	11-2
11.3	Using the Resource Administrator Option	11-3
11.3.1	Assigning Roles as Administrators for Resources	11-3
11.3.2	Updating Permissions of an Administrative Role	11-4
11.4	Using the Resource Authorizers Option	11-4
11.5	Using the Resource Workflows Option to View Workflows.....	11-4
11.5.1	Opening the Workflow Visualizer	11-5
11.5.2	Elements of the Workflow Visualizer	11-5
11.5.2.1	Using the Provisioning Workflow Definition Event Tabs	11-9
11.5.2.1.1	Provisioning Tab	11-9
11.5.2.1.2	Reconciliation Tab	11-9
11.5.2.1.3	Service Account Tab	11-9
11.5.2.1.4	User Event Tab	11-10
11.5.2.1.5	Org Event Tab	11-10
11.5.2.1.6	Resource Event Tab	11-10
11.5.2.1.7	Form Event Tab	11-10
11.5.2.1.8	Attestation Tab	11-10

11.5.3	Operations on the Workflow Visualizer.....	11-10
11.5.3.1	Rearranging Elements	11-11
11.5.3.2	Using the Expansion Nodes.....	11-13
11.5.3.3	Accessing the Task Details	11-14
11.5.3.3.1	General Tab	11-14
11.5.3.3.2	Automation Tab	11-15
11.5.3.3.3	Task Assignment Tab	11-15
11.5.3.3.4	Depends On Tab	11-15
11.5.3.3.5	Resource Status Management Tab	11-16
11.6	Using the Resource Workflows Option to Create and Modify Workflows	11-16
11.6.1	Opening the Workflow Designer	11-16
11.6.2	Creating a Workflow	11-17
11.6.3	Workflow Designer Main Page.....	11-18
11.6.3.1	Information.....	11-19
11.6.3.2	Toolbar	11-20
11.6.3.2.1	Workflow Configuration.....	11-20
11.6.3.2.2	Task Library	11-21
11.6.3.2.3	Display Options.....	11-22
11.6.3.2.4	Generate Image.....	11-23
11.6.3.2.5	Legend.....	11-23
11.6.3.2.6	Refresh	11-25
11.6.3.2.7	Save.....	11-25
11.6.3.3	Designer Page	11-25
11.6.3.4	Menu Section.....	11-25
11.6.4	Creating and Configuring Tasks and Responses	11-31
11.6.4.1	General Menu Options	11-32
11.6.4.2	Task Options	11-32
11.6.4.3	Response Options	11-33
11.6.4.4	Link Options.....	11-33
11.6.4.5	Configuring Tasks	11-34
11.6.4.6	Configuring Responses.....	11-42
11.6.5	Configuring Data Flows	11-43
11.7	Creating IT Resources	11-45
11.8	Managing IT Resources.....	11-46
11.8.1	Viewing IT Resources.....	11-47
11.8.2	Modifying IT Resources.....	11-47
11.8.3	Deleting IT Resources.....	11-48
11.9	Managing Resources By Using the Design Console	11-48
11.9.1	Overview of Resource Management.....	11-48
11.9.2	IT Resources Type Definition Form	11-49
11.9.2.1	Defining a Template (a Resource Type) for IT Resources	11-50
11.9.2.2	Tabs on the IT Resource Type Definition Form	11-50
11.9.2.2.1	IT Resource Type Parameter Tab	11-50
11.9.2.2.2	IT Resource Tab	11-51
11.9.2.3	IT Resource Type Definition Table	11-51
11.9.3	Rule Designer Form.....	11-51
11.9.3.1	Creating a Rule.....	11-53

11.9.3.2	Tabs on the Rule Designer Form	11-54
11.9.3.2.1	Rule Elements Tab	11-54
11.9.3.2.2	Usage Tab	11-57
11.9.3.3	Rule Designer Table	11-57
11.9.4	Resource Objects Form	11-58
11.9.4.1	Creating a Resource Object	11-60
11.9.4.2	Tabs on the Resource Objects Form	11-61
11.9.4.2.1	Depends On Tab	11-62
11.9.4.2.2	Object Authorizers Tab	11-62
11.9.4.2.3	Process Determination Rules Tab	11-63
11.9.4.2.4	Event Handlers/Adapters Tab	11-64
11.9.4.2.5	Resource Audit Objectives	11-64
11.9.4.2.6	Status Definition Tab	11-65
11.9.4.2.7	Administrators Tab	11-67
11.9.4.2.8	Password Policies Rule Tab	11-68
11.9.4.2.9	User-Defined Fields Tab	11-69
11.9.4.2.10	Process Tab	11-69
11.9.4.2.11	Object Reconciliation Tab	11-70
11.9.4.3	Multiple Trusted Source Reconciliation	11-73
11.9.4.3.1	Multiple Trusted Source Reconciliation Using MTS-Compatible Connectors.....	11-74
11.9.4.3.2	Multiple Trusted Source Reconciliation Using Connectors That Are Not MTS-Compatible	11-76
11.9.5	Service Account Management	11-81

12 Developing Provisioning Processes

12.1	Overview of Process Management	12-1
12.2	Email Definition Form	12-1
12.2.1	Specifying the E-Mail Server	12-2
12.2.2	Email Definition Form	12-3
12.2.3	Creating an E-Mail Definition	12-4
12.3	Process Definition Form	12-5
12.3.1	Creating a Process Definition	12-7
12.3.2	Tabs on the Process Definition Form	12-9
12.3.2.1	Tasks Tab	12-9
12.3.2.1.1	Adding a Process Task	12-10
12.3.2.1.2	Editing a Process Task	12-10
12.3.2.1.3	Deleting a Process Task	12-10
12.3.2.2	Reconciliation Field Mappings Tab	12-10
12.3.2.2.1	User Account Status Reconciliation	12-11
12.3.2.2.2	Mapping a Target Resource Field to Oracle Identity Manager	12-12
12.3.2.2.3	Deleting a Mapping	12-14
12.3.2.3	Administrators Tab	12-14
12.3.2.3.1	Assigning a Role to a Process Definition	12-14
12.3.2.3.2	Removing a Role From a Process Definition	12-14
12.3.3	Modifying Process Tasks	12-15
12.3.3.1	General Tab	12-15

12.3.3.1.1	Modifying a Process Task's General Information.....	12-17
12.3.3.1.2	Triggering Process Tasks for Events Defined in Lookup.USR_PROCESS_TRIGGERS Fields	12-18
12.3.3.2	Integration Tab.....	12-20
12.3.3.2.1	Assigning an Adapter or Event Handler to a Process Task.....	12-20
12.3.3.2.2	Mapping Adapter Variables	12-21
12.3.3.2.3	Removing an Adapter or Event Handler from a Process Task	12-22
12.3.3.3	Task Dependency Tab.....	12-22
12.3.3.3.1	Assigning a Preceding Task to a Process Task.....	12-22
12.3.3.3.2	Removing a Preceding Task from a Process Task	12-22
12.3.3.3.3	Assigning a Dependent Task to a Process Task.....	12-23
12.3.3.3.4	Removing a Dependent Task from a Process Task	12-23
12.3.3.4	Responses Tab.....	12-23
12.3.3.4.1	Adding a Response to a Process Task	12-23
12.3.3.4.2	Removing a Response from a Process Task.....	12-24
12.3.3.4.3	Assigning a Generated Task to a Process Task.....	12-24
12.3.3.4.4	Removing a Generated Task From a Process Task.....	12-25
12.3.3.5	Undo/Recovery Tab	12-25
12.3.3.5.1	Assigning an Undo Task to a Process Task	12-25
12.3.3.5.2	Removing an Undo Task From a Process Task.....	12-26
12.3.3.5.3	Assigning a Recovery Task to a Process Task.....	12-26
12.3.3.5.4	Removing a Recovery Task from a Process Task	12-26
12.3.3.6	Notification Tab	12-26
12.3.3.6.1	Assigning an E-Mail Notification to a Process Task	12-27
12.3.3.6.2	Removing an E-mail Notification from a Process Task.....	12-27
12.3.3.7	Task to Object Status Mapping Tab	12-28
12.3.3.7.1	Mapping a Process Task Status to a Provisioning Status	12-28
12.3.3.7.2	Unmapping a Process Task Status From a Provisioning Status.....	12-29
12.3.3.8	Assignment Tab of the Editing Task Window	12-29
12.3.3.8.1	Adding a Rule to a Process Task.....	12-30
12.3.3.8.2	Removing a Rule from a Process Task.....	12-31

13 Developing Process Forms

13.1	Form Designer Form.....	13-1
13.1.1	Creating a Form	13-3
13.1.2	Tabs of the Form Designer Form.....	13-4
13.1.2.1	Additional Columns Tab.....	13-4
13.1.2.1.1	Adding a Data Field to a Form.....	13-6
13.1.2.1.2	Removing a Data Field From a Form	13-7
13.1.2.2	Child Table(s) Tab	13-8
13.1.2.2.1	Assigning a Child Table to a Form	13-9
13.1.2.2.2	Removing a Child Table from a Form.....	13-9
13.1.2.3	Object Permissions Tab.....	13-9
13.1.2.3.1	Assigning a User Group to a User-Created Form	13-10
13.1.2.3.2	Removing a User Group From a User-Created Form	13-11
13.1.2.4	Properties Tab	13-11
13.1.2.4.1	Adding a Property and Property Value to a Data Field	13-12

13.1.2.4.2	Adding a Property and Property Value for Customized Look up Query	13-13
13.1.2.4.3	Removing a Property and Property Value From a Data Field	13-15
13.1.2.5	Administrators Tab	13-15
13.1.2.5.1	Assigning Privileges to a User Group for a Record of a User-Created Form.....	13-16
13.1.2.5.2	Removing User Group Privileges for a Record of a User-Created Form.	13-16
13.1.2.6	Usage Tab	13-17
13.1.2.7	Pre-Populate Tab	13-17
13.1.2.8	Default Columns Tab	13-17
13.1.2.9	User Defined Fields Tab	13-17
13.1.3	Creating an Additional Version of a Form	13-18
13.2	Error Message Definition Form	13-18
13.2.1	Creating an Error Message	13-19

14 Customizing Reconciliation Operations

14.1	Developing Reconciliation Scheduled Tasks	14-1
14.2	Understanding Reconciliation APIs	14-2
14.3	Postprocessing for Trusted Reconciliation.....	14-3
14.4	Troubleshooting Reconciliation	14-3
14.4.1	Troubleshooting General Reconciliation Issues	14-3
14.4.2	Troubleshooting Trusted Source Reconciliation	14-4
14.4.3	Troubleshooting Target Resource Reconciliation	14-7
14.4.4	Troubleshooting Database-Related Reconciliation Issues	14-9

15 Developing Lookup Definitions, UDFs, and Remote Manager

15.1	Overview	15-1
15.2	Lookup Definition Form	15-1
15.2.1	Creating a Lookup Definition	15-3
15.2.2	Lookup Code Information Tab	15-3
15.2.2.1	Creating and Modifying a Lookup Value	15-4
15.2.2.2	Deleting a Lookup Value.....	15-4
15.2.3	Configuring Challenge Questions for the User	15-5
15.3	User Defined Field Definition Form.....	15-5
15.3.1	Selecting the Target Form for a User-Defined Field	15-6
15.3.2	Tabs on the User Defined Field Definition Form	15-6
15.3.2.1	User Defined Columns Tab.....	15-6
15.3.2.2	Properties Tab	15-10
15.3.2.3	Administrators Tab	15-11
15.4	Remote Manager Form.....	15-11

Part III Identity Connector Framework

16 Understanding the Identity Connector Framework

16.1	Introducing the ICF Architecture	16-1
16.2	Using the ICF API.....	16-3
16.2.1	The ConnectorInfoManagerFactory Class	16-3

16.2.2	The ConnectorInfoManager Interface.....	16-3
16.2.3	The ConnectorKey Class.....	16-4
16.2.4	The ConnectorInfo Interface	16-4
16.2.5	The APIConfiguration Interface	16-4
16.2.6	The ConfigurationProperties Interface.....	16-4
16.2.7	The ConnectorFacadeFactory Class	16-5
16.2.8	The ConnectorFacade Interface	16-5
16.3	Introducing the ICF SPI.....	16-5
16.3.1	Implementing the Required Interfaces	16-5
16.3.1.1	org.identityconnectors.framework.spi.Connector.....	16-6
16.3.1.1.1	Implementing the init Method	16-7
16.3.1.1.2	Implementing the dispose Method.....	16-7
16.3.1.1.3	Implementing the getConfigMethod.....	16-8
16.3.1.2	org.identityconnectors.framework.spi.Configuration	16-8
16.3.1.2.1	The validate() Method	16-9
16.3.1.2.2	The setConnectorMessages() Method	16-9
16.3.1.2.3	The getConnectorMessages() Method.....	16-10
16.3.2	Implementing the Feature-based Interfaces	16-10
16.3.2.1	org.identityconnectors.framework.spi.PoolableConnector.....	16-10
16.3.2.2	org.identityconnectors.framework.spi.AttributeNormalizer	16-11
16.3.3	Implementing the Operation Interfaces	16-12
16.3.3.1	Implementing the SchemaOp Interface.....	16-13
16.3.3.2	Implementing the CreateOp Interface.....	16-13
16.3.3.3	Implementing the DeleteOp Interface.....	16-14
16.3.3.4	Implementing the SearchOp Interface.....	16-15
16.3.3.4.1	Implementing the createFilterTranslator Method	16-15
16.3.3.4.2	Implementing the executeQuery Method.....	16-16
16.3.3.5	Implementing the UpdateOp Interface	16-17
16.3.4	Common Classes.....	16-18
16.4	Extending an Identity Connector Bundle.....	16-19
16.5	Using an Identity Connector Server.....	16-20
16.5.1	Using the Java Connector Server.....	16-22
16.5.1.1	Installing and Configuring a Java Connector Server	16-22
16.5.1.2	Running the Java Connector Server on Microsoft Windows	16-23
16.5.1.3	Running the Java Connector Server on Solaris and Linux	16-24
16.5.1.4	Installing an Identity Connector in a Java Connector Server	16-25
16.5.1.5	Using SSL to Communicate with a Connector Server.....	16-25
16.5.2	Using the Microsoft .NET Framework Connector Server.....	16-25
16.5.2.1	Installing the .NET Connector Server.....	16-26
16.5.2.2	Configuring the .NET Connector Server.....	16-26
16.5.2.3	Configuring Trace Settings.....	16-27
16.5.2.4	Running the .NET Connector Server	16-27
16.5.2.5	Installing Multiple Connectors on a .NET Connector Server	16-27

17 Developing Identity Connectors

17.1	Developing a Flat File Connector	17-1
17.1.1	Supporting Classes for File Input and Output Handling	17-9

17.2	Uploading the Identity Connector Bundle to Oracle Identity Manager Database.....	17-19
17.2.1	Registering the Connector Bundle with Oracle Identity Manager.....	17-19
17.2.2	Creating Basic Identity Connector Metadata.....	17-19
17.2.2.1	Creating the IT Resource Type Definition	17-19
17.2.2.2	Creating the Resource Object.....	17-20
17.2.2.3	Creating Lookups	17-21
17.2.2.3.1	Creating the Main Configuration Lookup.....	17-21
17.2.2.3.2	Creating Object Type Configuration Lookup	17-22
17.2.3	Creating Provisioning Metadata	17-23
17.2.3.1	Creating a Process Form	17-24
17.2.3.2	Creating Adapters	17-26
17.2.3.3	Creating A Process Definition	17-27
17.2.3.4	Creating a Provisioning Attribute Mapping Lookup.....	17-30
17.2.3.4.1	Field Flags Used in the Provisioning Attributes Map.....	17-31
17.2.4	Creating Reconciliation Metadata	17-32
17.2.4.1	Creating a Reconciliation Schedule Task	17-32
17.2.4.1.1	Defining the Schedule Task	17-32
17.2.4.1.2	Creating a Scheduled Task.....	17-33
17.2.4.2	Creating a Reconciliation Profile.....	17-34
17.2.4.3	Setting a Reconciliation Action Rule.....	17-35
17.2.4.4	Creating Reconciliation Mapping	17-36
17.2.4.4.1	Field Flags Used in the Reconciliation Attributes Map	17-37
17.2.4.5	Defining a Reconciliation Matching Rule	17-37
17.3	Provisioning a Flat File Account.....	17-38

Part IV Generic Technology Connectors

18 Understanding Generic Technology Connectors

18.1	Requirement for Generic Technology Connectors.....	18-1
18.2	Functional Architecture of Generic Technology Connectors	18-2
18.2.1	Providers and Data Sets of the Reconciliation Module.....	18-3
18.2.2	Providers and Data Sets of the Provisioning Module	18-4
18.2.3	Oracle Identity Manager Data Sets	18-5
18.3	Features of Generic Technology Connectors	18-5
18.3.1	Features Specific to the Reconciliation Module.....	18-5
18.3.1.1	Trusted Source Reconciliation	18-6
18.3.1.2	Account Status Reconciliation	18-6
18.3.1.3	Full and Incremental Reconciliation	18-7
18.3.1.4	Batched Reconciliation.....	18-7
18.3.1.5	Reconciliation of Multivalued Attribute Data (Child Data) Deletion.....	18-7
18.3.1.6	Failure Threshold for Stopping Reconciliation	18-8
18.3.2	Other Features	18-8
18.3.2.1	Custom Data Fields and Field Mappings	18-8
18.3.2.2	Custom Providers.....	18-8
18.3.2.3	Multilanguage Support.....	18-8
18.3.2.4	Custom Date Formats	18-8

18.3.2.5	Propagation of Changes in Oracle Identity Manager User Attributes to Target Systems	18-9
18.4	Connector Objects Created by the Generic Technology Connector Framework	18-9
18.4.1	Both Reconciliation and Provisioning Are Selected	18-9
18.4.2	Only Reconciliation Is Selected	18-11
18.4.3	Only Provisioning Is Selected	18-11
18.5	Roadmap for Information on Generic Technology Connectors in This Guide	18-11

19 Predefined Providers for Generic Technology Connectors

19.1	Shared Drive Reconciliation Transport Provider	19-1
19.2	CSV Reconciliation Format Provider	19-7
19.3	SPML Provisioning Format Provider	19-7
19.3.1	Run-Time Parameters	19-9
19.3.2	Design Parameters	19-9
19.3.3	Nonmandatory Parameters	19-11
19.3.4	Parameters with Predetermined Values	19-11
19.4	Web Services Provisioning Transport Provider	19-12
19.4.1	Configuring SSL Communication Between Oracle Identity Manager and the Target System Web Service	19-12
19.5	Transformation Providers	19-15
19.5.1	Concatenation Transformation Provider	19-15
19.5.2	Translation Transformation Provider	19-16
19.5.2.1	Configuring Account Status Reconciliation	19-18
19.6	Validation Providers	19-21

20 Creating Custom Providers for Generic Technology Connectors

20.1	Role of Providers	20-1
20.1.1	Role of Providers During Generic Technology Connector Creation	20-1
20.1.2	Role of Providers During Reconciliation	20-3
20.1.3	Role of Providers During Provisioning	20-5
20.2	Creating Custom Providers	20-7
20.2.1	Determining Provider Requirements	20-8
20.2.1.1	Determining the Reconciliation Provider Requirements	20-8
20.2.1.2	Determining the Provisioning Provider Requirements	20-8
20.2.2	Identifying the Provider Parameters	20-9
20.2.3	Developing Java Code Implementations of the Value Objects	20-9
20.2.4	Developing Java Code Implementations of the Provider SPI Methods	20-10
20.2.5	Developing Java Code for Logging and Exception Handling	20-10
20.2.6	Creating the Provider XML File	20-10
20.2.7	Creating Resource Bundle Entries for the Provider	20-13
20.2.8	Deploying the Provider	20-14
20.3	Reusing Providers	20-15
20.3.1	Reusing Reconciliation Providers	20-16
20.3.2	Reusing Provisioning Providers	20-16
20.4	Deploying the Custom Providers	20-17

21 Creating and Managing Generic Technology Connectors

21.1	Overview	21-1
21.2	Creating Generic Technology Connectors	21-1
21.2.1	Determining Provider Requirements	21-2
21.2.2	Selecting the Providers to Include	21-2
21.2.3	Addressing the Prerequisites	21-2
21.2.4	Using the Administrative and User Console to Create the Connector	21-3
21.2.4.1	Step 1: Provide Basic Information Page	21-3
21.2.4.2	Step 2: Specify Parameter Values Page.....	21-6
21.2.4.3	Step 3: Modify Connector Configuration Page	21-15
21.2.4.3.1	Adding or Editing Fields in Data Sets.....	21-21
21.2.4.3.2	Removing Fields from Data Sets	21-28
21.2.4.3.3	Removing Mappings Between Fields.....	21-28
21.2.4.3.4	Removing Child Data Sets	21-28
21.2.4.4	Step 4: Verify Connector Form Names Page	21-29
21.2.4.5	Step 5: Verify Connector Information Page.....	21-30
21.2.5	Configuring Reconciliation	21-33
21.2.6	Configuring Provisioning.....	21-33
21.2.7	Enabling Logging.....	21-35
21.3	Managing Generic Technology Connectors.....	21-35
21.3.1	Modifying Generic Technology Connectors.....	21-36
21.3.2	Exporting Generic Technology Connectors.....	21-37
21.3.3	Importing Generic Technology Connectors.....	21-37
21.4	Using the Generic Connection Pool Framework in Custom Connectors	21-39
21.4.1	Providing concrete implementation for ResourceConnection interface.....	21-39
21.4.2	Defining Additional ITResource Parameters.....	21-40
21.4.3	Getting and Releasing Connections from the Pool	21-41
21.4.4	Using a Third-party Pool.....	21-42
21.4.5	Example: Implementation of ResourceConnection	21-42
21.5	Best Practices	21-44
21.5.1	Working with the Provide Basic Information Page	21-45
21.5.2	Working with the Specify Parameter Values Page	21-45
21.5.3	Working with the Modify Connector Configuration Page.....	21-47
21.5.3.1	Names of Fields	21-47
21.5.3.2	Password Fields	21-47
21.5.3.3	Password-Like Fields	21-47
21.5.3.4	Mappings	21-48
21.5.3.5	Oracle Identity Manager Data Sets	21-49
21.5.4	Working with Shared Drive Reconciliation Transport Provider	21-49
21.5.5	Working with Custom Providers	21-50
21.5.6	Working with Connector Objects	21-50
21.5.7	Modifying Generic Technology Connectors.....	21-51

22 Troubleshooting Generic Technology Connectors

22.1	General Issues for Generic Technology Connectors	22-1
22.1.1	Creation Issues	22-1

22.1.2	Multi-language Support	22-2
22.1.3	Other General Issues	22-6
22.2	Configuration Issues for Generic Technology Connectors	22-7
22.2.1	Names of Generic Technology Connectors and Connector Objects	22-7
22.2.2	Step 3: Modify Connector Configuration Page	22-8
22.2.3	Errors During Connector Creation.....	22-11
22.2.4	Errors During Reconciliation	22-11
22.2.5	Errors During Provisioning	22-13

Part V Requests and Approval Processes

23 Configuring Requests

23.1	Step 1: Creating a Request Dataset for the Resources	23-1
23.1.1	Elements and Properties	23-3
23.1.1.1	The request-data-set Element	23-3
23.1.1.2	The DataSetValidator Element	23-4
23.1.1.3	The AttributeReference Element	23-5
23.1.1.3.1	The PrePopulationAdapter Element	23-9
23.1.1.3.2	The lookupValues Element.....	23-10
23.1.1.3.3	The lookupQuery Element.....	23-11
23.1.1.4	The Attribute Element	23-12
23.1.2	Sample Request Dataset.....	23-12
23.1.3	Child Data	23-15
23.1.4	Common Request Dataset	23-20
23.1.5	Configuring Localized Values for Request Datasets	23-20
23.1.5.1	Localization for Request Dataset Attributes	23-20
23.1.5.2	Localization of Column Names in LookupQuery for Dataset Attributes	23-23
23.2	Step 2: Uploading Request Datasets into MDS.....	23-24
23.3	Step 3: Creating SOA Composites Required for Approval.....	23-24
23.4	Step 4: Registering the SOA Composites in Oracle Identity Manager.....	23-24
23.5	Step 5: Defining Request Approvals	23-25
23.5.1	Approval Workflows	23-25
23.5.2	Approval Levels.....	23-26
23.5.2.1	Template-Level Approval	23-27
23.5.2.2	Request-Level Approval.....	23-27
23.5.2.3	Operation-Level Approval.....	23-28
23.5.3	Creating Approval Policies	23-29
23.6	Step 6: Creating Request Templates.....	23-30
23.7	Extending Request Management Operations	23-30
23.7.1	Running Custom Code Based on Request Status Change.....	23-30
23.7.2	Validating Request Data	23-31
23.7.3	Prepopulation of an Attribute Value During Request Creation.....	23-31

24 Understanding Approval Process Development in Oracle SOA Suite

24.1	Integration with Oracle SOA Suite.....	24-1
24.1.1	Integration Prerequisites.....	24-1

24.1.2	Integration Components.....	24-2
24.2	Predefined SOA Composites.....	24-3
24.3	Developing an Approval Process for Oracle Identity Manager.....	24-4
24.4	Monitoring Oracle Identity Manager SOA Composites.....	24-5
24.5	Enabling Oracle Identity Manager to Connect to SOA	24-5

25 Developing SOA Composites

25.1	Creating New SOA Composites	25-2
25.1.1	Creating a New SOA Composite.....	25-2
25.1.2	Deploying a SOA Composite in Oracle SOA Server	25-3
25.1.3	Prerequisites for Communication to Oracle Identity Manager Through SSL Mode	25-4
25.1.4	Registering a SOA Composite with Oracle Identity Manager	25-4
25.2	Modifying Existing SOA Composites	25-5
25.2.1	Modifying a SOA Project in JDeveloper.....	25-6
25.2.2	Disabling a SOA Composite on Oracle Identity Manager.....	25-6
25.2.3	Deploying a SOA Composite in Oracle SOA Server	25-7
25.2.4	Enabling a SOA Composite with Oracle Identity Manager	25-7

26 Using Oracle Identity Manager APIs in SOA Composites

26.1	Software Prerequisites.....	26-1
26.2	Configuring the SOA Composite By Using JDeveloper.....	26-1
26.2.1	Setting an Application Server Connection in JDeveloper.....	26-2
26.2.2	Setting Up the SOA Composite in JDeveloper	26-2
26.2.3	Updating the SOA Composite	26-3
26.2.4	Deploying the SOA Composite	26-6
26.2.5	Testing the Setup	26-7

Part VI Segregation of Duties

27 Using Segregation of Duties (SoD)

27.1	Understanding the SoD Validation Process.....	27-1
27.2	Introducing the SoD Invocation Library	27-2
27.3	Installing the SoD-enabled Connectors	27-4
27.4	Deploying the SIL and SIL Providers	27-4
27.5	Configuring the SoD Engine	27-4
27.5.1	Configuring Oracle Application Access Controls Governor.....	27-4
27.5.2	Configuring SAP GRC	27-6
27.5.3	Configuring Oracle Identity Analytics	27-7
27.6	Enabling and Disabling SoD	27-8
27.6.1	Enabling SoD	27-8
27.6.2	Disabling SoD.....	27-9
27.7	Enabling SSL Communication	27-10
27.7.1	Enabling SSL Between Oracle Application Access Controls Governor and Oracle Identity Manager	27-10
27.7.2	Enabling SSL Between SAP GRC and Oracle Identity Manager	27-11

27.7.3	Calling SoD Check Web Service Over SSL.....	27-12
27.8	Configuring Workflows on Non SoD-enabled Connectors	27-13
27.8.1	Modifying the Approval Workflow for SoD	27-13
27.8.2	Modifying the Provisioning Workflow for SoD.....	27-30
27.9	Marking Fields as Entitlements	27-34
27.9.1	Marking Request Dataset Attributes That Hold Entitlement Data	27-34
27.9.2	Marking Child Process Form Tables That Hold Entitlement Data	27-34
27.10	Custom Combination of Target Systems and SoD Engines.....	27-35
27.10.1	Using a Custom Target System.....	27-35
27.10.1.1	Addressing Prerequisites	27-35
27.10.1.2	Creating the Transformation Layer	27-36
27.10.1.3	Deploying the Transformation Layer	27-36
27.10.1.4	Modifying the Registration XML File.....	27-36
27.10.1.5	Registering the New Target System.....	27-38
27.10.1.5.1	Running the Registration Script and Providing Registration Information.....	27-38
27.10.1.5.2	Recording the Names of the System Types	27-41
27.10.2	Adding Custom SoD Engine.....	27-42
27.10.2.1	Addressing Prerequisites	27-43
27.10.2.2	Creating an IT Resource to Hold Information about the SoD Engine	27-43
27.10.2.3	Implementing the Service Components for the Provider.....	27-44
27.10.2.4	Deploying the Service Components	27-44
27.10.2.5	Modifying the Registration XML File for the New SoD Engine.....	27-44
27.10.2.6	Registering the New SIL Provider	27-46
27.11	Performing Role SoD Check with Oracle Identity Analytics	27-46
27.11.1	Enabling Role SoD Check	27-47
27.11.2	Using Role SoD Check	27-47
27.11.2.1	SoD Check When A User Requests a Role.....	27-47
27.11.2.2	SoD Check When A User Revokes a Role.....	27-48
27.11.2.3	SoD Check When an Administrator Requests To Assign Roles.....	27-49
27.11.2.4	SoD Check When an Administrator Requests To Revoke Roles.....	27-51
27.11.2.5	SoD Check for Assigning/Revoking Roles with Callback Policy Request.....	27-52
27.12	Using SoD in Provisioning Workflow	27-52
27.12.1	Direct Provisioning.....	27-52
27.12.2	Updating Entitlements.....	27-53
27.12.3	Request Provisioning	27-54
27.12.4	Creating a Request to Modify Provisioned Resource.....	27-54
27.12.5	Request Provisioning With the DefaultSODApproval Workflow	27-55
27.12.6	Request Provisioning with Approver-Only Field and With the DefaultSODApproval Workflow 27-56	
27.12.7	Requesting for Self.....	27-56
27.12.8	Provisioning Based on Access Policies	27-56
27.12.9	Updating Entitlements By Using Provisioning Based on Access Policies.....	27-57
27.13	Enabling Logging for SoD-Related Events.....	27-57
27.14	Troubleshooting SoD Check.....	27-57

Part VII Customization

28 Customizing Oracle Identity Manager Interfaces

28.1	Branding Customization	28-1
28.1.1	Login Page	28-2
28.1.2	Identity Administration	28-4
28.1.3	Unauthenticated Self-Service	28-6
28.1.4	Authenticated Self Service	28-7
28.1.5	Advanced Administration	28-9
28.2	Style Sheet Modifications	28-10
28.2.1	Introduction to the Style Sheets	28-10
28.2.2	Creating Custom Skins and Overriding Style Sheets	28-11
28.2.3	Style Sheets in Transitional UI	28-12
28.2.3.1	Files to Modify	28-12
28.2.3.2	Customizing the Appearance of the Transitional UI	28-12
28.3	Renaming Button Labels	28-12
28.3.1	Identity Administration	28-12
28.3.2	Other Consoles	28-13
28.3.3	Transitional UI Pop-ups	28-14
28.3.3.1	Files to Modify	28-15
28.3.3.2	Customizing Descriptive Text and Labels	28-15
28.4	Working with Menus and Tabs	28-15
28.4.1	Oracle Identity Administration	28-16
28.4.2	Other Consoles	28-17
28.5	Disabling Features	28-18
28.5.1	Disabling Access to Features Through the Authorization Policies	28-18
28.5.2	Other Administration Features	28-19
28.5.3	Other Consoles	28-19
28.6	Adding or Deleting Columns in Console Tables	28-19
28.6.1	Identity Administration	28-19
28.6.2	Transitional UI	28-30
28.6.2.1	Customizing Search Drop-Down Item	28-31
28.6.2.2	Customizing Number of Search Drop-Down Items and Search Results	28-31
28.7	Data Customization	28-32
28.7.1	Advanced Administration	28-32
28.7.2	Unauthenticated Self Service	28-32
28.7.3	Authenticated Self Service	28-32
28.8	Injecting Custom URLs	28-32
28.8.1	Custom URLs for the Identity Administration	28-32
28.8.2	Custom URLs for Other Consoles	28-33
28.9	Changing Popup Properties	28-33
28.10	Customizing the Workflow Designer	28-34

29 Adding Custom ADF Tabs to Self Service

30 General Customization Concepts

30.1	Rule Elements, Variables, Data Types, and System Properties	30-1
30.2	Service Accounts	30-14

30.2.1	Service Account Customization: Scenario One	30-15
30.2.2	Service Account Customization: Scenario Two.....	30-16
30.3	Design Console Actions	30-16

Part VIII APIs and Web Services

31 Using APIs

31.1	Accessing Oracle Identity Manager Services	31-1
31.1.1	Using OIMClient.....	31-1
31.1.2	Using the tcUtilityFactory	31-2
31.2	Oracle Identity Manager Services.....	31-2
31.2.1	Services Introduced in Oracle Identity Manager <i>11g Release 1 (11.1.1)</i>	31-3
31.2.2	Legacy Services or Utilities.....	31-3
31.3	Commonly Used Services	31-3
31.3.1	Mapping Between Legacy and New Services	31-4
31.4	Developing Clients for Oracle Identity Manager	31-4
31.4.1	Prerequisites for Developing Clients	31-4
31.4.2	Setup and Configuration	31-4
31.5	Working With Legacy Oracle Identity Manager APIs	31-5
31.5.1	Using a Result Set Object.....	31-5
31.5.2	Handling Oracle Identity Manager Exceptions.....	31-6
31.5.3	Cleaning Up.....	31-6
31.6	Code Sample	31-6

32 Using SPML Services

32.1	Introduction	32-1
32.1.1	About SPML Interactions	32-2
32.1.2	Integration Interface	32-2
32.2	Create Identity (SPML Core Service: addRequest)	32-2
32.3	Modify Users, Roles, Change Attributes and Role Memberships (SPML Core Service: modifyRequest) 32-3	
32.4	Delete an Identity or Role (SPML Core Service: deleteRequest).....	32-4
32.5	Check Request Status (SPML Core Service: statusRequest)	32-5
32.6	List Available Targets (SPML Core Service: listTargets).....	32-5
32.7	Disable a User (SPML Suspend Service: suspendRequest)	32-6
32.8	Enable a User (SPML Suspend Service: resumeRequest)	32-6
32.9	Check if User is Active (SPML Suspend Service: activeRequest)	32-7
32.10	Validate a Username (SPML Username Service: validateUsername)	32-7
32.11	Obtain a Username (SPML Username: suggestUsername)	32-8
32.12	Reset Password (SPML Core Service: resetPasswordRequest)	32-8
32.13	Lookup Username Policy (SPML Username Service: lookupUsernamePolicy).....	32-8
32.14	Cancel/Withdraw Request (SPML Async Service: cancelRequest)	32-9
32.15	Batch Request (SPML Batch Request Service: batchRequest).....	32-9
32.16	Securing SPML Web Services.....	32-10
32.16.1	About Web Services Security	32-10
32.16.2	A Request Example	32-10
32.16.3	Applying Policies.....	32-11

32.17	Operations Not Supported	32-11
-------	--------------------------------	-------

Part IX Utilities

33 MDS Utilities and User Modifiable Metadata Files

33.1	Setting up the Environment for MDS Utilities	33-1
33.2	Structure of Properties File	33-3
33.3	User Modifiable Metadata Files	33-4
33.4	Example of MDS Utility Usage	33-5

34 Using the Bulk Load Utility

34.1	Features of the Bulk Load Utility.....	34-1
34.2	Installing the Bulk Load Utility	34-2
34.2.1	Scripts That Constitute the Utility.....	34-3
34.2.2	Temporary Tables Used During a Bulk Load Operation.....	34-3
34.2.3	Options Offered by the Utility	34-4
34.3	Preparing Your Database for a Bulk Load Operation	34-5
34.3.1	Creating a Tablespace for Temporary Tables	34-5
34.3.2	Creating a Datafile in the Oracle Identity Manager Tablespace.....	34-5
34.4	Running the Utility	34-6
34.5	Loading OIM User Data.....	34-6
34.5.1	Setting a Default Password for OIM Users Added by the Utility	34-7
34.5.2	Creating the Input Source for the Bulk Load Operation.....	34-8
34.5.2.1	Using CSV Files As the Input Source	34-8
34.5.2.2	Creating Database Tables As the Input Source	34-10
34.5.3	Determining Values for the Input Parameters of the Utility.....	34-11
34.5.4	Monitoring the Progress of the Operation	34-12
34.5.5	Handling Exceptions Recorded During the Operation	34-13
34.5.6	Fixing Exceptions and Reloading Data Records	34-14
34.5.7	Verifying the Outcome of the Bulk Load Operation	34-14
34.6	Loading Account Data	34-15
34.6.1	Creating the Input Source for the Bulk Load Operation.....	34-16
34.6.1.1	Using CSV Files As the Input Source	34-16
34.6.1.2	Creating Database Tables As the Input Source	34-17
34.6.2	Determining Values for the Input Parameters of the Utility.....	34-18
34.6.3	Monitoring the Progress of the Operation	34-19
34.6.4	Handling Exceptions Recorded During the Operation.....	34-20
34.6.5	Fixing Exceptions and Reloading Data Records	34-21
34.6.6	Verifying the Outcome of the Bulk Load Operation	34-21
34.7	Loading Role, Role Hierarchy, Role Membership, and Role Category Data	34-22
34.7.1	Creating the Input Source for the Bulk Load Operation.....	34-22
34.7.1.1	Using CSV Files As the Input Source	34-22
34.7.1.2	Creating Database Tables As the Input Source	34-24
34.7.2	Determining Values for the Input Parameters of the Utility.....	34-24
34.7.3	Monitoring the Progress of the Operation	34-26
34.7.4	Handling Exceptions Recorded During the Operation.....	34-26

34.7.5	Fixing Exceptions and Reloading Data Records	34-27
34.7.6	Verifying the Outcome of the Bulk Load Operation	34-28
34.8	Data Recorded During the Operation.....	34-28
34.9	Gathering Performance Data from the Bulk Load Operation	34-30
34.10	Cleaning Up After a Bulk Load Operation	34-30
34.11	Generating an Audit Snapshot.....	34-31

35 Upload JAR and Resource Bundle Utilities

35.1	Upload JAR Utility.....	35-2
35.2	Download JAR Utility	35-3
35.3	Delete JAR Utility.....	35-3
35.4	Upload Resource Bundle Utility.....	35-4
35.5	Download Resource Bundle Utility	35-4
35.6	Delete Resource Bundle Utility.....	35-4

Part X Reporting

36 Configuring Reports

36.1	What is Oracle Identity Manager Reports?	36-1
36.2	What is Oracle BI Publisher?	36-2
36.3	Supported Products.....	36-2
36.4	Licensing	36-3
36.5	Prerequisites for Deploying Oracle Identity Manager Reports	36-3
36.5.1	Creating the Metadata Repository	36-3
36.5.2	Installing BI Publisher 11g (11.1.1.6)	36-5
36.6	Configuring Oracle Identity Manager Reports	36-6
36.6.1	Deploying Oracle Identity Manager Reports on BI Publisher 11g (11.1.1.6).....	36-6
36.6.2	Configuring Data Sources for Running Oracle Identity Manager Reports.....	36-7
36.6.2.1	Configuring Oracle Identity Manager JDBC Connection.....	36-7
36.6.2.2	Configuring BPEL-Based JDBC Connection.....	36-8
36.7	Generating Oracle Identity Manager Reports	36-9
36.7.1	Generating Sample Reports Against the Sample Data Source.....	36-9
36.7.2	Generating Reports Against the Oracle Identity Manager JDBC Data Source.....	36-9
36.7.3	Generating Reports Against the BPEL-Based JDBC Data Source.....	36-10

37 Developing Entitlements

37.1	Available Entitlements and Assigned Entitlements	37-2
37.2	Entitlement Data Capture Process.....	37-2
37.2.1	Capture of Data About Available Entitlements	37-2
37.2.2	Capture of Data About Assigned Entitlements.....	37-2
37.3	Marking Entitlement Attributes on Child Process Forms.....	37-4
37.4	Configuring Scheduled Tasks for Working with Entitlement Data	37-6
37.4.1	Entitlement List	37-7
37.4.2	Entitlement Assignments.....	37-7
37.4.3	Entitlement Updates.....	37-7
37.5	Disabling the Capture of Modifications to Assigned Entitlements.....	37-8

37.6	Entitlement-Related Reports	37-8
37.6.1	Entitlement Access List	37-9
37.6.2	Entitlement Access List History	37-9
37.6.3	User Resource Entitlement	37-9
37.6.4	User Resource Entitlement History.....	37-9

Part XI Appendixes

A Scheduled Task Configuration File

A.1	Structure of the Scheduler XML File	A-1
A.2	The scheduledTasks Element	A-2
A.3	The task Element	A-2
A.4	The name Element	A-2
A.5	The class Element	A-3
A.6	The description Element	A-3
A.7	The retry Element.....	A-4
A.8	The parameters Element	A-4
A.9	The string-param Element	A-4
A.10	The number-param Element	A-5
A.11	The boolean-param Element	A-6

B SPML Attributes and LDAP Mappings, and Oracle Identity Manager Attributes

B.1	Identity PSO Attributes.....	B-1
B.1.1	Custom Identity Attributes	B-4
B.2	Role PSO Attributes.....	B-4
B.2.1	Custom Role Attributes	B-5
B.3	Preference Attributes.....	B-5
B.4	Special Character Restrictions in Oracle Identity Manager Attributes.....	B-10
B.4.1	Characters Available in All Attributes	B-10
B.4.2	Special Characters in the Password Field	B-10
B.4.3	Usage of Single Quotation Mark	B-10
B.4.4	Usage of Semicolon	B-11
B.4.5	Unsupported Special Characters.....	B-11
B.5	Operation Data	B-11
B.5.1	Passing Operation Data	B-11
B.5.2	Passing Reference Data	B-12

C SPML Examples

C.1	SPML Example - Add User.....	C-2
C.2	SPML Example - Delete User	C-5
C.3	SPML Example - Modify User	C-6
C.4	SPML Example - Resume User	C-7
C.5	SPML Example - Suggest User Name	C-7
C.6	SPML Example - Suspend User	C-8
C.7	SPML Example - Validate User Name.....	C-8
C.8	SPML Example - Check If User is Active	C-9

C.9	SPML Example - Lookup Username Policy	C-9
C.10	SPML Example – Add User with Role Assignment.....	C-10
C.11	SPML Example - Assign Role Membership	C-11
C.12	SPML Example - Add User Request with Notification	C-12
C.13	SPML Example – Revoke Role Membership.....	C-14
C.14	SPML Example - Add Role	C-14
C.15	SPML Example - Add Role with Parent	C-15
C.16	SPML Example - Modify Role.....	C-16
C.17	SPML Example - Add Parent to a Role.....	C-17
C.18	SPML Example - Role Grant.....	C-17
C.19	SPML Example - Delete Role.....	C-18
C.20	SPML Example - Status Request.....	C-18
C.21	SPML Example - Reset Password	C-21
C.22	SPML Example - Reset Password with Notification.....	C-22
C.23	SPML Example - Lookup User Name Policy	C-23
C.24	SPML Example - Cancel Request.....	C-23
C.25	SPML Example - Batch Request.....	C-24

Index

List of Examples

4-1	Sample CallbackConfiguration.xml	4-9
6-1	Sample XML for a Scheduled Task	6-3
6-2	Directory Structure for the Scheduled Task.....	6-5
7-1	Example Plugin.xml	7-8
8-1	Mandatory Namespace Definition for Custom Event Handlers	8-4
8-2	Sample Custom Validation Handler	8-5
8-3	Sample Custom Pre Process Event Handler	8-6
8-4	Sample Custom Post Process Event Handler.....	8-6
8-5	Sample Metadata XML File for Custom Event Definitions	8-8
16-1	ConnectorInfoManagerFactory Implementation	16-3
16-2	ConnectorInfoManager Implementation.....	16-3
16-3	ConnectorKey Implementation.....	16-4
16-4	ConnectorInfo Implementation	16-4
16-5	APIConfiguration Definition.....	16-4
16-6	setPropertyMethod Method Signature.....	16-4
16-7	ConfigurationProperties Implementation	16-5
16-8	ConnectorFacadeFactory Definition	16-5
16-9	ConnectorFacade Implementation	16-5
16-10	Flat File Connector Implementation	16-6
16-11	init Method Implementation	16-7
16-12	dispose Method Implementation.....	16-7
16-13	getConfigMethod Method Implementation.....	16-8
16-14	Configuration Implementation	16-9
16-15	validate Method Implementation.....	16-9
16-16	setConnectorMessages Method Definition	16-10
16-17	getConnectorMessages Method Definition.....	16-10
16-18	Flat File Poolable Connector Implementation	16-11
16-19	checkAlive Method Implementation	16-11
16-20	normalizeAttribute Method Definition.....	16-12
16-21	schema Method Signature	16-13
16-22	schema Method Implementation.....	16-13
16-23	create Method Signature	16-14
16-24	create Method Implementation.....	16-14
16-25	delete Method Signature	16-15
16-26	delete Method Implementation	16-15
16-27	createFilterTranslator Method Signature	16-15
16-28	createFilterTranslator Method Implementation	16-16
16-29	executeQuery Method Signature	16-16
16-30	executeQuery Method Implementation.....	16-16
16-31	update Method Signature	16-17
16-32	update Method Implementation.....	16-17
16-33	Defined Trace Settings.....	16-27
17-1	Implementation of AbstractConfiguration.....	17-2
17-2	Implementation of PoolableConnector	17-4
17-3	Implementation of AbstractFilterTranslator<T>.....	17-8
17-4	The MANIFEST.MF File.....	17-9
17-5	FlatFileIOFactory.....	17-9
17-6	FlatFileMetadata.....	17-10
17-7	FlatFileParser	17-12
17-8	FlatFileWriter	17-15
17-9	Deployment Manager XML with Scheduled Task Details	17-32
21-1	An Example of ResourceConnection Implementation	21-42
23-1	Provisioning AD Resource	23-10
23-2	Create User Dataset	23-12

23-3	Provision E-Business Resource Dataset.....	23-16
26-1	Embedded Java Source Code	26-3
27-1	Sample Run of the Registration Script	27-41
31-1	Retrieving Oracle Identity Manager Information	31-6
34-1	Sample Log File Generated After Loading OIM User Data.....	34-13
34-2	Sample Log File Generated After Loading Account Data	34-20
34-3	Sample Log File Generated After Loading OIM Role Data.....	34-26

List of Figures

1-1	Design Console Main Screen.....	1-2
1-2	Design Console Toolbar.....	1-5
1-3	Design Console Explorer.....	1-6
1-4	Design Console Workspace.....	1-7
1-5	Table View.....	1-8
1-6	Lookup Dialog Box.....	1-10
1-7	The Date & Time Window.....	1-11
1-8	The List Field.....	1-11
1-9	The Notes Window.....	1-12
1-10	Design Console - Tab on Forms.....	1-13
1-11	User Form Assignment Window.....	1-14
1-12	Using a Filter in a Search Query.....	1-15
1-13	Multiple Records Returned.....	1-16
1-14	Query Size Exceeded Dialog Box.....	1-17
1-15	The Design Console Main Screen.....	1-18
2-1	Adapter Functionality.....	2-3
2-2	Adapter Factory Form.....	2-8
2-3	Adapter Manager Form.....	2-9
4-1	Callback Service Process.....	4-4
5-1	Event Handler Manager Form.....	5-2
5-2	Data Object Manager Form.....	5-4
5-3	Reconciliation Rules Form.....	5-8
11-1	Legend Page.....	11-9
11-2	Sample Workflow Displayed in the Workflow Visualizer.....	11-11
11-3	Using Drag-and-Drop in the Workflow Visualizer.....	11-12
11-4	Using the Task Node (Shortcut Menu).....	11-13
11-5	Collapsed Response Subtree in the Workflow Visualizer.....	11-14
11-6	Create Workflow Dialog Box.....	11-17
11-7	Workflow Designer Main Page.....	11-19
11-8	Workflow Configuration Dialog Box.....	11-20
11-9	Task Library Page.....	11-22
11-10	Set Display Options Dialog Box.....	11-23
11-11	Legend Dialog Box.....	11-24
11-12	Add User Event Lookups Dialog Box.....	11-26
11-13	Create Lookup Event Dialog Box.....	11-27
11-14	Edit Lookup Event Dialog Box.....	11-27
11-15	Remove Lookup Event Dialog Box.....	11-27
11-16	Add Organization Event Lookups Dialog Box.....	11-28
11-17	Create Lookup Event Dialog Box.....	11-28
11-18	Edit Lookup Event Dialog Box.....	11-29
11-19	Remove Lookup Event Dialog Box.....	11-29
11-20	Add Resource Event Lookups Dialog Box.....	11-30
11-21	Add Form Event Lookups Dialog Box.....	11-31
11-22	Task Details Dialog Box.....	11-34
11-23	General Tab.....	11-35
11-24	Automation Tab.....	11-36
11-25	Notification Tab.....	11-38
11-26	Task Assignment Tab.....	11-39
11-27	Task Assignment Rule Dialog Box.....	11-40
11-28	Task Details Dialog Box.....	11-41
11-29	Resource Status Management Tab.....	11-42
11-30	Response Details Dialog Box.....	11-43
11-31	Configure Reconciliation Data Flows Page.....	11-44
11-32	The IT Resources Type Definition Form.....	11-49

11-33	Rule Designer Form.....	11-52
11-34	Rule Elements Tab of the Rule Designer Form.....	11-55
11-35	Edit Rule Element Window	11-56
11-36	Usage Tab of the Rule Designer Form	11-57
11-37	Rule Designer Table.....	11-58
11-38	The Resource Objects Form	11-65
11-39	Trusted Source Reconciliation by User Type.....	11-79
11-40	Trusted Source Reconciliation for Specific OIM User Attributes	11-81
12-1	Email Definition Form.....	12-2
12-2	Process Definition Form.....	12-6
12-3	Tasks Tab of the Process Definition Form.....	12-9
12-4	Reconciliation Field Mappings Tab of the Process Definition Form	12-10
12-5	Handler Selection Dialog Box	12-21
13-1	Form Designer Form.....	13-2
13-2	Object Permissions Tab of the Form Designer Form	13-10
13-3	Properties Tab of the Form Designer Form.....	13-11
13-4	Add Property Dialog Box	13-12
13-5	Add Property Dialog Box - Filled	13-13
13-6	Add Property Dialog Box	13-13
13-7	Edit Property Dialog Box.....	13-15
13-8	Administrators Tab of the Form Designer Form.....	13-16
13-9	Error Message Definition Form	13-19
14-1	Invalid Action Rule.....	14-8
14-2	Valid Action Rule.....	14-8
15-1	Lookup Definition Form	15-2
15-2	User Defined Columns Tab of the User Defined Field Definition Form	15-7
15-3	User Defined Fields Dialog Box.....	15-7
15-4	User Defined Fields Dialog Box - Filled	15-9
15-5	Properties Tab of the User Defined Field Definition Form.....	15-10
15-6	Administrators Tab of the User Defined Field Definition Form.....	15-11
15-7	Remote Manager Form.....	15-11
16-1	Identity Connector Framework Deployment	16-2
16-2	Compatibility Between the ICF and Connector Bundles	16-2
16-3	ICF Connectors and Connector Server	16-21
17-1	IT Resource Type Definition in Design Console	17-20
17-2	Resource Objects in Design Console	17-21
17-3	Lookup Definition in Design Console	17-22
17-4	Second Lookup Definition in Design Console.....	17-23
17-5	Form Designer in Design Console.....	17-24
17-6	Properties of Form Designer in Design Console	17-25
17-7	Adapter Factory Variable List in Design Console.....	17-26
17-8	Adapter Factory in Design Console	17-27
17-9	Process Definition in Design Console	17-28
17-10	Editing Task Screen in Design Console	17-28
17-11	Integration Tab in Design Console.....	17-29
17-12	Configure Responses in Design Console.....	17-30
17-13	Task to Object Status Mapping	17-30
17-14	Schedule Task Screen in Advanced Console	17-34
17-15	Object Reconciliation in Design Console.....	17-35
17-16	Reconciliation Action Rules in Design Console	17-36
17-17	Reconciliation Field Mapping in Design Console.....	17-37
17-18	Adding Reconciliation Matching Rule.....	17-38
18-1	Functional Architecture of a Generic Technology Connector	18-3
19-1	Communication Between the SPML Provisioning Format Provider and the Target System..	19-8

20-1	Metadata Detection Process	20-2
20-2	Role of Providers During Reconciliation.....	20-4
20-3	Role of Providers During Provisioning	20-6
21-1	Step 1: Provide Basic Information Page.....	21-6
21-2	First Section of the Step 2: Specify Parameter Values Page	21-14
21-3	Second Section of the Step 2: Specify Parameter Values Page	21-14
21-4	Third Section of the Step 2: Specify Parameter Values Page	21-15
21-5	Step 3: Modify Connector Configuration Page.....	21-17
21-6	Step 3: Modify Connector Configuration Page After Addition of a Field.....	21-29
21-7	Step 4: Verify Connector Form Names Page.....	21-30
21-8	First Section of the Step 5: Verify Connector Information Page.....	21-32
21-9	Second Section of the Step 5: Verify Connector Information Page.....	21-33
23-1	Request Service and SOA Integration.....	23-25
27-1	SoD Validation Process in Oracle Identity Manager	27-2
27-2	Architecture of SoD Implementation in Oracle Identity Manager	27-3
27-3	The SoDCheckAttributes System Attributes.....	27-16
27-4	The TopologyName Parameter	27-17
27-5	Request History for Asynchronous SoD Check	27-17
27-6	Workflow with SoDCheck Web Service Call.....	27-19
27-7	Switch Case With Approval Tasks	27-20
27-8	Assignment of the Approval Task.....	27-21
27-9	Modified Workflow To Perform SoD Check	27-22
27-10	SoD Check Partner Link.....	27-23
27-11	Final Assign Activity	27-24
27-12	The Invoke Dialog Box	27-24
27-13	The Receive Dialog Box.....	27-25
27-14	Switch Case	27-26
27-15	Configuring WS Policies for Request.....	27-27
27-16	Select Client Security Policies.....	27-27
27-17	Select Server Security Policies	27-28
36-1	Oracle Identity Manager Reports Architecture	36-2

List of Tables

2-1	Items on the Map To Menu	2-14
2-2	Options in the Object Instance Selection Window	2-18
2-3	Regions of the Add an Adapter Factory Task Window	2-18
2-4	Regions of the Add an Adapter Factory Task Window	2-24
2-5	Types of Operands.....	2-29
2-6	Actions Resulting from Particular Conditional Statements.....	2-31
2-7	Regions of the Add Adapter Factory Logic Task Window.....	2-32
2-8	Add Adapter Factory Logic Task Parameters for FOR Conditional Statement	2-32
3-1	Fields of the Data Mapping for Variable Dialog Box.....	3-3
3-2	Fields of the Edit Data Mapping for Variable Dialog Box	3-7
3-3	Fields of the Prepopulate Adapter Dialog Box.....	3-10
3-4	Fields of the Map Adapter Variables WInDow.....	3-11
3-5	Fields of the Data Mapping for Variable WInDow	3-16
4-1	Oracle Identity Manager / Callback Service User Attribute Mapping.....	4-5
4-2	Oracle Identity Manager / Callback Service Role Attribute Mapping	4-6
4-3	Callback Initiated Events	4-6
4-4	Trobleshooting Callback Service	4-13
5-1	Fields of the Event Handler Manager Form	5-2
5-2	Fields of the Data Object Manager Form.....	5-4
5-3	Transformation Properties.....	5-10
7-1	Plug-in Points	7-10
8-1	Typical Sub-elements within the eventhandlers Element.....	8-3
8-2	Typical Attributes of Sub-elements within the eventhandlers Element.....	8-3
8-3	SPIs to Write Custom Event Handlers.....	8-5
11-1	Information Fields in the Workflow Visualizer.....	11-5
11-2	Toolbar Menu items in the Workflow Visualizer.....	11-6
11-3	Fields on the General Tab	11-15
11-4	Fields on the Automation Tab.....	11-15
11-5	Fields on the Resource Status Management Tab.....	11-16
11-6	Fields in the Create Workflow Dialog Box.....	11-18
11-7	Fields in the Workflow Configuration Dialog Box	11-21
11-8	Fields of the IT Resources Type Definition Form.....	11-49
11-9	Fields of the Rule Designer Form	11-52
11-10	Fields of the Edit Rule Element Dialog Box	11-55
11-11	Information in the Rule Designer Table	11-58
11-12	Fields of the Resource Objects Form	11-59
11-13	Rule Conditions and Possible Rule Actions.....	11-72
12-1	Fields of the Email Definition Form	12-3
12-2	Fields of the Process Definition Form	12-6
12-3	Fields of the General Tab of the Editing Task Dialog Box	12-15
12-4	Fields of the Assignment Tab of the Editing Task Window	12-29
13-1	Fields of the Form Designer Form.....	13-2
13-2	Fields of the Additional Columns Tab.....	13-4
13-3	Fields of the Add Property Dialog Box.....	13-12
13-4	Fields of the Add Property Dialog Box.....	13-14
13-5	Fields of the Error Message Definition Form.....	13-19
14-1	Troubleshooting Reconciliation.....	14-4
15-1	Fields of the Lookup Definition Form	15-2
15-2	Fields of the User Defined Field Definition Form.....	15-5
15-3	Fields of the User Defined Fields Dialog Box	15-8
16-1	Properties in the ConnectorServer.properties File.....	16-22
16-2	Options Supported by the ConnectorServer.bat Script.....	16-23
16-3	Options Supported by the connectorserver.sh Script.....	16-24

17-1	Form Designer Fields	17-24
19-1	Validation Providers.....	19-21
20-1	Value Objects Used During Provider Operations.....	20-9
20-2	Logging Modules Specific to the Supported Provider Types	20-10
20-3	Elements of the Provider XML File	20-11
21-1	Sample Entries for the Step 1: Provide Basic Information Page.....	21-5
21-2	Sample Entries for the Step 2: Specify Parameter Values Page.....	21-12
21-3	Display of Data Sets and Fields Under Various Input Conditions.....	21-20
21-4	Lookup Properties	21-25
21-5	Methods of ResourceConnection.....	21-40
21-6	ITResource Parameters.....	21-40
21-7	Methods of the GenericPool Interface.....	21-42
22-1	Common Errors Encountered During Reconciliation	22-12
22-2	Common Errors Encountered During Provisioning.....	22-13
23-1	Default Request Datasets Shipped with Oracle Identity Manager.....	23-2
23-2	Request Datasets for Resource Entity	23-4
23-3	Request Types and Associated keys	23-28
24-1	Predefined Workflow Composites	24-3
25-1	Location of Default SOA Composites	25-6
27-1	Variables to Assign	27-23
27-2	Troubleshooting SoD Check.....	27-58
28-1	Files to Modify For Customizing Tables in Identity Administration.....	28-22
28-2	Properties that Determine the Number of Menus on a Search Page.....	28-31
30-1	Rule Elements to Create Oracle Identity Manager Rules.....	30-1
30-2	Variables to Create Templates	30-8
30-3	Properties Associated with Data Types for Creating Oracle Identity Manager Forms.....	30-10
30-4	Service Account Management Tasks and Corresponding APIs	30-15
30-5	Oracle identity Manager Actions, Conditions, and Results	30-17
31-1	Commonly Used Services.....	31-3
31-2	Mapping Between Legacy and New Services.....	31-4
32-1	Identity Creation with addRequest.....	32-3
32-2	Role Membership Management with modifyRequest	32-3
32-3	Role Membership Deletion with deleteRequest	32-4
32-4	Check Request Status	32-5
32-5	Obtaining Targets with listTargets.....	32-5
32-6	Suspending a User with suspendRequest	32-6
32-7	Re-enabling a User with resumeRequest.....	32-6
32-8	Checking if User Has Been Suspended with activeRequest.....	32-7
32-9	Checking Username Validity with resumeRequest.....	32-7
32-10	Obtaining a Username with suggestUsername	32-8
32-11	Resetting the user password with resetPasswordRequest.....	32-8
32-12	Lookup Username policy details with lookupUsernamePolicy	32-9
32-13	Cancel a Request with cancelRequest	32-9
32-14	Executing Batch Request with batchRequest	32-10
33-1	Parameters in the Properties File.....	33-2
34-1	Structure of a Sample Database Table	34-11
34-2	Structure of a Sample Database Table	34-18
34-3	Structure of a Sample Child Database Table.....	34-18
34-4	Structure of a Sample Database Table	34-24
34-5	Structure of the OIM_BLKLD_LOG Table	34-28
36-1	Supported Products	36-2
A-1	Properties of the scheduledTasks Element	A-2
A-2	Properties of the task Element	A-2
A-3	Properties of the name Element.....	A-3

A-4	Properties of the class Element	A-3
A-5	Properties of the description Element.....	A-3
A-6	Properties of the retry Element	A-4
A-7	Properties of the parameters Element.....	A-4
A-8	Properties of the string-param Element.....	A-4
A-9	Properties of the number-param Element.....	A-5
A-10	Properties of the boolean-param Element.....	A-6
B-1	Identity PSO Attributes.....	B-1
B-2	Valid Values of employeeType	B-4
B-3	PSO Role Attributes.....	B-5
B-4	Preference Attributes.....	B-7

Preface

The *Oracle Fusion Middleware Developer's Guide for Oracle Identity Manager* describes how to develop and customize various components and features of Oracle Identity Manager.

Audience

This guide is intended for developers who use Oracle Identity Manager development tools to customize the product according to the requirements of an organization. The customization involves using APIs, configuring request datasets and approval workflows, developing connectors by using Generic Technology Connector or Adapter Factory, and customizing the user interface.

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 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 more information, refer to the following documents:

- *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager*
- *Oracle Fusion Middleware User's Guide for Oracle Identity Manager*
- *Oracle Fusion Middleware Installation Guide for Oracle Identity Management*
- *Oracle Fusion Middleware Enterprise Deployment Guide for Oracle Identity Management*
- *Oracle Fusion Middleware Suite Integration Overview*
- *Oracle Fusion Middleware User Reference for Oracle Identity Management*
- *Oracle Fusion Middleware High Availability Guide*
- *Oracle Fusion Middleware Administrator's Guide for Oracle Access Manager*

- *Oracle Fusion Middleware Administrator's Guide for Oracle Adaptive Access Manager*
- *Oracle Fusion Middleware Developer's Guide for Oracle Adaptive Access Manager*
- *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Navigator*
- *Oracle Fusion Middleware Administrator's Guide for Authorization Policy Manager*

Conventions

The following text conventions are used in this document:

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

Part I

Concepts

This part familiarizes you with tools and features for Oracle Identity Manager developers, and provides some simple examples to illustrate the concepts.

It contains the following chapters:

- [Chapter 1, "Design Console Overview"](#)
- [Chapter 2, "Developing Adapters"](#)
- [Chapter 3, "Using Adapters"](#)
- [Chapter 16, "Understanding the Identity Connector Framework"](#)
- [Chapter 4, "Using the Callback Service"](#)
- [Chapter 5, "Developing Rules"](#)
- [Chapter 6, "Developing Scheduled Tasks"](#)
- [Chapter 7, "Developing Plug-ins"](#)
- [Chapter 8, "Developing Event Handlers for Extending User Management Operations"](#)
- [Chapter 9, "Configuring LDAP Container Rules"](#)
- [Chapter 10, "Understanding Context"](#)

Design Console Overview

You can use the Design Console to configure system settings that control the systemwide behavior of Oracle Identity Manager and affect its users. The Design Console allows you to perform user management, resource management, process management, and other administration and development tasks.

This chapter contains an overview of the Oracle Identity Manager Design Console and describes the basic operations of the console. It is recommended that you review this information before proceeding to subsequent chapters that describe in-depth Design Console features.

This chapter contains the following sections:

- [Starting the Design Console](#)
- [Navigating Around the Design Console](#)
- [Special Field and Form Types](#)
- [Assignment Windows](#)
- [Search Operations](#)
- [Forms Accessible from the Design Console](#)

1.1 Starting the Design Console

To start the Design Console:

1. Double-click the **Oracle Identity Manager** client icon on the desktop.
The Login window is displayed.
2. Enter your user ID and password.
3. Click **Login**.

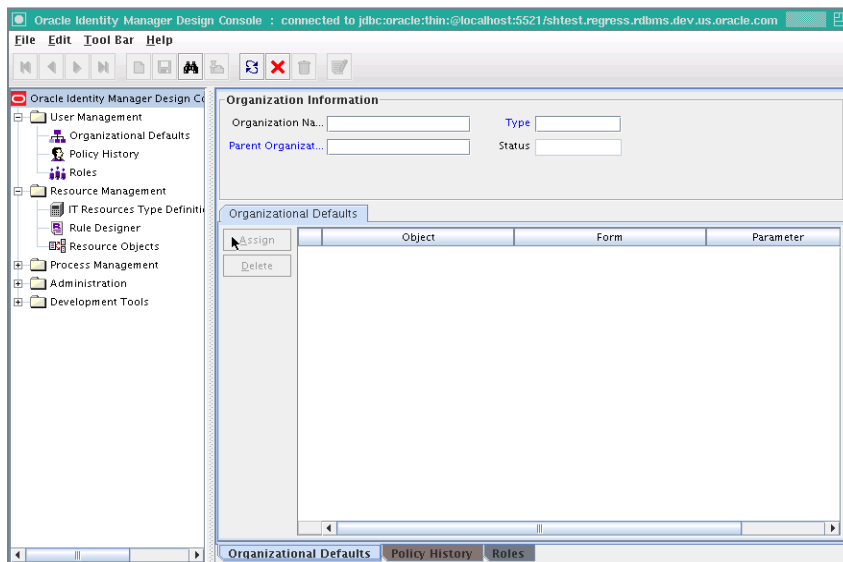
The Design Console main screen is displayed.

Note: You can also access the basic features of Oracle Identity Manager by using the Oracle Identity Manager Administrative and User Console.

1.2 Navigating Around the Design Console

You can create, track, and analyze a business process by using the main screen in the Design Console. [Figure 1–1](#) shows the main screen of the Design Console.

Figure 1–1 Design Console Main Screen



The Design Console main screen consists of these regions:

- [Design Console Menu Bar](#)
- [Design Console Toolbar](#)
- [Design Console Explorer](#)
- [Design Console Workspace](#)

1.2.1 Design Console Menu Bar

The menu bar is displayed at the top of the main screen. It contains menus that enable you to perform all operations in the Design Console user interface.

The Design Console menu bar provides the following menus:

- [File Menu](#)
- [Edit Menu](#)
- [Toolbar Menu](#)
- [Help Menu](#)

You can use keyboard shortcuts to use the menu items for performing various operations in the Design Console. See "[Keyboard Shortcuts in the Design Console](#)" on page 1-3 for information about keyboard shortcuts available in the Design Console.

1.2.1.1 File Menu

The File menu provides the following options:

Menu Item	Action
Print	Prints the active form
Login	Logs out of the Design Console, and log in again
Exit	Exits the Design Console

1.2.1.2 Edit Menu

The Edit menu provides the following options:

Menu Item	Action
Cut	Deletes selected text from editable fields and copies it to the system Clipboard
Copy	Copies the selected text to the system Clipboard
Paste	Pastes text from the system Clipboard to the selected field
Clear	Clears the selected text

1.2.1.3 Toolbar Menu

The **Toolbar** menu operations are described in the following table.

Menu Item	Action
New	Clears the contents of the active form
Save Changes	Saves all changes made to the active form
Query	Runs a query on the active form
Notes	Displays any notes that are attached to the active form
Refresh	Refreshes the record of the active form
Close	Closes the active form
Delete	Deletes the current record
Next	Displays the next record when you query more than one record
Previous	Displays the previous record when you query more than one record
First	Displays the first record when you query more than one record
Last	Displays the last record when you query more than one record
Close All	Closes all open forms and clears the Design Console Workspace

1.2.1.4 Help Menu

The Help menu provides you with access to the Design Console version number and copyright information. These are displayed when you select **About** from the Help menu.

1.2.1.5 Keyboard Shortcuts in the Design Console

The Design Console provides the following keyboard shortcuts to help you perform functions quickly and provide you with easy access to menu commands.

Shortcut Name	Keystroke Combination	Description
File menu	Alt+F	Activates the File menu
Edit menu	Alt+E	Activates the Edit menu

Shortcut Name	Keystroke Combination	Description
Toolbar menu	Alt+T	Activates the Toolbar menu
Help menu	Alt+H	Activates the Help menu
Print	Ctrl+P	Prints the active form
Cut	Ctrl+X	Deletes selected text from editable fields, and copies it to the system Clipboard
Copy	Ctrl+C	Copies the selected text to the system Clipboard
Paste	Ctrl+V	Pastes text from the system Clipboard to the selected field
Clear	Ctrl+Delete	Clears the selected text
New	Ctrl+N	Clears the active form
Save Changes	Ctrl+S	Saves all changes made to the active form
Query	Ctrl+Q	Runs a query on the active form
Notes	Ctrl+Shift+N	Displays notes that are attached to the active form
Refresh	Ctrl+R	Refreshes the active form
Close	Ctrl+W	Closes the active form
Delete	Ctrl+D	Deletes the current record
Next	Number pad + (plus)	Displays the next record, when you have queried more than one record
Previous	Number pad - (minus)	Displays the previous record, when you have queried more than one record
First	Ctrl+F	Display the first record, when you have queried more than one record
Last	Ctrl+L	Displays the last record, when you have queried more than one record
Prepopulate	Ctrl+U	Populates designated fields of a customized form with data
Help	F1	Opens context-sensitive Help for the active form
Explorer	F3	Selects the Design Console icon, which is displayed at the top of the Design Console Explorer
Lookup	F4	Displays the Lookup window for the selected lookup field
Menu	F10	Activates the File menu

1.2.2 Design Console Toolbar

The toolbar consists of a series of buttons below the menu bar. These buttons provide single-click access to frequently used actions. The toolbar buttons apply to the active form.

[Figure 1–2](#) shows the Design Console Toolbar.

Figure 1–2 Design Console Toolbar



When you hold the mouse over a toolbar button for a few seconds, a tool tip that describes the button is displayed.

The following table describes the toolbar buttons:

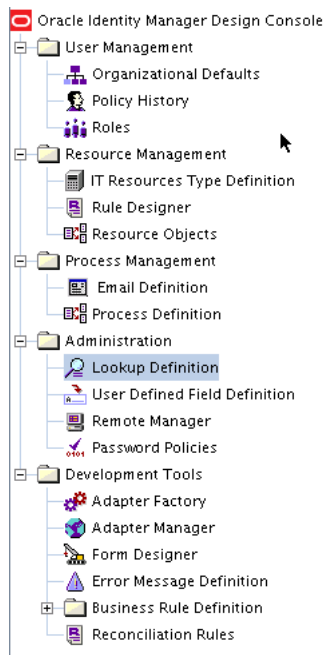
Button	Action
First	Displays the first record when you have queried more than one record.
Previous	Displays the previous record when you have queried more than one record.
Next	Displays the next record when you have queried more than one record.
Last	Displays the last record when you have queried more than one record.
New	Clears the active form.
Save	Saves all changes made to the active form.
Query	Runs a query on the active form.
Notes	Displays any notes that are attached to the active form.
Refresh	Refreshes the active form.
Close	Closes the active form.
Delete	Deletes the current record.
Prepopulate	Populates designated fields with data. These fields are user defined, and have prepopulate adapters attached to them.

1.2.3 Design Console Explorer

The Design Console Explorer contains a list of icons that represent forms that you have permission to access.

Figure 1–3 shows the Design Console Explorer. You can customize the Explorer. Depending on the permissions assigned to you, you can see different icons in the Explorer. If you want to access a form icon that you do not have permissions for, contact your system administrator.

Figure 1–3 Design Console Explorer



Tip:

- If the system administrator changes your permissions, you must refresh the Explorer window.
- You can adjust the size of the Design Console Explorer by moving the divider to the right or left.

1.2.3.1 Starting a Form

To start a form:

1. Expand the folder that contains the required form.
2. Double-click the form that you want to open.

The corresponding form is displayed in the Design Console Workspace.

1.2.3.2 Refreshing the List of Forms

To refresh the list of forms:

1. Right-click the Oracle Identity Manager logo at the top of the Oracle Identity Manager Explorer window. A menu is displayed.
2. Click **Refresh Explorer**.

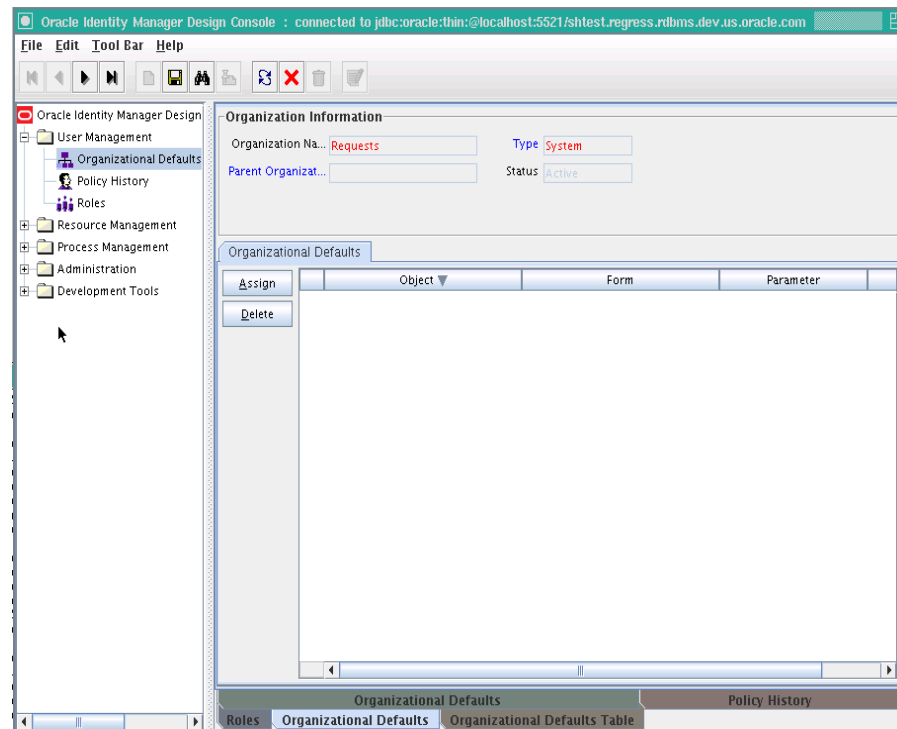
The Design Console refreshes the Explorer with all forms that you can access, including any forms that a system administrator recently gave you permission to access.

1.2.4 Design Console Workspace

The Design Console Workspace is the region of the main screen that displays forms that you access using the Explorer.

Figure 1–4 shows the Workspace.

Figure 1–4 Design Console Workspace



If you access multiple forms, the Design Console places the active form on top and layers the remaining forms on tabs along the bottom of the main screen. To switch between forms, click the desired form's tab.

The Design Console can display each form in two views: a form view and a table view.

1.2.4.1 The Form View

The form view provides detailed information about a single record. The form view is displayed when you initially access a form by using the Explorer, for example, before you perform a query.

1.2.4.2 The Table View

The table view lists general information about multiple records of a form. When you submit a query that produces more than one result, the Design Console displays a table that contains the records that match the criteria in the query.

For example, a query of the Organizations form can return several records. Both the form and table view tabs of the Organizations form can be displayed. [Figure 1–5](#) shows the table view of the Design Console.

Figure 1–5 Table View

	Organization Name	Parent Organization	Type	Status
1	Requests		System	Active
2	Xellerate Users		System	Active
3	org_tc1_2_1		Company	Active

Organizational Defaults		Policy History
Roles	Organizational Defaults	Organizational Defaults Table

The following applies to all table views:

- To select a record in a table view, click it.
- The data associated with a record is displayed in cells.
Cells are also referred to as fields.
- Forms contain column headings, which are boxes with labels above each column.
Column headings display the name of the column. If a column contains a Lookup dialog box, which provides acceptable values for some field or attribute, then the column heading is displayed in blue.
- The Design Console forms contain row headings, which are boxes with numeric labels at the beginning of each row.
To view a detailed form view of a record, double-click its row header. To display a record in the form view, select the record in the table view. Then, click the applicable form tab at the bottom of the Workspace.
- If a query returns more records than can be displayed in the Workspace, a vertical scroll bar is displayed along the right edge of the table view.
Click the up or down arrows in the vertical scroll bar to scroll through the records of the table.
- If the table view contains more columns than can be displayed in the Workspace, then a horizontal scroll bar is displayed along the bottom edge of the table view.
Click the left or right arrows in the horizontal scroll bar to display additional columns not initially visible in the Workspace.
- You can edit record information in the individual cells (fields) of the table view.

To edit the information in a particular field, click it and make the desired changes.

- Fields whose column headings are displayed in blue have Lookup dialog boxes.

You can double-click these fields to access their Lookup dialog boxes, and select the desired value. When you edit the value in any field, the row header for the corresponding record changes to black. This indicates that the data in that field has changed and must be saved.

- To select consecutive records, press the Shift key and use the mouse to select records.
- To select nonconsecutive record rows, press the Ctrl key and use the mouse to select records.
- To export a record, right-click the row heading.

To select more than one record, press the Shift key before clicking the row heading.

A dialog box is displayed.

- Select **Copy to Clipboard** to copy the selected records to the Clipboard.

You can paste copied records into a Microsoft Excel worksheet or a Microsoft Word document.

- To save the records as a tab-delimited file, select **Copy to File**.
- You can control the order in which the records in a table view are displayed by using the sort feature.

To change the sort order of displayed records, click the heading of the column by which you want the records to be sorted. A triangle is displayed beside the column heading text. This indicates the direction, ascending or descending order, in which the records were sorted.

1.3 Special Field and Form Types

The actions of the basic features of the Design Console are standard for all forms. This section describes the standard actions of the Design Console and the field and window types in the Design Console main screen.

1.3.1 Data Fields

Data fields are display areas in forms that present information related to a specific record. For example, First Name can be a data field on the Users form.

The label of a field can be displayed in black or blue.

- A black label indicates that this field is a standard field.
You can query, create, modify, or delete information in a standard field.
- A blue label indicates that the data in this field is derived from a predefined list of values supplied by using a Lookup or a Date & Time window.

When you double-click this type of field, the applicable Date & Time window or Lookup window is displayed. You can select a date, time, or a lookup value.

The value of a field can be displayed in black or red.

- If the field value is displayed in black, the data in this field is supplied by the user.
You can query or edit the information in these types of fields.

- If the field value is displayed in red, the data in this field is supplied by Oracle Identity Manager.

These values are read-only. This prevents you from overwriting critical information.

1.3.2 Lookup Fields

A lookup field enables you to search for a value. Lookup fields are displayed in blue. The following procedure describes how to use lookup fields.

To use lookup fields:

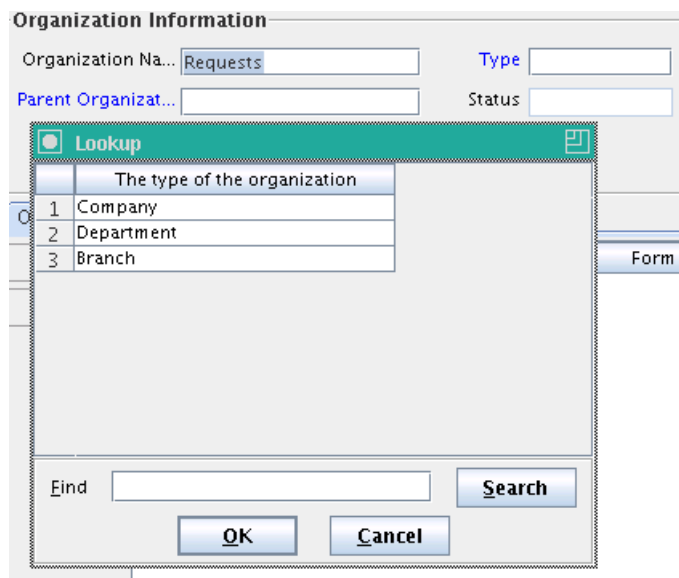
1. Double click the lookup field. The Lookup dialog box is displayed.
2. In the Lookup dialog box, to select a value, click the field, and then click **OK**.

Alternatively, you can select the field and press F4.

Click **Cancel** to close the Lookup window without selecting anything.

Figure 1–6 shows the Lookup dialog box.

Figure 1–6 Lookup Dialog Box



3. If the Lookup dialog box contains a long list of values, enter the first few characters of the value in the Find box, followed by an asterisk (*), and click **Search**. Alternatively, you can scroll through the list of values to locate it.

The Lookup dialog box displays the results that match your search.

1.3.3 Date and Time Fields

The Date & Time window enables you to select a month, year, day, and time. This window is displayed when you double-click a field that is equipped with a an option to open this window. The Date & Time fields have labels in blue.

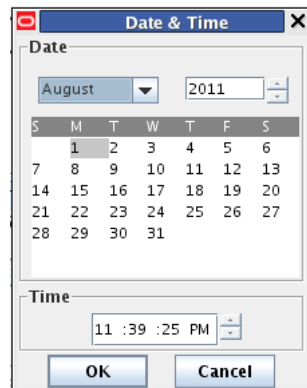
To select a date and time:

1. Double-click the field in which you want to enter a date and time.

You can also display the Date & Time window by selecting a field and pressing **F4**.

The Date & Time window is displayed, as shown in [Figure 1-7](#):

Figure 1-7 The Date & Time Window



2. From the menu, select the month.
3. From the **Date** scroll box, select the year.
4. Click the date on the calendar.
5. From the **Time** box, select the time.
6. Click **OK** to save your changes.

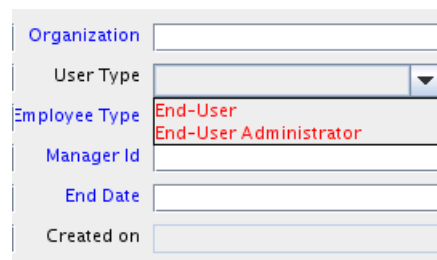
The Date & Time window closes. The field that you double-clicked in Step 1 now displays the date and time you selected.

Click **Cancel** to exit without saving.

1.3.4 List

Lists have predefined values. When you click a list, its values are displayed. If the list contains more values that can be displayed at one time, then a vertical scroll bar is displayed to the right of the list. In [Figure 1-8](#), the Employee Type field is a list:

Figure 1-8 The List Field



When you select a value, the list is replaced by a field in which the selected value is displayed.

1.3.5 Notes Window

The Notes window enables you to enter supplemental information for a record. When used with adapters, this window also displays the code that the Design Console generated while compiling the adapter. For more information about adapters, see [Chapter 2, "Developing Adapters"](#) and [Chapter 3, "Using Adapters"](#).

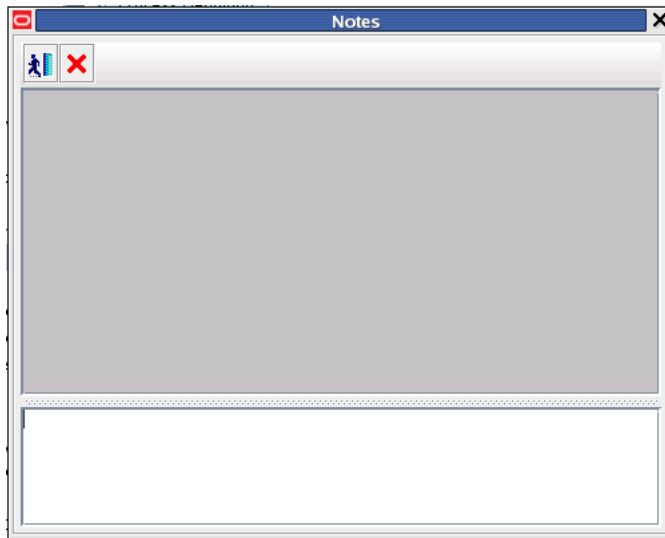
Note: In the following procedure, if the Notes button is red, the current record has a note. To view the note, click the button. You can enter supplemental information in this record. Each entry receives a unique date, time, and user stamp.

To use the Notes window:

1. Select the required record.
2. Click **Notes**.

The Notes window is displayed, as shown in [Figure 1-9](#):

Figure 1-9 The Notes Window



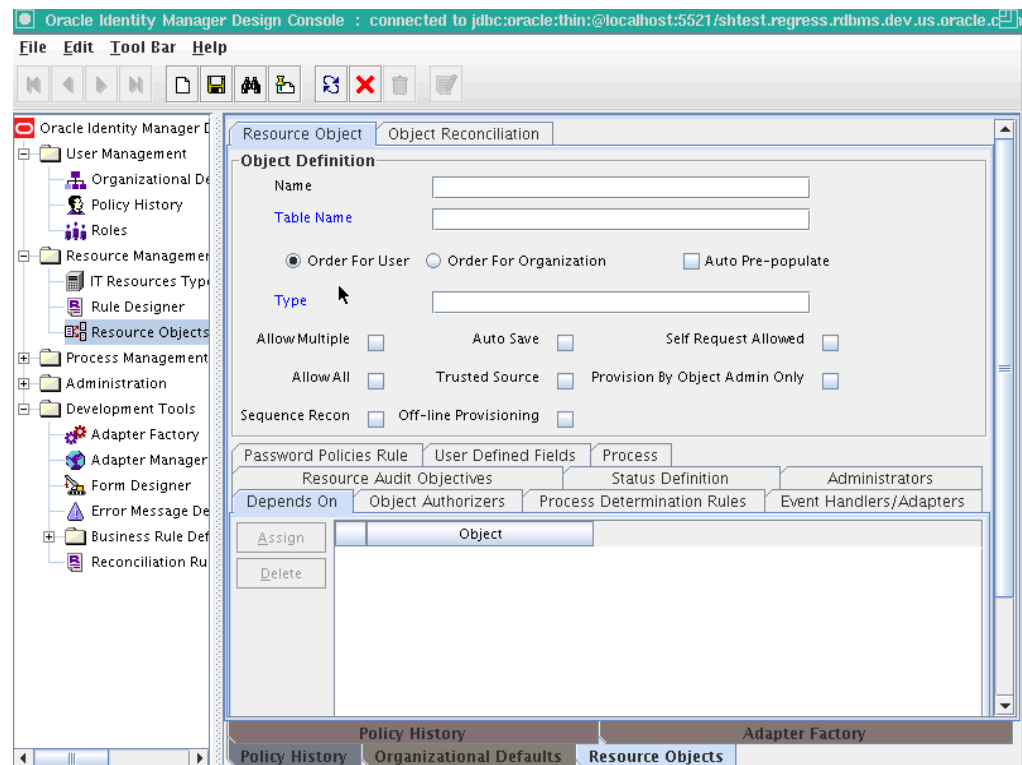
3. Enter information in the text area of the Notes window.
4. Click the icon that represents a man to store your information in the Notes window.
Or, click **Close** to close the Notes window without saving.
5. From the Toolbar, click **Save**.

The information you entered into the Notes window is saved.

1.3.6 Tabs on Forms

Most forms in the Design Console contain multiple tabs. The tabs are usually in the bottom of the form. The tabs display additional information about a record, for example, the users who are employed at an organization, as shown in [Figure 1-10](#).

Figure 1–10 Design Console - Tab on Forms



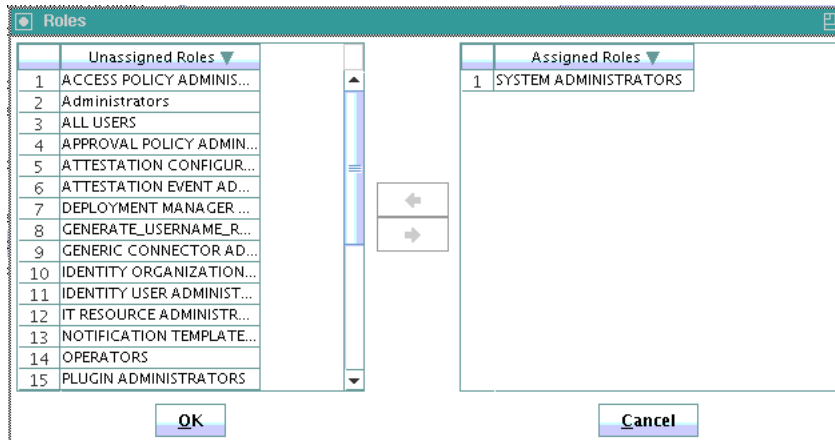
Each tab has its own tables and function buttons. Usually, the buttons on a tab are grayed out until the information in the upper portion of the form is saved. The table displayed in the tab enables you to view and edit the records associated with that tab item.

To modify information in a row of a tab's table, either double-click the field that contains the information you want to edit, or double-click the associated row heading.

1.4 Assignment Windows

The User Form Assignment windows enable you to select and assign entities to a record. The Assignment window is displayed when you click the **Assign** button.

Figure 1–11 shows a User Form Assignment window for selecting and assigning roles to a record.

Figure 1–11 User Form Assignment Window

The left pane lists items that you can assign to the record, for example, Organization. The right pane lists the items that have already been assigned to the record. Although the values available for selection in the left and right panes are unique to what is being assigned or unassigned, the buttons and general use of this dialog box are consistent throughout the application.

The following are methods for working with this window:

- To select multiple unconsecutive items, hold down the Ctrl key while selecting items with the mouse.
For example, you can select the User Group, the IT Resource Type Definition object, and the Form Information object, but not the Process Definition object.
- To select multiple items that are listed consecutively, hold down the Shift key and select the first and last items with the mouse.
- To assign one or more items, select the item and click the right arrow.
- To unassign one or more items, select them, and click the left arrow.

When you are done, click **OK**. If you click **Cancel**, all assignment changes are discarded.

1.5 Search Operations

This section describes the search operation that you can perform in the Design Console. It contains the following sections:

- [Starting a Search](#)
- [Constructing a Search Filter](#)
- [Results of a Search](#)
- [Working with a Set of Query Results](#)
- [Optimizing Query Performance](#)
- [Exceeding the Limit for a Result Set](#)

1.5.1 Starting a Search

The Design Console enables you to perform searches (queries) for records in the database. Every form in the Design Console provides a search function. The search function is also available in lookup fields.

To conduct a search on a blank form or after entering a search filter, click the binoculars icon on the toolbar.

After you enter the search criteria in the query fields, click the binoculars symbol or press Ctrl+Q.

1.5.2 Constructing a Search Filter

You can filter the search results in a form field. Filtering limits the results that are returned to only the records that match the criteria you entered. If you leave all form fields blank before conducting the search, all records in the table are returned.

You can use a wildcard character in a search. The asterisk (*) wildcard character represents unspecified portions of the search criteria. You can use a wildcard character at the beginning, middle, or end of the value that you enter in a field. For example, if you enter B* in the Location field of a Design Console form and execute a search, you retrieve all records with locations that begin with the letter B (for example, Burbank, Boston, Bristol, and so on). If the asterisk is placed in the middle of a search value, as in B*on, you retrieve all records that begin with B and end with ON (for example, Brighton, Boston, and so on). If you place the asterisk at the beginning of the search value, as in *A, you retrieve all records that end in A (for example, Philadelphia, Tampa, and so on).

In [Figure 1–12](#), a query is performed on the Organizational Defaults form and the Organization Name field is used to filter the search criteria. The filter Xell* ensures that only organizations with names that begin with Xell are retrieved.

Figure 1–12 Using a Filter in a Search Query

The screenshot shows a window titled "Organization Information" with several input fields. The "Organization Na..." field contains the text "Xell*" in red. Other fields include "Type", "Parent Organizat...", and "Status". Below this window is another window titled "Organizational Defaults" which contains a table with columns "Object", "Form", and "Parameter". On the left side of the "Organizational Defaults" window, there are buttons for "Assign" and "Delete". A mouse cursor is visible over the table area.

1.5.3 Results of a Search

When you submit a search request by clicking the toolbar icon (Query for records), one of the following occurs:





- **No records are returned.** No records in the database match your search criteria for this form. Either the record that you are searching for no longer exists in the database, or you must modify your search criteria.
- **One record is returned.** One record in the database matches your search criteria. The Form view displays that record.
- **More than one record is returned.** Multiple records in the database match your search criteria. A Table view is displayed, listing all records that meet your search criteria. The first record is displayed in the Form view, as shown in [Figure 1–13](#).

Figure 1–13 Multiple Records Returned

	Organization N...	Parent Organization	Type	Status
1	Requests		System	Active
2	Xellerate Users		System	Active
3	org_tc1_2_1		Company	Active

1.5.4 Working with a Set of Query Results

If multiple records in the database match your search criteria, you can view details about each record. Several buttons can assist you when viewing these records in the Form view. These directional buttons, referred to as VCR buttons, are located in the toolbar. The following table describes the VCR buttons:

Buttons	Description
	Click this button to display the first record in the result set in the Form view.
	Click this button to display the preceding record according to the display sequence in the Table view. The record is displayed in the result set in the Form view.
	Click this button to display the next record (according to the display sequence in the Table view) in the result set in the Form view.
	Click this button to display the last record in the result set in the Form view.

1.5.5 Optimizing Query Performance

A query that returns a large result set can require significant time to run and can affect your computer's performance. To optimize performance, use the following search techniques:

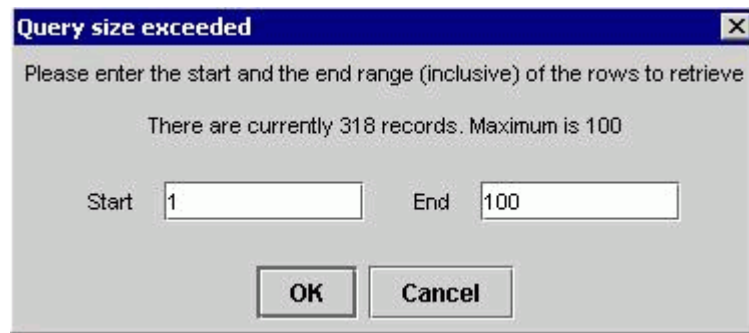
- Define the scope of a search strategy as precisely as possible.
Enter the most specific information that you can when constructing your query. For example, if the first name of a contact is JOHN and the last name is JACKSON, enter both pieces of information, rather than searching only for contacts with the last name JACKSON.
- Use the asterisk (*) wildcard character where possible.

If you place the asterisk in front of an alphabetic character (for example, *A), fewer records are returned as compared to when you leave a field blank.

1.5.6 Exceeding the Limit for a Result Set

If you have both read and write access to all forms and records in the Design Console (that is, if you are a system administrator), you can set the maximum number of records that are displayed in the result set for a search. If the number of records retrieved for a search exceeds this value, the Design Console displays the Query size exceeded dialog box, as shown in [Figure 1-14](#).

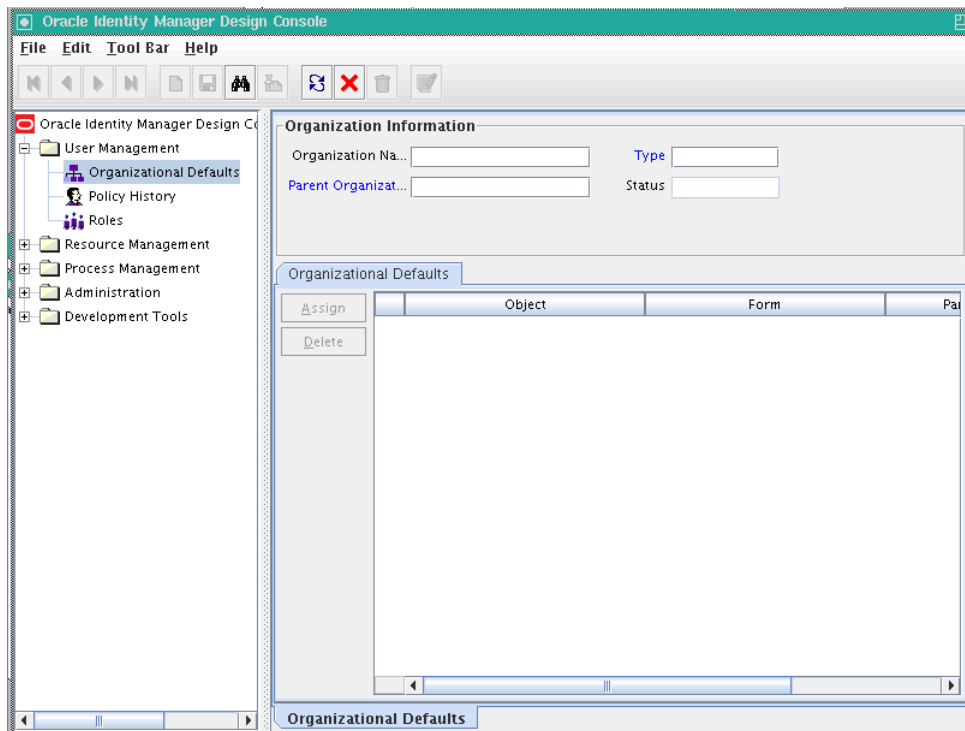
Figure 1-14 Query Size Exceeded Dialog Box



You are prompted to enter a specific range or subset of the result set to be viewed. In [Figure 1-14](#), the maximum result set of 100 has been exceeded. Only records 1 through 100 will be displayed.

1.6 Forms Accessible from the Design Console

The left side of the Design Console main screen is the Design Console Explorer. The Explorer consists of a list of icons that represent forms that you have permissions to access. These icons are grouped under folders based on functionality and are displayed to the users according to the access rights that you assign to them. [Figure 1-15](#) shows the Explorer in the Design Console main screen.

Figure 1–15 The Design Console Main Screen

The forms in the various folders in the Design Console Explorer are described in the following sections:

- [User Management](#)
- [Resource Management](#)
- [Process Management](#)
- [Administration](#)
- [Development Tools](#)
- [Business Rule Definition](#)

1.6.1 User Management

The User Management folder provides tools to create and manage information about organizations, users, and roles. This folder contains the following forms:

- **Organizational Defaults:** This form is used to specify the default values that the organization users should have for certain resources. The organization hierarchy is considered while getting the values specified on the organization defaults by traversing from the bottom of the hierarchy to the top.
- **Policy History:** This form is used to view resources that are allowed and disallowed for users through policies.
- **Roles:** This form is used to specify which Design Console forms are available for which roles.

1.6.2 Resource Management

The Resource Management folder provides you tools for managing Oracle Identity Manager resources. This folder contains the following forms:

- **IT Resources Type Definition:** This form is used to create resource types that are displayed as lookup values on the IT Resources form.

IT resource types store the set of fields that are used to specify connectivity and other configuration of a particular target type. For example, the AD Server IT resource type indicates what fields must be specified for connecting to Microsoft Active Directory. This can include the host, port, username, password, and root context. This is essentially a list of parameters. From this type, you can create an IT resource instance by specifying concrete values for all the parameters. While provisioning an Active Directory account, one of the fields on the process form is of type AD Server, and the value of this field is the IT resource instance that points to the actual target where the account is created.

- **Rule Designer:** This form is used to create rules that can be applied to password policy selection, auto-group membership, provisioning process selection, task assignment, and prepopulating adapters.
- **Resource Objects:** This form is used to create and manage resource objects. These objects represent resources that you want to make available to users and organizations.

Resource objects specify account types in Oracle Identity Manager. Resource objects encapsulate the following:

- The type of the entity that the resource objects are representing in the target, for example AD User and AD Group.
- The provisioning mechanism. This is not a configuration specific to resource objects but a provisioning process definition. However, the process definition itself is tied to the resource object.
- Flags related to permissioning that determines who is the resource allowed for, if the user can get more than one instances of this resource, and who can provision the resource.
- Object administrator and object authorizer configuration.
- Reconciliation fields, action rules and other reconciliation related flags.
- Resource dependency.
- Status flags for a resource, which determines the states that a resource can go through.
- Resource audit objectives used in attestation.

Examples of resource objects are AD User, SAP User, and LDAP Account.

See Also: [Chapter 11, "Developing Resource Objects"](#) for more information about the forms in the Resource Management folder

1.6.3 Process Management

The Process Management folder provides you tools for creating and managing Oracle Identity Manager processes and e-mail templates.

This folder contains the following forms:

- **Email Definition:** This form is used to create templates for e-mail notifications.

- **Process Definition:** This form is used to create and manage approval and provisioning processes. It also lets you start the Workflow Definition Renderer that displays your workflow definition in a graphical presentation.

See Also: [Chapter 12, "Developing Provisioning Processes"](#) for more information about the forms in the Process Management folder

1.6.4 Administration

The Design Console Administration folder provides you tools for managing Oracle Identity Manager administrative features. This folder contains the following forms:

- **Lookup Definition:** This form is used to create and manage lookup definitions. A lookup definition represents a lookup field and the values you can access from that lookup field.
- **User Defined Field Definition:** This form is used to create and manage user-defined fields. A user-defined field enables you to store additional information, such as user, request, and resource information.
- **Remote Manager:** This form is used to display information about the servers that Oracle Identity Manager uses to communicate with third-party programs. These servers are known as remote managers.
- **Password Policies:** This form is used to set password restrictions for the users and view the rules and resource objects that are associated with a password policy.

See Also: [Chapter 15, "Developing Lookup Definitions, UDFs, and Remote Manager"](#) for more information about the forms in the Administration folder

1.6.5 Development Tools

The Design Console provides a suite of development tools that enable system administrators or developers to customize Oracle Identity Manager. This folder contains the following forms:

- **Adapter Factory:** This form is used to create and manage the code that enables Oracle Identity Manager to communicate with any IT Resource by connecting to that resource's API. This code is known as an adapter.
- **Adapter Manager:** This form is used to compile multiple adapters simultaneously.
- **Form Designer:** This form is used to create process and resource object forms that do not come packaged with Oracle Identity Manager.
- **Error Message Definition:** This form is used to create error messages that can be used for reporting when certain problems occur while using Oracle Identity Manager. This form also enables a system administrator or developer to define the error messages that users can access when they create error handler tasks by using the Adapter Factory form.
- **Reconciliation Rules:** This form is used to create and manage reconciliation rules in Oracle Identity Manager.

1.6.5.1 Business Rule Definition

The Development Tools folder consists of the Business Rules Definition subfolder. The Business Rule Definition folder provides system administrators and developers with tools to manage the event handlers and data objects of Oracle Identity Manager. This folder contains the following forms:

- **Event Handler Manager:** This form is used to create and manage the event handlers that are used with Oracle Identity Manager.
- **Data Object Manager:** This form is used to define a data object, assign event handlers and adapters to it, and map any adapter variables associated with it.

See Also: [Chapter 5, "Developing Rules"](#) for more information about the forms in the Business Rule Definition folder

Developing Adapters

Adapters are Java programs that enable you to integrate Oracle Identity Manager with other software solutions. This chapter describes how to create adapters using the Adapter Factory form. It contains these sections:

- [Introduction to Adapters](#)
- [Types of Adapters](#)
- [Adapter Environment and Tools](#)
- [Defining Adapters](#)
- [Tabs of the Adapter Factory Form](#)
- [Disabling and Re-enabling Adapters](#)
- [About Adapter Variables](#)
- [Creating Adapter Tasks](#)
- [Modifying Adapter Tasks](#)
- [Changing the Order and Nesting of Tasks](#)
- [Deleting Adapter Tasks](#)
- [Working with Responses](#)
- [Scheduling Rule Generators and Entity Adapters](#)

2.1 Introduction to Adapters

To be effective, it must be possible to integrate an access rights management application, such as Oracle Identity Manager, with other software solutions. This is necessary not only because there are many resources, but also because there is no single integration standard for connecting to these resources.

The traditional way to tackle this challenge is by using the common functionality that is supported by all the integrations. To do this, you need developers who can write this code. In addition, every time an existing software resource is modified, or a new one is added, you must write more code.

The Adapter Factory is a code-generation tool provided by Oracle Identity Manager. It helps you create Java classes, known as adapters, that simplify the integration challenge.

Note: Oracle Identity Manager can connect to external systems such as databases and directory servers by using Java APIs for JDBC and LDAP. In addition, for all other APIs, such as C, C++, VB, and COM/DCOM, you can create a Java wrapper so that Oracle Identity Manager can communicate with the API directly.

A resource has an associated provisioning process, which in turn has various tasks associated with it. Each task in turn has an adapter associated to it, which in turn can connect to the target resource to carry out the required operations.

An adapter provides the following benefits:

- It extends the internal logic and functionality of Oracle Identity Manager.
- It interfaces with any software resource, by connecting to that resource by using the API of the resource.
- It enables the integration between Oracle Identity Manager and an external system.
- It can be generated without manually writing code. However, Oracle Identity Manager does not restrict you from writing your own code for creating adapters.
- It is lightweight and specific to your needs.
- It can be maintained easily because all of the definitions for the adapter are stored in a repository. This repository can be edited through a GUI.
- One Oracle Identity Manager user can retain the domain knowledge about the integration, while another user can maintain the adapter.
- It can be modified and upgraded efficiently.

Adapters can be developed for a range of tasks:

- A process task adapter, which allows Oracle Identity Manager to automate the completion of a process task.
- A task assignment adapter, which enables Oracle Identity Manager to automate the assignment of a process task to a user or group.
- A rule generator, which incorporates business rules to the fields of either an Oracle Identity Manager form or a user-defined form (created by using the Form Designer form), so these fields can be populated automatically and saved to the Oracle Identity Manager database.

Note: For more information about the Form Designer form, see ["Form Designer Form"](#) on page 13-1.

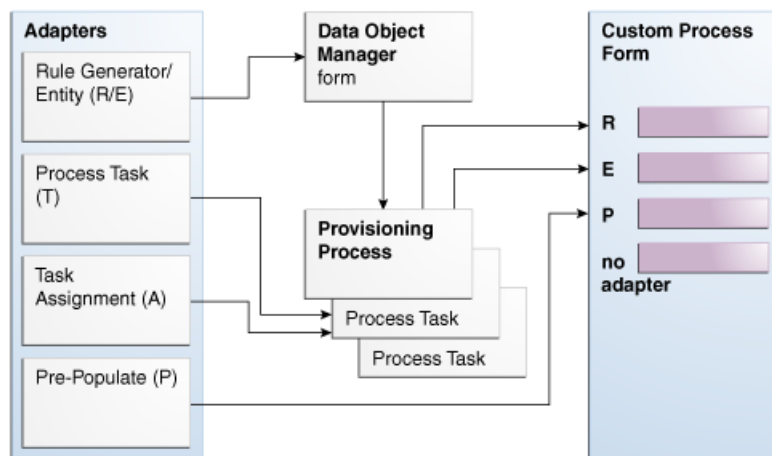
- A pre-populate adapter, which is a specific type of rule generator adapter that can be attached to a user-created form field. The data generated by this type of adapter can appear either automatically or manually. In addition, it uses criteria that enable Oracle Identity Manager to determine which pre-populate adapter will be applied to the designated form field. It populates the designated form field without saving this information to the Oracle Identity Manager database.
- An entity adapter, which is attached to an Oracle Identity Manager or user-created form field. Oracle Identity Manager triggers an entity adapter on preinsert, preupdate, predelete, postinsert, postupdate, or postdelete. After this occurs, the

field to which the adapter is attached is populated automatically and saved to the Oracle Identity Manager database.

Note: Oracle Identity Manager 11g Release 1 (11.1.1) also allows you to create postprocessing handlers on entities, such as user, role, and organization.

Figure 2–1 illustrates the functionality of five different types of adapters.

Figure 2–1 Adapter Functionality



2.2 Types of Adapters

This section provides additional details about the five adapter types.

Rule Generator Adapters

Certain business rules must be applied to perform field validations and enter default values into the forms which either come packaged with Oracle Identity Manager or are created by Oracle Identity Manager users. For example, for the Users form, you might want Oracle Identity Manager to generate the User ID automatically by concatenating the user's first name and last name.

To do this, you must create a specific type of adapter, which is designed to modify the field value in a form. This type of adapter, which can generate, modify, or verify the value of a form field automatically, is called a rule generator. Oracle Identity Manager triggers a rule generator on preinsert and preupdate.

After you create this adapter and attach it to a form, Oracle Identity Manager automatically updates the field value for all records of that form, and saves this information to the Oracle Identity Manager database.

If you create a rule generator that contains adapter variables, you must map these adapter variables to their proper locations. Otherwise, the adapter will not be functional.

You can also attach this type of adapter to a provisioning process. Once the process is provisioned to a target user or organization, Oracle Identity Manager will trigger the associated rule generator.

On occasion, a rule generator which has been assigned to a provisioning process might no longer be needed to complete the process. If this happens, you can remove the rule generator from the provisioning process. Similarly, after you attach one rule generator to a form field, you can connect a different rule generator to that form field. When this occurs, you must first remove the rule generator currently attached to the form field.

Entity Adapters

Similar to rule generator adapters, entity adapters are also responsible for generating, modifying, or verifying the value of a form field automatically, and saving this information to the Oracle Identity Manager database.

Some differences between rule generators and entity adapters are:

- **Execution schedule.** Entity adapters can be triggered by Oracle Identity Manager on preinsert, preupdate, predelete, postinsert, postupdate, and postdelete. A rule generator adapter can be executed only on preinsert and preupdate.
- **Manual field value modification.** The adapter populates the form field to which an entity adapter is attached. An Oracle Identity Manager user should not edit this value because the entity adapter will overwrite this modification. As a result, the modification will not be saved to the database.

Similarly, the adapter also populates the form field to which a rule generator adapter is attached. However, an Oracle Identity Manager user can edit this value because this modification will take precedence over the value that the rule generator adapter generates. Because of this, the modification will be saved to the database.

- **Background color of form field.** If a rule generator is attached to a form field, the field will appear in a particular background color such as pink. This is a visual indicator that the field has a rule generator attached to it. On the other hand, when an entity adapter is attached to a form field, the field will not have a distinct background color.

Task Assignment Adapters

For a process task that must be completed manually, you can configure Oracle Identity Manager to automate the assignment of the task to either a specific user or a user who belongs to a particular role. This is achieved through the use of a task assignment adapter. Task assignment adapters are used only for assigning a task to a particular user or role.

When a task that is associated with specific provisioning process is created using the Tasks tab in the Process Definition form of the Design Console, you can choose the rule that decides if adapter will be picked up for execution. Note that this rule is defined in the Rule Definition form of the Design Console. An example of a rule is "Target User's Org name is XYZ. If this rule is satisfied, then the corresponding task assignment is picked up. However, you can have multiple rules defined and used while deciding task assignment. For multiple rules, Oracle Identity Manager associates priority with the task assignment functionality to decide the order in which the rule determination must occur. When the rule is determined, corresponding task assignment is run.

Note: In other words, the task assignment rule allows Oracle Identity Manager to decide whether to assign a process task to a user or role. The task assignment adapter enables Oracle Identity Manager to determine which user or role will be the recipient of the process task.

For this example, Oracle Identity Manager will trigger the Associate Adapter with User rule first (because it has the highest priority). If the condition of this rule is TRUE, it is successful. As a result, Oracle Identity Manager will associate the related task assignment adapter (the Assign Task to User adapter) with the process task.

On the other hand, when the condition of a rule is FALSE, the rule has failed. Oracle Identity Manager triggers the rule with the next highest priority. If this rule is successful, then Oracle Identity Manager assigns the designated adapter to the target process task.

So, in this example, if the Associate Adapter with User rule fails, then Oracle Identity Manager triggers the Associate Adapter with Role rule. If this rule is successful, then Oracle Identity Manager associates the related task assignment adapter (the Assign Task to Role adapter) to the process task.

After assigning a rule to a task assignment adapter, if this type of adapter contains adapter variables, you must map these variables to their proper locations. Otherwise, the adapter will not be functional.

Finally, when a task assignment adapter becomes invalid, or is no longer necessary for Oracle Identity Manager to allocate the process task to a user or group, you must remove the adapter from the task.

Prepopulate Adapters

Sometimes a user-created form contains both fields that can be populated by Oracle Identity Manager and fields into which an Oracle Identity Manager user must enter data. When the information that the user types into a field is contingent upon the data that appears in a system-generated field, Oracle Identity Manager must first populate this field. When the form is displayed, the user can view the system-generated data to enter information into the appropriate fields.

This is achieved by creating a type of rule generator known as a prepopulate adapter. By attaching it to a field designated to be system-generated, you enable Oracle Identity Manager to automatically populate this field with the appropriate information, without saving this information to the Oracle Identity Manager database.

The data generated by a prepopulate adapter can appear automatically or it can be manually entered. Oracle Identity Manager displays this information automatically when the Auto-prepopulate check box is selected for a provisioning process. When this check box is cleared, an Oracle Identity Manager user must manually generate the displaying of the data that is generated by the prepopulate adapter. To do this, click the prepopulate button on the form section of the Direct Provisioning wizard in the Web client, while provisioning the form to a user.

You can use the same prepopulate adapter for different form fields. In addition, you can designate multiple prepopulate adapters to be associated with a particular field. As a result, Oracle Identity Manager must know which prepopulate adapter it must select for the form field. This requires the use of prepopulate rules. These rules enable Oracle Identity Manager to select one prepopulate adapter, which is associated with a form field, when this prepopulate adapter is assigned to the field.

Each prepopulate adapter has a prepopulate rule associated with it. In addition every rule has a priority number which indicates the order in which Oracle Identity Manager triggers it.

For example, Oracle Identity Manager can trigger the Rule for Uppercase User ID rule first because it has the highest priority. If the condition of this rule is TRUE, it is successful. As a result, Oracle Identity Manager will attach the related prepopulate adapter (the Display Uppercase Letters for User ID adapter) to the User ID field.

On the other hand, when the condition of a rule is FALSE, the rule has failed. Oracle Identity Manager will trigger the rule with the next highest priority. If this rule is successful, Oracle Identity Manager will attach the associated adapter to the designated field.

So, in this example, if the Rule for Uppercase User ID rule fails, Oracle Identity Manager will trigger the Rule for Lowercase User ID rule. If this rule is successful, Oracle Identity Manager will attach the related prepopulate adapter (the Display Lowercase Letters for User ID adapter) to the User ID field.

After assigning a rule to a prepopulate adapter, if this type of adapter contains adapter variables, you must map these adapter variables to their proper locations. Otherwise, the adapter will not be functional.

Finally, when a prepopulate adapter associated with a field is no longer valid, you must remove the adapter from the field.

Process Task Adapters

A process task adapter enables Oracle Identity Manager to automatically execute process tasks in provisioning processes.

Each process and process task has a status, which indicates the stage of its completion. The statuses for a process or process task are listed in the following table in order of importance.

Task Status	Description
C	Completed: This process/process task has been completed successfully.
MC	Manually Completed: This process task has been completed successfully by an Oracle Identity Manager user (that is, manually).
P	Pending: This process/process task is in the process of being completed. All preceding tasks and processes, respectively, have been completed.
PX	Pending Cancellation: This process task will be canceled, but this task has to be completed first before it can be canceled.
R	Rejected: This process/process task has not been completed successfully or has not been approved. The status of rejected process tasks can only be changed to <i>Canceled</i> or <i>Unsuccessfully Completed</i> .
S	Suspended: This process/process task has been put on hold temporarily.
UC	Unsuccessfully Completed: This process task has been set to <i>Completed</i> . However, it had been rejected before.
W	Waiting: This process/process task cannot be completed until all preceding process tasks or processes are completed.

Task Status	Description
X	This process/process task has been stopped. Its status cannot change anymore

The status level of a process represents the most important status level of its process tasks, which must be completed for the process to be completed. Suppose a process has three process tasks, each process task has a different status level (*Completed*, *Waiting*, and *Rejected*), and all three process tasks must be completed for the process to complete. Because the highest task status level is *Rejected*, the status level of the process is also *Rejected*.

A process task can be managed in these ways:

- It can be handled manually by using the Object Process Console tab of the Organizations or Users forms, or the Oracle Identity Manager Web Application.
- An Oracle Identity Manager process can be configured so that one (or more) of its tasks is triggered automatically once it achieves a status of *Pending*.

2.3 Adapter Environment and Tools

This section contains these topics:

- [Configuring the Adapter Environment](#)
- [Remote Manager](#)
- [The Adapter Factory](#)
- [Compiling Adapters](#)

2.3.1 Configuring the Adapter Environment

To construct adapter tasks, ensure that Oracle Identity Manager has access to the target API JAR files and third-party applications to which you want to connect.

When your adapter uses Java tasks, you must configure Oracle Identity Manager to find the appropriate Java APIs. To do this, you must place the .jar files that contain these APIs into the Meta Data Store (MDS).

See Also: [Chapter 33, "MDS Utilities and User Modifiable Metadata Files"](#) for information about utilities to modify Oracle Identity Manager metadata

Then, you can access the Java classes associated with these Java APIs and use them in the Java task you are creating.

To configure Oracle Identity Manager to reference JAR and class files:

1. Open the JavaTasks subdirectory, which can be found within the *OIM_HOME*/ directory path. For example, C:\oracle\Xellerate\JavaTasks.
2. Place the JAR file or files into this subdirectory. You can use these files to create Java tasks within an adapter without restarting the server.

2.3.2 Remote Manager

Sometimes, instead of directly communicating with the third-party system, Oracle Identity Manager must use an Oracle Identity Manager component that acts like a proxy. This component is known as Remote Manager.

The Remote Manager is used for:

- Invoking nonremotable APIs through Oracle Identity Manager
- Invoking APIs that do not support Secure Sockets Layer (SSL) over secure connections

To configure the Remote Manager, follow the instructions described in *Oracle Fusion Middleware Installation Guide for Oracle Identity Management*.

2.3.3 The Adapter Factory

As stated earlier, an adapter is a Java class created by an Oracle Identity Manager user through the Adapter Factory, which is accessed through the Design Console.

Adapters extend the internal logic and functionality of Oracle Identity Manager. In addition, they interact with any IT resource by connecting to that resource's API.

The Adapter Factory is a code-generation tool provided by Oracle Identity Manager that enables a user to create Java classes, known as adapters. [Figure 2–2](#) shows the Adapter Factory Form in the Design Console.

Figure 2–2 Adapter Factory Form

2.3.4 Compiling Adapters

Oracle Identity Manager provides various options for compilation, including:

- compile individual adapters one at a time
- compile a set of adapters at once
- compile all adapters that exist in the Oracle Identity Manager database with a single click

2.3.4.1 Automatic Compilation of Adapters

Adapters are compiled automatically when you import connector files by using the Deployment Manager. The compiled adapter class files are stored in the Oracle Identity Manager database, as opposed to the file system, from where they are loaded

at run time. The following two APIs are available to compile adapters programmatically:

- `public void compileAdapter (String adapterName)`: This API compiles a single adapter and stores the compiled classfile in the database. It takes the name of the adapter as a parameter. If the adapter is not found or if there are any errors, the API throws an appropriate exception.
- `public void compileAll`: This API compiles all adapters in a system. If it encounters any errors during compilation, it throws an exception of the type `tcBulkException`. This exception comprises all the individual errors that the API encounters during compilation.

You can modify the adapters manually if you make any changes.

Note: You must set the path of the JDK directory in the `XL.CompilerPath` system property. Otherwise, an error is encountered during the adapter compilation stage when you import an XML file using the Deployment Manager.

Refer to the "System Properties in Oracle Identity Manager" in the *Oracle Fusion Middleware System Administrator's Guide for Oracle Identity Manager* for information about setting values of system properties.

2.3.4.2 Compiling Adapters Manually

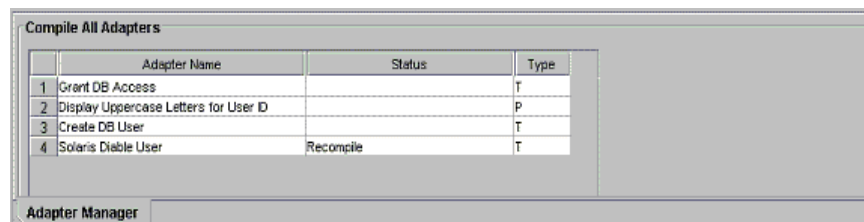
The Adapter Manager form is located in the Development Tools folder. You use it to compile multiple adapters simultaneously.

To manually compile multiple adapters, perform these steps:

1. Open the Adapter Manager form.

The Adapter Manager form is in the Development Tools folder. It is used to compile multiple adapters simultaneously, as shown in [Figure 2-3](#).

Figure 2-3 Adapter Manager Form



2. To compile every adapter that resides within the Oracle Identity Manager database, select the **Compile All** option.

To compile multiple adapters, select the adapters you want to compile. Then, select the **Compile Selected** option.

To compile all adapters that do not have an OK status, select the **Compile Previously Failed** option.

3. Click the **Start** button.

Oracle Identity Manager will compile the adapters that match the criteria you specified in Step 2.

Tip: Oracle Identity Manager lets you review the record of any adapter that appears within the Adapter Manager form to see detailed information about the adapter.

To view an adapter's record, select the desired adapter and either double-click its row header, or right-click the adapter, and select the Launch Adapter command from the menu that appears.

2.4 Defining Adapters

To define an adapter:

1. Log in to Oracle Identity Manager Design Console.
2. Open the Adapter Factory form. This form is in the Development Tools folder in the Design Console.
3. In the Adapter Name field, enter the name of the adapter, for example, `Create Solaris User`.

Note: Although the adapter name can contain special characters, Oracle recommends that you do not use them because there might be run-time errors.

4. Double-click the Adapter Type lookup field.

The Lookup window is displayed, displaying the five types of Oracle Identity Manager adapters. These are:

- Process Task
 - Rule Generator
 - Pre-populate Rule Generator
 - Entity
 - Task Assignment
5. To enable the adapter to automate a process task, select **Process Task (T)**.

To incorporate business rules into an Oracle Identity Manager or user-defined form field, select **Rule Generator (R)**. For example, for the User ID field of a form, you can configure Oracle Identity Manager to concatenate the initial letter of the user's first name with the user's last name.

You can attach a type of rule generator adapter to a user-created form field, so that it can:

- Display the data, which is generated by the adapter, automatically or manually.
- Use criteria that enable Oracle Identity Manager to determine which adapter is applied to the designated form field.

To attach the adapter to an Oracle Identity Manager or user-defined form field, and have Oracle Identity Manager trigger the adapter on preinsert, preupdate, predelete, postinsert, postupdate, or postdelete, select **Entity (E)**.

To allow the adapter to automate the allocation of a process task to a user or group, select **Task Assignment (A)**.

Tip: If you create an entity adapter, then an error might be generated while compiling the adapter on computers with less file limits. To avoid this problem, change the file limits in the `/etc/security/limits.conf` file to the following:

```
soft nofile 4096
```

```
hard nofile 4096
```

Then, restart Oracle Identity Manager.

6. Select the type of adapter you want, for example, Process Task (T). Then, click **OK**.

See Also: "[Form Designer Form](#)" on page 13-1 for more information about the Form Designer form

7. In the Description field, type a description for the adapter, for example, This adapter is used to create a new user for the Solaris environment.

8. From the toolbar, click **Save**.

The adapter is now stored in the Oracle Identity Manager database.

2.5 Tabs of the Adapter Factory Form

The Adapter Factory form in the Design Console contains the following tabs:

- [Adapter Tasks](#)
- [Execution Schedule](#)
- [Resources](#)
- [Variable List](#)
- [Usage Lookup](#)
- [Responses](#)

2.5.1 Adapter Tasks

In the Adapter Tasks tab, you can create and manage the atomic function calls of an adapter. These function calls are known as adapter tasks.

The sequence of calls is vital because these calls in turn gets converted into Java statements. In other words, if you put an Else call before an If call, then the adapter is not compiled. In addition, you must understand the logical flow of java program while creating adapter. Analogically, this is like writing an algorithm instead of a program with Java syntax.

2.5.2 Execution Schedule

The Execution Schedule tab lets you specify when you want Oracle Identity Manager to trigger a rule generator or an entity adapter. You can schedule Oracle Identity Manager to run a rule generator (Adapter Type *R*) on preinsert and/or preupdate. You can also configure Oracle Identity Manager to execute an entity adapter (Adapter Type *E*) on preinsert, preupdate, predelete, postinsert, postupdate, or postdelete.

Caution: Process task adapters and task assignment adapters, which are attached to process tasks, are triggered once the process task's status becomes *Pending*. Therefore, you do not specify when Oracle Identity Manager will trigger these types of adapters, Oracle Identity Manager disables the Execution Schedule tab for them.

Also, because Oracle Identity Manager always triggers pre-populate adapters on preinsert, Oracle Identity Manager disables the check boxes of this tab for pre-populate adapters.

2.5.3 Resources

From the Resources tab, you can:

- Click the Java APIs subtab to see the Java APIs that are being used by the adapter.
- Click the Other subtab to document a non-Java API file to the adapter, if necessary.

Note: This Resources tab does not represent resource objects.

2.5.4 Variable List

For prepopulation adapters, the data is passed to adapter input variables and are processed by using adapter logic. The adapter returns output variable, which is then assigned to process form field.

From the Variable List tab, you can:

- Create, modify, and delete adapter variables.
- Set the data type and provide a description for each variable.
- Map an adapter variable to a literal or an adapter reference. You can also postpone the mapping until it is attached to a process task or a form field.

You also can resolve the value of the adapter variable at run time, when it is attached to a process task and the process task is run. As a result, process-specific data is available to map to this variable.

2.5.5 Usage Lookup

For a process task or task assignment adapter, the Usage Lookup tab displays the process task to which the adapter is attached, as well as the process of which this process task is a member.

For a rule generator or entity adapter, this tab shows the Oracle Identity Manager form and associated data object to which the adapter is attached. In addition, it displays the execution schedule of the adapter, along with a sequence number that represents the order in which Oracle Identity Manager will trigger the adapter.

For a pre-populate adapter, this tab displays the user-defined form and form field to which the adapter is attached. Also, it shows the pre-populate rule that is associated with the adapter.

2.5.6 Responses

The Responses tab is used for defining meaningful responses to the process task. These responses depend on the execution result of the adapter. The various error messages returned by the external system can be mapped to these responses in a way that they make sense in the context of the process task. On attaching the adapter to a process task, the status bucket, which consists of Pending, Completed, and Rejected, of the process task (and subsequently the Object status) can be set, based on the adapter response code.

Tip: Oracle Identity Manager enables the Responses tab only for process task adapters. If an adapter is a task assignment, rule generator, pre-populate, or entity adapter, Oracle Identity Manager disables this tab.

2.6 Disabling and Re-enabling Adapters

To disable an adapter so that it cannot be used with a process task or form field, select the **Disable Adapter** option, and save the adapter.

To re-enable it, clear the **Disable Adapter** option, and save the adapter.

2.7 About Adapter Variables

For a newly-created adapter to work, you can map data to the parameters of the adapter tasks. For this reason, you create placeholders, also known as adapter variables, to map the data at run time.

Note: An adapter variable can be reused for all adapter tasks.

Once an adapter variable is not needed for the adapter to run, you can remove it from the adapter. After you have deleted the adapter variable, ensure to recompile the adapter.

2.7.1 Creating an Adapter Variable

To create an adapter variable:

1. Select the adapter to which you wish to add an adapter variable, for example, the `Create Solaris User` adapter.
2. Select the Variable List tab.
3. Click **Add**.
The Add a Variable window is displayed.
4. When you do not want Oracle Identity Manager to be able to change the adapter variable value after it is activated, select **Final**.
5. In the Variable Name field, enter the name of the adapter variable, for example, `SolarisUserID`.

Caution: The adapter variable name cannot contain spaces.

6. From the Type menu, select the classification type of the adapter variable, such as String. The available items are:
 - Object
 - IT Resource
 - String
 - Boolean
 - Character
 - Byte
 - Date
 - Integer
 - Float
 - Long
 - Short
 - Double
7. Within the Description text area, you can enter explanatory information about the adapter variable.
8. From the Map To menu, you can map your adapter variable to one of the items listed in [Table 2-1](#).

Table 2-1 *Items on the Map To Menu*

Name	Description
Literal	This adapter variable is mapped to a constant (or literal).
Resolve at Run time	This adapter variable's mapping occurs later, at run time. Selecting this option increases the reusability of the adapter.
Adapter References	This adapter variable gives access to an Oracle Identity Manager database reference or an Oracle Identity Manager data object reference.
System Date	When this adapter variable is triggered by Oracle Identity Manager, it is mapped to the current date and time of the Server. Note: This option appears only when you select the Date type.

Note: When you select the object type, a Qualifier menu is displayed within the Add a Variable window. From this menu, you can select either of the following:

- Database Reference. If you select this item, the adapter variable is mapped to the reference of the database that the Oracle Identity Manager is currently running against.
 - Data Object Reference. If you select this item, the adapter variable is mapped to an Oracle Identity Manager data object.
-

Note: If you select the IT Resource type, a Resource Type menu is displayed within the Add a Variable window. From this menu, you can select one of the IT resource types that have been created by using the IT Resource Type Definition form. By doing so, you can map the adapter variable to a parameter of this IT resource type.

9. On the toolbar in the Add a Variable window, click **Save**. The information for your adapter variable is stored in the Oracle Identity Manager database.

Close the Add a Variable window to activate the main screen. The name, classification type, mapping selection, and description of the adapter variable you created appear in the child table of the Variable List tab.

This adapter variable now belongs to the adapter in the Adapter Factory form. It is saved to the Oracle Identity Manager database, and the adapter variable is ready to use.

2.7.2 Modifying an Adapter Variable

To modify an adapter variable:

1. Select the adapter that contains the adapter variable you want to edit, for example, the `Create Solaris User` adapter.
2. Click the Variable List tab and double-click the row header of the adapter variable you want to modify. The Edit a Variable window is displayed, showing information about the adapter variable.
3. Make the necessary edits, for example, changing the adapter variable's data type from String to Character.
4. On the Edit a Variable toolbar, click **Save**. The modified information about the adapter variable is stored in the Oracle Identity Manager database.
5. Close the Edit a Variable window to activate the main screen. The adapter variable you modified appears within the child table of the Adapter Factory form.

Note: Ensure that you check your data mappings and recompile the adapter, especially if you change the adapter variable's data type.

2.7.3 Deleting an Adapter Variable

When an adapter variable is no longer necessary for the adapter to run, you can remove it from the adapter. To do this:

1. Select the adapter that contains an adapter variable you want to remove, for example, the `Create Solaris User` adapter.
2. Select the Variable List tab.
3. From the list of this tab, select the adapter variable you want to delete.
4. Click **Delete**.
5. Recompile the adapter after deleting any variable.

The adapter variable disappears from the child table. The adapter variable has been deleted.

2.8 Creating Adapter Tasks

After you construct the adapter and create its variables, you can create the atomic function calls of an adapter. These function calls are known as adapter tasks.

This section explains adapter tasks and how to create tasks:

- [Types of Adapter Tasks](#)
- [Creating a Java Task](#)
- [Creating a Remote Task](#)
- [Creating a Stored Procedure Task](#)
- [Creating a Utility Task](#)
- [To Create an Oracle Identity Manager API Task](#)
- [Reassigning the Value of an Adapter Variable](#)
- [Adding an Error Handler Task](#)
- [Creating a Logic Task](#)

2.8.1 Types of Adapter Tasks

Oracle Identity Manager allows you to create the following adapter tasks:

- A Java task, which allows an adapter to communicate with an external source by invoking Java API.
- A remote task, which enables an adapter to call a method on an API. This API resides on a computer that is external to Oracle Identity Manager.

This type of task is used mostly with integrations of third-party APIs that are not network-enabled. A remote manager executes the remote API method, which is located on a remote computer. In addition, if the third-party API does not use SSL, you can use the remote manager to invoke third-party APIs over SSL-protected communication. Remote tasks can also be used with integrations of third-party APIs, which are network-enabled, but are not located on the Oracle Identity Manager server for scalability purposes. The remote API method is still executed by a remote manager. However, because the third-party API is network-enabled, the remote manager does not have to reside on the target system.

- A stored procedure task, which allows Oracle Identity Manager to map to and execute SQL programs located within a particular database schema. These programs are known as stored procedures. They contain information, such as SQL statements, which are pre-compiled for greater efficiency.

By incorporating a stored procedure task into an adapter, and attaching this adapter to a process task, Oracle Identity Manager can incorporate stored procedures on any Oracle Database or Microsoft SQL Server database that is accessible on its network. This includes retrieving primitive values from stored procedures.

- A utility task, which enables you to populate an adapter with methods and APIs that come packaged with Oracle Identity Manager. In addition, this type of task provides you with access to the Java Standard Library APIs.
- An Oracle Identity Manager API task, which enables access to Oracle Identity Manager published APIs from adapter tasks. This allows for enhanced portability of adapter code.

- A set variable task, which allows you to set a variable within an adapter.
- An error handler task, which lets you display any errors associated with an adapter that occur at run time. In addition, you can see the reasons for the errors, along with possible solutions.
- A logic task, which lets you build a conditional statement within an adapter.

You can create the following types of logic tasks:

- FOR loops
- WHILE loops
- IF statements
- ELSE statements
- ELSE IF statements
- BREAK statements
- RETURN statements
- CONTINUE statements
- SET VARIABLE statements
- Handle Error statements

See Also: [Section 2.8.9, "Creating a Logic Task"](#) for more information about the types of logic tasks you can build

For classification purposes, Oracle Identity Manager represents each type of adapter task by an icon. The icon, which precedes the task name, is a visual indicator of the type of task it is. For example, "J" represents a Java task, and "LT" represents a logic task.

To see a list of these icons, select the Adapter Tasks tab, and click **Legend**. The Legend window appears, displaying the following list of icons:

- Functional Task
 - Java
 - Remote
 - Stored Procedure
- Utility Task
 - Utility
 - Oracle Identity Manager API
- Logical Task

2.8.2 Creating a Java Task

Oracle Identity Manager can handshake with an external source through a Java API. To make this happen, you must add a task to an adapter which, when triggered by Oracle Identity Manager, initiates communications with the external source. This type of task is called a Java task.

To create a Java task:

1. Select the adapter to which you want to add a Java task, for example, the Update Solaris Password adapter.
2. Select the Adapter Tasks tab.
3. Click **Add**.
After the Adapter Task Selection window is displayed, select the **Functional Task** option.
4. From the display area to the right of this option, select the Java item, and click **Continue**.

The Object Instance Selection window is displayed.

[Table 2–2](#) explains the options in the Object Instance Selection window.

Table 2–2 Options in the Object Instance Selection Window

Option	Description
New Object Instance	When you click this option, you are creating a new Java object instance.
Persistent Instance	You can call the method on a persistent object by clicking this option, clicking the adjacent combo box, and selecting an object instance from the drop-down menu.
Task Return Value Instance	You can call this method on an object returned by an adapter task defined earlier by clicking this option, clicking the combo box, and selecting an adapter task from the drop down list.

Note: When the Persistent Instance option is grayed out, it indicates that you have not defined any persistent objects for your adapter. Similarly, if the Task Return Value Instance option is grayed out, none of the tasks have Java Object return values associated with them.

5. Click an option—for example, New Object Instance—and click **Continue**. The Add an Adapter Factory Task window is displayed.

[Table 2–3](#) lists and describes the various regions of the Add an Adapter Factory Task window:

Table 2–3 Regions of the Add an Adapter Factory Task Window

Name	Description
Task Name	This field displays the name of the Java task.
Persistent Instance	If this Java object is to be used again, the check box is selected, and the name of the task instance is entered in the adjacent field.
API Source	This combo box contains a list of all JAR and class files to which you have access.
Application API	This combo box contains a list of all class files to which you have access, and which belong to the JAR file that has been selected from the API Source list.
Constructors	This text area displays all the constructors, which are available for the Java object.
Methods	This text area shows a list of all the methods, which are available for the Java object.

Table 2–3 (Cont.) Regions of the Add an Adapter Factory Task Window

Name	Description
Application Method Parameters	This area contains the parameters of the selected constructor and method. These parameters are mapped to the adapter variables and Oracle Identity Manager components.

6. In the Task Name field, enter the name of the task you are creating, for example, Update Password.
7. (Optional.) To make your Java object reusable, select **Persistent Instance**, type the name of the instance of this task in the text field located to the right of the check box.

Caution: Ensure that name of the instance contains no spaces.

Note: To reference a session with the target resource multiple times during the life of the adapter, and not just once, select **Persistent Instance**.

Tip: By setting the Java object to be persistent, the next time you create a Java object, it appears in the Persistent Instance list of the Object Instance Selection window. In addition, you do not have to map the constructor to all adapter tasks of the same Java object.

8. Select the API Source. The JAR files appear, which Oracle Identity Manager references from the JavaTasks subdirectory of the *OIM_HOME/* directory path—for example, C:\oracle\Xellerate\JavaTasks.

See Also: [Section 2.3.1, "Configuring the Adapter Environment"](#) for instructions on how to enable Oracle Identity Manager to use third-party JAR files with a Java task

9. Select the Application API. The class files, which belong to the JAR file you selected in the API Source, appear.
10. From the Constructors area, select the method to be used to initialize the Java class you selected.
11. From the Methods area, select the method that will be used with your Java task.
12. From the toolbar, click **Save**.

The information pertaining to the Java task is stored in the Oracle Identity Manager database. You can now access the parameters of your Java task's constructors and methods. These parameters appear in the Application Method Parameters region of the Add an Adapter Factory Task window.

13. To display the Java class constructors and methods for which you must set mappings, click the plus icons displayed to the left of the Constructor and Method icons.
14. Select the parameter of the constructor or method for which you must set a mapping.
15. In the Description text area, you can enter a description for this mapping.

16. Click the Map to combo box, and select an item that you can map to the parameter of the constructor or method, for example, Adapter Variables.
17. Set the appropriate mappings.

See Also: ["Adapter Mapping Information"](#) on page 3-18 for more information about which mappings to set

18. Click **Set**.

The parameter of the selected constructor or method now appears in blue. This signifies that it has been mapped.

Tip: To remove a parameter mapping, right-click the appropriate parameter, and select Un-Map Parameter from the popup menu that appears.

19. Repeat steps 15 through 18 for all parameters of the constructors and methods that appear in the Application Method Parameters region.
20. On the Add an Adapter Factory Task window toolbar, click **Save**. The information pertaining to the Java task is stored in the Oracle Identity Manager database.
21. On the toolbar, click **Close**. The Add an Adapter Factory Task window disappears, and the main screen is active once again. The Java task that you created—for example, Update Password—appears within the Adapter Factory form.
22. (Optional.) To create additional Java tasks for the adapter, repeat steps 3-21.

Tip: You can create different types of adapter tasks, and add them to the adapter.

If the adapter is logically complete, and all variables on the adapter tasks are mapped, you can compile it to use with a process task or form field.

23. To compile the adapter, click **Build**.

The text in the Compile Status field changes from Recompile to OK. This indicates that Oracle Identity Manager compiled the adapter and found no errors. You can now attach the adapter to a process task or form field.

24. (Optional.) To see the code that Oracle Identity Manager generates, from the toolbar, click **Notes**.

The Notes window is displayed, containing the code that Oracle Identity Manager generated.

Note: If, after clicking Build, CODE GEN ERROR appears in the Compile Status field, it means that Oracle Identity Manager encountered one of two types of errors while validating and compiling the adapter:

- Validation Error

While Oracle Identity Manager is checking the adapter to verify that it is valid, an error is found. This error can result from a parameter of an adapter task not being mapped, a parameter being mapped improperly, or an adapter task being placed out of order.

Because Oracle Identity Manager generates code for an adapter only after it is validated, if Oracle Identity Manager encounters a validation error, it does not create any code.

- Java Compilation Error

Oracle Identity Manager has verified that the adapter is valid. However, while Oracle Identity Manager is compiling the adapter, an error is found. This error can result from assigning an incorrect data type to an adapter task parameter.

Because Oracle Identity Manager has validated the adapter, it generates code. However, Oracle Identity Manager stops building code at the point of the compilation where it encounters the error.

Tip: Once you create a Java task, and add it to an adapter, you can see the following information by accessing the Resources tab of the Adapter Factory form:

- The JAR and class files used to create the Java task.
- The name, which represents the directory path that contains these JAR and class files.

2.8.3 Creating a Remote Task

A remote task enables an adapter to invoke an API method by using the Remote Manager. This API resides on a computer that is external to Oracle Identity Manager. This section explains how to create a remote task.

Note: Before creating a remote task, ensure that you define an adapter variable with a classification type of IT Resource, as well as select one of the IT resources that have been created by using the IT Resource Type Definition form.

1. Select the adapter to which you wish to add a remote task.
2. Click the Adapter Tasks tab.
3. Click **Add**.
The Adapter Task Selection window is displayed.
4. Select the Functional Task option.

5. From the display area to the right of the button, select the Remote item to create a remote task. Then, click **Continue**.

The Object Instance Selection window is displayed.

Note: To learn more about the choices of this window, refer to [Section 2.8.2, "Creating a Java Task"](#).

6. Click **Continue**.

The Add an Adapter Factory Task window is displayed.

7. In the Task Name field, enter the name of the remote task you are creating.
8. (Optional.) If you want your remote task to be reusable, select the **Persistent Instance** option. Then, type the name of the instance of this task in the text field, located to the right of the check box.

Caution: Ensure that the name of the instance contains no spaces.

See Also:

[Section 2.8.2, "Creating a Java Task"](#) for more information about the regions of the Add an Adapter Factory Task window

[Section 2.3.1, "Configuring the Adapter Environment"](#) for information about how to enable Oracle Identity Manager to use third-party JAR files with a Java task

9. From the Add an Adapter Factory Task window, select a JAR file, class file, constructor, and method. Then, set the mappings for the parameters of the constructor and method.

Note: One of the input parameters will have a classification type of IT Resource. You must associate this parameter with an adapter variable of type IT Resource.

See Also: ["Adapter Mapping Information"](#) on page 3-18 for more information about which mappings to select

10. From the Add an Adapter Factory Task window toolbar, click **Save**.

The information pertaining to the remote task is stored in the Oracle Identity Manager database.

11. From this window toolbar, click **Close**.

The Add an Adapter Factory Task window disappears, and the main screen is active once again. The remote task that you created appears within the Adapter Factory form.

12. (Optional.) To create additional remote tasks for the adapter, repeat Steps 3 through 11.

You are now ready to compile the adapter, so it can be used with a process task or form field.

13. To compile the adapter, click **Build**.

The text in the Compile Status field changes from Recompile to OK. This indicates that Oracle Identity Manager compiled the adapter and did not find any errors. You can now attach the adapter to a process task or form field, so Oracle Identity Manager can communicate with the external API.

2.8.4 Creating a Stored Procedure Task

Through Oracle Identity Manager, you can map to and execute SQL programs that are located within a particular database schema. These SQL programs are known as stored procedures. Stored procedures contain information, such as SQL statements, which are precompiled for greater efficiency.

For this to occur, you must add a stored procedure task to an adapter. When triggered by Oracle Identity Manager, this task incorporates stored procedures on any Oracle Database or Microsoft SQL Server database that is accessible on its network. This includes retrieving primitive values from stored procedures.

Take these steps to create a stored procedure task:

Note: The parameter values and server type for the database schema are set within the IT Resources form.

The server type of the schema must be set to Database. Otherwise, Oracle Identity Manager cannot reference the database schema during the creation of a stored procedure task, the execution of a stored procedure task, or both.

In addition, Oracle Identity Manager uses values, which are represented by parameters—for example, Database Name or *URL*—to connect to the schema. As a result, the stored procedures contained within the schema, can be executed.

1. For Oracle Identity Manager Installations that use Oracle Database, copy the `ojdbc14.jar` file from the `OIM_HOME/ext/` directory to the `OIM_DC_HOME/xlclient/ext` directory.

For Oracle Identity Manager Installations that use Microsoft SQL Server, you must obtain the following files from Microsoft and copy them to the `OIM_DC_HOME/xlclient/ext` directory:

- `msbase.jar`
- `mssqlserver.jar`
- `msutil.jar`

2. Select the adapter to which you wish to add a stored procedure task, for example, the Update User ID adapter.
3. Click the Adapter Tasks tab.
4. Click **Add**.

The Adapter Task Selection window is displayed.

5. Select the **Functional Task** option.
6. From the display area to the right of the option, select **Stored Procedure**, and click **Continue**. The Add an Adapter Factory Task window is displayed.

The following table lists and describes the regions of the Add an Adapter Factory Task window.

Table 2–4 Regions of the Add an Adapter Factory Task Window

Name	Description
Task Name	Displays the name of the stored procedure task.
Description	Displays explanatory information about the stored procedure task.
Database	Lists the databases defined in the IT Resources form. Important: Only those IT resources with a server type of Database appear in the Database list.
Schema	Lists the schemas, which are associated with the database that appears in the Database list.
Procedure	Lists the stored procedures, which reside within the database schema that is displayed in the Schema list.
Connection Status	<p>Displays the status of the connection between Oracle Identity Manager and the database that contains the target stored procedure.</p> <p>When Oracle Identity Manager can connect to the database, Connection Established is displayed in the Connection Status region.</p> <p>Note: If Oracle Identity Manager cannot connect, Connection Failed appears in the display area. In addition, the Notes button of the Add an Adapter Factory Task window is enabled. Clicking this button shows you why a connection could not be established, for example:</p> <pre data-bbox="769 1062 1341 1205"> Exception Type: java.lang.ClassNotFoundException: java.lang.ClassNotFoundException: oracle.jdbc.driver.OracleDriver </pre> <p>In this example, Oracle Identity Manager could not connect to the designated database because it could not find a particular Java class.</p>
Parameters	Contains parameters that can be mapped to the stored procedure. These parameters appear after you select a database, schema, and stored procedure and save this information to the Oracle Identity Manager database.

7. In the Task Name field, enter the name of the stored procedure task you are creating (for example, Update ID).
8. In the Description text area, you can enter a description for this stored procedure task.
9. Click the Database list. The databases, which are defined in the IT Resources form, appear.

Note: If Oracle Identity Manager cannot connect to the database you selected, *Connection Failed* appears in the display area. In addition, the **Notes** button of the Add an Adapter Factory Task window is enabled. Clicking this button shows you why a connection could not be established.

Tip: Schemas and stored procedures appear only after you select a database to which Oracle Identity Manager can connect. Based on this selection, related schemas and stored procedures appear in the corresponding combo boxes.

10. Click the **Schema list**. The schemas appear, which are associated with the database you selected.
11. Click the **Procedure list**. The stored procedures, which reside within the database schema that you selected from the Schema combo box, appear.
12. From the Add an Adapter Factory Task window's toolbar, click **Save**. The information pertaining to the stored procedure task is saved into the Oracle Identity Manager database.

You can now set the mappings for the parameter(s) of the stored procedure. These parameters appear in the Parameters region of the Add an Adapter Factory Task window.

Note: Oracle Identity Manager automatically maps the database and schema of the selected stored procedure. However, Oracle Identity Manager enables you to override these mappings.

See Also: "[Adapter Mapping Information](#)" on page 3-18 for more information about which mappings to select

13. From the Add an Adapter Factory Task window's toolbar, click **Save**. The mappings that you have set for the parameter(s) of the stored procedure task are stored in the Oracle Identity Manager database.
14. From this window's toolbar, click **Close**.
The Add an Adapter Factory Task window disappears, and the main screen is active once again. The stored procedure task you created (for example, Update ID) appears within the Adapter Factory form.
15. (Optional.) Repeat steps 3 through 13 to create additional stored procedure tasks for the adapter.
16. To compile the adapter, click **Build**.

The text in the Compile Status field changes from Recompile to OK. This indicates that Oracle Identity Manager compiled the adapter and did not find any errors. You can now attach the adapter to a process task or form field, so Oracle Identity Manager can map to and execute the stored procedure you selected.

2.8.5 Creating a Utility Task

The Adapter Factory is shipped with a library of utility classes and methods, which increase the efficiency of developing adapters.

These utility classes and methods are contained within the **xlUtils.jar**, **xlIntegration.jar**, and **rt.jar** files. A Java task you create by using a class or method from one of these JAR files is called a **utility task**.

See Also: *Oracle Fusion Middleware Java API Reference for Oracle Identity Manager* for more information about the class files that contains the xlUtils.jar, xlAPI.jar, xlIntegration.jar, and rt.jar files

1. Select the adapter to which you wish to add a utility task, for example, the Update Solaris User Group adapter.
2. Click the Adapter Tasks tab.
3. Click **Add**.
The Adapter Task Selection window is displayed.
4. Select the **Utility Task** option.
5. From the display area to the right of the option, select **Utility**, and click **Continue**. The Object Instance Selection window is displayed.

See Also: ["Creating a Java Task"](#) on page 2-17 to learn more about the choices of this window

6. Click **Continue**. The Add an Adapter Factory Task window is displayed
7. In the **Task Name** field, enter the name of the utility task you are creating, for example, Update User Group.
8. (Optional.) If you want your utility task to be reusable, select **Persistent Instance**, type the name of the instance of this task in the text field to the right of the check box.

Caution: Ensure that name of the instance does not contain any spaces.

See Also:

["Creating a Java Task"](#) on page 2-17 for more information about the regions of the Add an Adapter Factory Task window

["Configuring the Adapter Environment"](#) on page 2-7

9. Click the Application API list. The class files appear, which belong to the `xlUtils.jar`, `xlIntegration.jar`, and `rt.jar` files.

Note: The `xlUtils.jar`, `xlIntegration.jar`, and `rt.jar` files contain all of the class files that you can use for a utility task. Therefore, you do not have to access the API Source list.

10. From the Add an Adapter Factory Task window, select a constructor and method. Then, set the mappings for the parameters of the constructor and method.

See Also: ["Adapter Mapping Information"](#) on page 3-18 for more information about which mappings to select

11. From the Add an Adapter Factory Task window's toolbar, click **Save**. The information pertaining to the utility task is stored in the Oracle Identity Manager database.

12. From this window's toolbar, click **Close**.

The Add an Adapter Factory Task window disappears, and the main screen is active once again. The utility task that you created (for example, Update User Group) appears within the Adapter Factory form.

13. (Optional.) Repeat steps 3 through 12 to create additional utility tasks for the adapter.

You are now ready to compile the adapter, so it can be used with a process task or form field.

14. To compile the adapter, click **Build**.

The text in the Compile Status field changes from Recompile to OK. This indicates that Oracle Identity Manager compiled the adapter and did not find any errors. You can now attach the adapter to a process task or form field.

2.8.6 To Create an Oracle Identity Manager API Task

For greater portability of the adapter code, an Oracle Identity Manager API task enables Adapter tasks to call APIs published by Oracle Identity Manager. This is better than accessing Oracle Identity Manager data directly through hardcoded SQL statements.

The Adapter Factory is shipped with a library of utility classes and methods, which increase the efficiency of developing adapters that contain Oracle Identity Manager API tasks. These utility classes and methods are contained within the xlAPI.jar file.

See Also: *Oracle Fusion Middleware Java API Reference for Oracle Identity Manager* for more information about the class files that contain the xlUtils.jar, xlAPI.jar, xlIntegration.jar, and rt.jar files

To create this type of adapter task:

1. Select the adapter to which you wish to add an Oracle Identity Manager API task, for example, the Get User's Password adapter.
2. Click the Adapter Tasks tab.
3. Click **Add**.

The Adapter Task Selection window is displayed.

4. Select the **Utility Task** option.
5. From the display area to the right of the option, select Xellerate API, and click **Continue**. The Object Instance Selection window is displayed.

See Also: "[Creating a Java Task](#)" on page 2-17 to learn more about this window

6. Click **Continue**. The Add an Adapter Factory Task window is displayed.
7. In the Task Name field, enter the name of the Oracle Identity Manager API task you are creating, for example, Retrieve Password).
8. (Optional.) If you want your Oracle Identity Manager API task to be reusable, select Persistent Instance. Then, type the name of the instance of this task in the text field to the right of the check box.

Tip: Ensure that name of the instance contains no spaces.

See Also:

["Creating a Java Task"](#) on page 2-17 for more information about the regions of the Add an Adapter Factory Task window

["Configuring the Adapter Environment"](#) on page 2-7 to learn how to enable Oracle Identity Manager to use third-party JAR files with a Java task

9. Click the Application API list. The class files appear, which belong to the `x1API.jar` file.

Note: The `x1API.jar` file contains all of the class files that you can use for an Oracle Identity Manager API task. Therefore, you do not have to access the API Source list.

10. From the Add an Adapter Factory Task window, select a class file, constructor, and method. Then, set the mappings for the parameters of the constructor and method.

See Also: ["Adapter Mapping Information"](#) on page 3-18 for more information about which mappings to select

11. From the Add an Adapter Factory Task window's toolbar, click **Save**. The information pertaining to the Oracle Identity Manager API task is stored in the Oracle Identity Manager database.
12. Close the Add an Adapter Factory window to activate the main screen. The Oracle Identity Manager API task that you created—for example, *Retrieve Password*—appears within the Adapter Factory form.
13. (Optional.) To create additional Oracle Identity Manager API tasks for the adapter, repeat steps 3 through 12.

You are now ready to compile the adapter, so it can be used with a process task or form field.

14. To compile the adapter, click **Build**.

The text in the **Compile Status** field changes from Recompile to OK. This indicates that Oracle Identity Manager compiled the adapter and did not find any errors. You can now attach the adapter to a process task or form field, so Oracle Identity Manager can communicate with a third-party application.

2.8.7 Reassigning the Value of an Adapter Variable

Sometimes, for an adapter to accomplish its required objective, you must reassign the value of one adapter variable to another adapter variable, a different type of adapter task, or a constant (or literal). The task that enables you to reallocate an adapter variable value is known as a set variable task.

See Also: ["About Adapter Variables"](#) on page 2-13 for information about adapter variables

For example, you can create a set variable task to set the adapter variable return value to equal the output of an adapter task (UserName) if the User ID length is fewer than 11 characters.

To create a set variable task:

1. Select the adapter to which you wish to add a set variable task (for example, the Check the Solaris User ID adapter).
2. Click the Adapter Tasks tab.
3. Click **Add**. The Adapter Task Selection window is displayed.
4. Select the Logic Task option.
5. From the display area, select SET VARIABLE, and click **Continue**. The Add Set Variable Task Parameters window is displayed.
6. From the Variable Name list, select the adapter variable that has a value you want to reassign—for example, Adapter return value.
7. From the Operand Type list, select the type of operand that will provide the value for the variable.

Tip: You can reassign an adapter variable's value to another adapter variable, a different type of adapter task, or a literal.

Use [Table 2-5](#) to understand the various types of operands.

Table 2-5 *Types of Operands*

Operand Name	Description
Variable	<p>If you select this operand type, adapter variables appear in the Operand Qualifier list. From this list, select the specific adapter variable that will provide the reassigned value.</p> <p>Note: The only adapter variables that will appear in the Operand Qualifier combo box will be those variables that have the same data type as the adapter variable that is displayed within the Variable Name combo box.</p>
Adapter Task	<p>By selecting this operand type, adapter tasks are displayed in the Operand Qualifier combo box. From this combo box, select the particular adapter task that will provide the reallocated value.</p> <p>Note: The only adapter tasks that will appear in the Operand Qualifier combo box will be those tasks that have the same data type as the adapter variable that is displayed within the Variable Name combo box.</p>
Literal	<p>When you select this operand type, types of literals appear in the Operand Qualifier combo box. From this combo box, select the type of literal that will provide the reallocated value. Then, type the specific literal into the field that appears underneath the combo box.</p>

The following task sets the adapter variable's return value to be equal to the UserName adapter variable.

1. On the toolbar in the Add Set Variable Task Parameters window, click **Save**. The set variable task you created is stored in the Oracle Identity Manager database.
2. On the Add Set Variable Task Parameters window toolbar, click **Close**. The Add Set Variable Task Parameters window disappears, and the main screen is active once again. The set variable task that you created, for example, `Set Adapter return value = UserName`, appears in the Adapter Factory form.
3. (Optional.) Repeat Steps 3-9 to create additional set variable tasks for the adapter.

You are now ready to compile the adapter, so it can be used with a process task or form field.

4. To compile the adapter, click **Build**. The text in the Compile Status field changes from Recompile to OK. Oracle Identity Manager compiled the adapter and found no errors. You can attach the adapter to a process task or form field.

2.8.8 Adding an Error Handler Task

To add an error handler task:

1. An adapter task can return errors. When this occurs, the process task or form field to which the adapter is attached gets rejected.

You can attach your own customizable error messages, which will be displayed to the user. These messages are known as error handler tasks.

For example, you can attach an error handler task to an adapter that will display an error message when the length of a User ID is greater than 10 characters.

2. Select the adapter to which you wish to add an error handler task (for example, the *Check the Solaris User ID* adapter).
3. Click the Adapter Tasks tab.
4. Click **Add**.

The Adapter Task Selection window is displayed.

5. Select the **Logic Task** option.
6. From the display area, select Handle Error, and click **Continue**. The Add an Adapter Factory Task window is displayed.
7. Double-click this window's lookup field. The Lookup window is displayed, displaying the error handler tasks you can add to the adapter.

Note: The only error handler tasks that appear in this Lookup window are the ones that begin with ADAPTER—such as ADAPTER.USERIDLENERR).

If you do not see the error handler task that you want to incorporate into the adapter, you can create one by accessing the Error Message Definition form. Refer to "[Creating an Error Message](#)" on page 13-19.

8. Select the error handler task you want, for example, ADAPTER.USERIDLENERR.
9. Click **OK**. The Lookup window disappears, and the Add an Adapter Factory Task window is active. In addition, the error handler task you selected appears in the field of this window.
10. From the Add an Adapter Factory Task window toolbar, click **Save**. The error handler task you incorporated into the adapter is stored in the Oracle Identity Manager database.
11. From this window's toolbar, click **Close**.

The Add an Adapter Factory Task window disappears, and the main screen is active once again. The error handler task you added, for example, Handle Error.ADAPTER.USERIDLENERR, appears within the child table of the Adapter Factory form.

12. (Optional.) Repeat Steps 3-10 to create additional error handler tasks for the adapter.

If the adapter is logically complete and all variables on the adapter tasks are mapped, you can compile it to use with a process task or form field.

13. To compile the adapter, click **Build**.

The text in the Compile Status field changes from *Recompile* to *OK*. This indicates that Oracle Identity Manager compiled the adapter and did not find any errors. You can now attach the adapter to a process task or form field.

See Also: "Creating an Error Message" on page 13-19 for information about creating error messages by using the Error Message Definition form

2.8.9 Creating a Logic Task

While defining the adapter, you can add conditional statements to the adapter to control its logic flow. These conditional statements are known as logic tasks. For example, you can create a logic task that will trigger an action if the length of a User ID is greater than 10 characters.

To create a logic task:

1. Select the adapter to which you wish to add a logic task (for example, the Check the Solaris User ID adapter).
2. Click the Adapter Tasks tab.
3. Click **Add**. The Adapter Task Selection window is displayed.
4. Select the **Logic Task** option.
5. From the display area, select the type of logic task you want to create. Then, click **Continue**.

Note: If you select a conditional expression, and click **Continue**, one of the following actions occurs:

Oracle Identity Manager adds the conditional statement to the adapter directly; or

A secondary window is displayed, containing fields about the conditional expression that you can configure.

To see what happens when you select a particular conditional statement, refer to [Table 2-6](#).

Table 2-6 Actions Resulting from Particular Conditional Statements

Conditional Statement	Statement Is Added to the Adapter Directly	Secondary Window Appears
FOR		X
WHILE		X
IF		X
ELSE	X	
ELSE IF		X

Table 2–6 (Cont.) Actions Resulting from Particular Conditional Statements

Conditional Statement	Statement Is Added to the Adapter Directly	Secondary Window Appears
BREAK	X	
RETURN	X	
CONTINUE	X	

Table 2–7 explains the various regions of the Add Adapter Factory Logic Task Parameters window:

Table 2–7 Regions of the Add Adapter Factory Logic Task Window

Name	Description
Operand Type	These combo boxes contain types of operands, such as adapter tasks and adapter variables.
Comparator Combo Box	From this combo box, you can set the relationship between two operands (for example, <, =, >).
Operand Qualifier	These combo boxes contain the qualifiers for the operands.
Literal Text Box	When you select the Literal operand type, enter the specific literal into this field.

Note: By selecting the FOR conditional statement, an Add Adapter Factory Logic Task Parameters window is displayed. However, it contains different text and combo boxes.

For the FOR conditional expression, use Table 2–8 to understand the various regions of this Add Adapter Factory Logic Task Parameters window.

Table 2–8 Add Adapter Factory Logic Task Parameters for FOR Conditional Statement

Name	Description
Operand Type	These combo boxes contain types of operands, such as adapter tasks and adapter variables.
Comparator Combo Box	From this combo box, you can set the relationship between two operands (for example, <, =, >).
Operand Qualifier	These combo boxes contain the qualifiers for the operands.
Increment Combo Box	Within this area, you can set whether the initial value will increase or decrease, and by how much.

Note: If you select the ELSE, BREAK, RETURN, or CONTINUE conditional expressions, proceed to Step 8.

6. Set the parameters for your conditional expression.
This logic task will check to see if the length of the User ID is greater than 10 characters.
7. From the Add Adapter Factory Logic Task Parameters window toolbar, click **Save**.
The logic task you created is stored in the Oracle Identity Manager database.

8. From this window toolbar, click **Close**. The Add Adapter Factory Logic Task Parameters window disappears, and the main screen is active once again. The logic task that you created—for example, If (Check ID Length > 10)—appears within the Adapter Factory form.
9. (Optional.) Repeat Steps 3-8 to create additional logic tasks for the adapter.

Caution: All adapter tasks that can be executed for a condition of a logic task should be nested properly under that logic task.

See Also: [Section 2.10, "Changing the Order and Nesting of Tasks"](#) for more information about nesting tasks

You are now ready to compile the adapter, so it can be used with a process task or form field.

10. To compile the adapter, click **Build**.

The text in the Compile Status field changes from *Recompile* to *OK*. This indicates that Oracle Identity Manager compiled the adapter and did not find any errors. You can now attach the adapter to a process task or form field.

2.9 Modifying Adapter Tasks

The following procedure will show you how to edit an adapter task, in case you must make changes to it. To modify an adapter task

1. Select the adapter that contains the adapter task you wish to edit (for example, the *Update Solaris User Group* adapter).
2. Click the **Adapter Tasks** tab.
3. Double-click the adapter task that you want to modify.

The Edit Adapter Factory Task Parameters window is displayed, displaying information that relates to the adapter task you selected. Within this window, make the necessary modifications.

4. On the Edit Adapter Factory Task Parameters window toolbar, click **Save**.

The information you modified is stored in the Oracle Identity Manager database.

5. On the toolbar, click **Close**.

The Edit Adapter Factory Task Parameters window disappears. The main screen is active again. The modified task appears within the child table of the **Adapter Factory** form. You must re-compile the adapter, so it can be used with a process task or form field.

6. To recompile the adapter, click **Build**.

The text in the **Compile Status** field changes from *Recompile* to *OK*. This indicates that Oracle Identity Manager compiled the adapter and did not find any errors. You can now attach the adapter to a process task or form field.

Caution: You cannot modify the API call inside a Java, Xellerate API, or Utility task. The adapter task has to be deleted and re-created.

In addition, if *CODE GEN ERROR* appears in the Compile Status field, Oracle Identity Manager encountered errors while compiling the adapter. Rectify the errors, if necessary re-do the adapter task modifications, and compile the adapter again.

2.10 Changing the Order and Nesting of Tasks

If you add multiple tasks to an adapter, you can either change the order in which the tasks are executed, or place one task inside of another task for the adapter to work.

The following procedure will show you how to change the order and nesting of tasks.

Caution: You should not change the order and nesting of adapter tasks unless you understand the mapping dependencies of the adapter tasks.

To change the order and nesting of tasks:

1. Select the adapter that contains tasks of which you want to change the order and/or nest (for example, the *Check the Solaris User ID* adapter).
2. Click the **Adapter Tasks** tab.

The tasks appear, which belong to the current adapter.

In this example, the following changes must occur:

- The error handler task must be nested inside of the *IF (Check ID Length > 10)* logic task.
- The set variable task has to be nested inside of the **ELSE** logic task.
- The **IF** logic task precedes the **ELSE** logic task.

Therefore, you must first reorganize the logic tasks. Then, you must nest the error handler task and set variable task inside of the **IF** and **ELSE** logic tasks, respectively. To reorganize tasks:

3. Select the task that must run before another task, and click the **Up** arrow button. The selected task will switch places with the task that precedes it.

or

Select the task that must be executed after another task, and click the **Down** arrow button. The highlighted task is displayed below the task that previously followed it.

To nest tasks/remove task nestings:

4. Select the task that must be placed inside of another task, and click the **Right** arrow button. The selected task will be nested inside of the task that appears above it.

or

Select the task that no longer be nested inside of another task, and click the **Left** arrow button. The highlighted task will not be nested inside of the task that is displayed above it.

5. On the toolbar, click **Save**.

The order and nesting of the adapter's tasks is stored in the Oracle Identity Manager database. If the adapter is logically complete and all variables on the adapter tasks are mapped, you can compile it to use with a process task or form field.

6. To compile the adapter, click **Build**.

The text in the Compile Status field changes from *Recompile* to *OK*. This indicates that Oracle Identity Manager compiled the adapter and did not find any errors. You can now attach the adapter to a process task or form field.

Caution: If you see *CODE GEN ERROR* in the Compile Status field, Oracle Identity Manager found errors while compiling the adapter. Rectify the errors, if necessary re-do the adapter task modifications, and compile the adapter again.

2.11 Deleting Adapter Tasks

When an adapter task is no longer necessary for the adapter to run, you must remove it from the adapter. To delete an adapter task:

1. Select the adapter that contains the task you wish to remove (for example, the *Update Solaris User Group* adapter).
2. Click the **Adapter Tasks** tab.
3. Select the task that you want to remove (for example, the *CONTINUE* logic task).
4. Click **Delete**.

The selected task is deleted and disappears from the child table.

5. On the toolbar, click **Save**.
6. Recompile the adapter.

Caution: While deleting adapter tasks, ensure that the logic of the adapter is consistent and maintained.

2.12 Working with Responses

Adapters can have different outcomes, called **responses**. Based on these responses, adapters can trigger other process tasks.

For example, if the adapter returns a *True* response, the process task's status can be set automatically to *Completed*. However, if the adapter returns a *False* response, the process task's status can be set automatically to *Rejected*, and another process task can be triggered.

These responses can be added, modified, or removed on the Responses tab of the Adapter Factory form.

The following procedures will show you how to create, modify, and delete responses.

Note: Responses are used only with process task adapters, because these adapters are attached to process tasks. Rule generators, pre-populate adapters, and entity adapters are not connected to processes. In addition, task assignment adapters are not associated with responses. Therefore, if the active adapter is a task assignment adapter, rule generator, pre-populate adapter, or entity adapter, Oracle Identity Manager disables the Responses tab.

2.12.1 To Create a Response

1. Select the adapter to which you want to add responses (for example, the *Create Solaris User* adapter).
2. Click the **Responses** tab.
3. Click **Add**.

An empty row is inserted into the **Responses** tab.

4. Click the field that appears within the **Code Name** column.
5. Enter a code, which represents a response type that can be generated (for example, *True*).
6. Click the field that appears within the **Description** column.
7. Enter a description for this response (for example, *The user was created successfully*).
8. Double-click the field that appears within the **Status** column.

The Lookup popup window is displayed, containing the different status levels that you can associate with the response.

Note: For more information about Oracle Identity Manager's status levels, refer to Chapter 4, "About Process Task Adapters" on page 4-1.

9. Click the desired status level (for example, *Completed (C)*). Then, click **OK**.
- The Lookup window disappears, and the **Responses** tab is active once again.
10. Create another response, by clicking the **Add** button, and entering *False* and *The user was not created successfully*. into the Code Name and Description fields, respectively. Then, access the Lookup window, and assign the *Rejected (R)* status level to this response.
 11. On the toolbar, click **Save**.

The responses that you created for this adapter have been stored in the Oracle Identity Manager database. After you attach this adapter to a process task, these responses will appear in the Responses tab of the Editing Task window of the Process Definition form.

2.12.2 To Modify a Response

The following procedure demonstrates how to edit a response.

1. Select the adapter that contains the response you want to edit (for example, the *Create Solaris User* adapter).
2. Click the **Responses** tab.

3. Double-click the field of the response, which contains information that you want to modify.
 - a. If the field is a text field, Oracle Identity Manager enables it. You can now edit the contents within this field.
 - b. When the field is a lookup field, the Lookup popup window is displayed, containing the different status levels that you can associate with the response. Click the desired **status level**, click **OK**.

For example, double-click the **Status** column of the *False* response, select the *Suspended (S)* status level, and click **OK**.

4. On the toolbar, click **Save**.

The information that you modified for the response is stored in the Oracle Identity Manager database.

2.12.3 To Delete a Response

When a response is no longer necessary, you can delete it from the adapter.

1. Select the adapter, which contains a response that you want to remove.
2. Click the **Responses** tab.
3. Select the response that you want to delete.
4. Click **Delete**.

The response disappears. This indicates that Oracle Identity Manager has deleted the response.

2.13 Scheduling Rule Generators and Entity Adapters

Oracle Identity Manager triggers a process task adapter or a task assignment adapter automatically if it is attached to a process task, and the process task's status is *Pending*. In addition, Oracle Identity Manager always triggers pre-populate adapters on pre-insert. Therefore, you do not schedule when process task adapters, task assignment adapters, or pre-populate adapters will be executed.

On the other hand, a rule generator and an entity adapter are attached to a form field. The only way that Oracle Identity Manager will be able to execute the rule generator or entity adapter is for you to specify when it will be triggered. You do this through the Execution Schedule tab.

Note: If an entity adapter is attached to a process form or an object form for validation of field values, these adapters will trigger if we edit data in these forms after completing direct or request provisioning.

Using this tab, you can determine that Oracle Identity Manager will trigger the rule generator or entity adapter on preinsert or preupdate. In addition, you can also schedule an entity adapter to be executed on predelete, postinsert, postupdate, and postdelete.

This procedure demonstrates how to configure Oracle Identity Manager to trigger a rule generator or entity adapter.

2.13.1 Scheduling Rule Generators and Entity Adapters

To schedule rule generator and entity adapters:

1. Select the rule generator or entity adapter that you want Oracle Identity Manager to trigger (for example, *Solaris User ID Generator*).

Note: When you work with process task adapters or pre-populate adapters, you do not use the Execution Schedule tab. As a result, this tab, as well as its contents, are grayed out.

2. Click the **Execution Schedule** tab.

The contents of the Execution Schedule tab appear.

The following table will help you understand the various check boxes of the **Execution Schedule** tab:

Name	Description
Pre-Insert	By clicking this check box, Oracle Identity Manager can trigger the rule generator or entity adapter before the record is inserted into the database.
Pre-Update	When you click this check box, Oracle Identity Manager can trigger the rule generator or entity adapter before the record is updated in the database.
Pre-Delete	By clicking this check box, Oracle Identity Manager can trigger the entity adapter before the record is deleted from the database.
Post-Insert	When you click this check box, Oracle Identity Manager can trigger the entity adapter after the record is inserted into the database.
Post-Update	By clicking this check box, Oracle Identity Manager can trigger the entity adapter after the record is updated in the database.
Post-Delete	When you click this check box, Oracle Identity Manager can trigger the entity adapter after the record is deleted from the database.

Note: By clicking the check boxes of the Execution Schedule tab, you are defining the times when Oracle Identity Manager *can* trigger the rule generator or entity adapter. The Data Object Manager form allows you to specify when Oracle Identity Manager *will* trigger the rule generator or entity adapter.

For more information about the Data Object Manager form, refer to "Mapping Rule Generator Adapter Variables".

3. Enable the desired check boxes. Then, from the toolbar, click **Save**.

The criteria you set for Oracle Identity Manager to execute the rule generator or entity adapter is stored in the Oracle Identity Manager database.

Using Adapters

This chapter describes how to work with and manage the adapters you have created to connect with external resources. It contains these sections:

- [Working with Rule Generator Adapters](#)
- [Working with Entity Adapters](#)
- [Working with Task Assignment Adapters](#)
- [Working with Prepopulate Adapters](#)
- [Working with Process Task Adapters](#)
- [Adapter Mapping Information](#)

3.1 Working with Rule Generator Adapters

This section explains how to work with rule generator adapters, and contains these topics:

- [Mapping Rule Generator Adapter Variables](#)
- [Associating Rule Generators with Processes](#)
- [Removing Rule Generators from Form Fields](#)

3.1.1 Mapping Rule Generator Adapter Variables

In [Chapter 2, "Developing Adapters"](#), you learned how to create a rule generator. Now, you must map the adapter variables of the rule generator to their proper locations to ensure that the adapter will function as intended.

To map these adapter variables, access the Data Object Manager form from the Development Tools/Business Rule Definition folder of the Design Console.

To map the adapter variables of a rule generator to their proper locations:

1. Open the Data Object Manager form. In the Design Console workshops, the Data Object Manager form is displayed.

The following table lists and describes the various regions of the Data Object Manager form:

Name	Description
Form Description Field	From this lookup field, select the form that contains the field to which you are attaching the rule generator.

Name	Description
Data Object Field	This field displays the name of the data object, which is represented by the selected form.
Attach Handlers Tab	This tab displays: <ul style="list-style-type: none"> ■ The rule generators that are attached to the selected form. ■ The execution schedule of the rule generators associated with this form. ■ The order in which Oracle Identity Manager will run the rule generators. ■ Insert, update, and delete permissions for roles.
Map Adapters Tab	This tab displays: <ul style="list-style-type: none"> ■ The names of the rule generators that are associated with the form; ■ The status of these adapters. ■ The names, descriptions, and mapping statuses of the rule generators' adapter variables. <p>Note: The Map Adapters tab is grayed out until an adapter is assigned to the current data object.</p>

2. Double-click the **Form Description** field. A Lookup dialog box appears with the forms to which you can attach rule generators.
3. Select the form you want. Then, click **OK**.
4. On the toolbar, click **Save**.

The selected form, the form's data object, and the rule generator adapters associated with the form appear. In addition, Oracle Identity Manager enables the Map Adapters tab.

Tip: To change the sequence of triggering a rule generator:

1. Click **Assign**. The Event Handlers dialog box is displayed.
2. Select the rule generator from.
3. Click the up arrow and down arrow buttons to modify the order of the rule generator.

For these rule generators to work properly, you must map the adapter variables to their proper locations.

5. Click the **Map Adapters** tab.
6. From the Name combo box, select the rule generator, which has adapter variables that can be mapped (for example, the `adpCONVERTTOLOWERCASE` rule generator).

The Map Adapters tab now displays the following:

- The name of the rule generator that is to be attached to the form.
- The status of the rule generator.
- The names, descriptions, and mapping statuses of the rule generator's adapter variables.

See Also: ["Attaching Process Task Adapters to Process Tasks"](#) on page 3-14 for information about various mapping statuses for an adapter

7. Set the mappings for each variable that appears in the Adapter Variables region of the Map Adapters tab. To do so, double-click the row header of the variable you want to map (for example, *Data*). The Data Mapping for Variable dialog box is displayed.

[Table 3–1](#) describes the various fields of the Data Mapping for Variable dialog box.

Table 3–1 Fields of the Data Mapping for Variable Dialog Box

Field Name	Description
Variable Name	This field displays the name of the adapter variable for which you are setting a mapping (for example, <i>Data</i>).
Data Type	This field shows the data type of the adapter variable (for example, <i>String</i> is the data type for the <i>Data</i> adapter variable).
Map To	<p>This field contains the source and target locations of the mappings you can set for the adapter variable (for example, <i>User Definition</i>).</p> <p>When you map the adapter variable to a location or a contact, Oracle Identity Manager enables the adjacent combo box. From this combo box, select the specific type of location or contact to which you are mapping the adapter variable.</p> <p>If you are not mapping the adapter variable to a location or contact, this combo box is grayed out.</p>
Qualifier	This field contains the qualifiers for the mapping you selected in the Map To combo box (for example, <i>User Login</i>).
IT Asset Type	<p>This field enables you to select a specific IT Resource (for example, <i>Solaris</i>) when you map an adapter variable to an IT Resource, and this variable's data type is <i>String</i>.</p> <p>If you are not mapping the adapter variable to an IT Resource, or the variable's data type is not <i>String</i>, this field does not appear.</p>
IT Asset Property	<p>This field enables you to select a specific field that will receive the results of the mapping (for example, <i>User Name</i>), when you map an adapter variable to an IT Resource, and this variable's data type is <i>String</i>.</p> <p>If you are not mapping the adapter variable to an IT Resource, or the variable's data type is not <i>String</i>, this field does not appear.</p> <p>Important: The IT Asset Type and IT Asset Property fields are included within this window for backward compatibility. The preferred way is to create an adapter variable with a data type of IT Resource, in which case these fields will not appear.</p>
Literal Value	<p>When you map the adapter variable to a literal, type the name of the specific literal in this field (for example, <i>IBM</i>).</p> <p>If you are not mapping the adapter variable to a literal, this field does not appear.</p>

Complete the Map To, Qualifier, IT Asset Type, IT Asset Property, and Literal Value fields.

See Also: ["Adapter Mapping Information"](#) on page 3-18 for more information about the mappings to select

8. Click **Save**. Then, click **Close**

The Data Mapping for Variable window disappears. The Map Adapters tab is active again.

9. On the main screen toolbar, click **Save**.

Repeat Steps 7 and 8 for all adapter variables that can be mapped.

The contents in the Status field change from Mapping Incomplete to Ready. In addition, the mapping statuses for the adapter variables change from *No (N)* to *Yes (Y)*.

This signifies that all the adapter variables for the rule generator adapter have been mapped correctly. You are now ready to attach this rule generator to a provisioning process, so it can be triggered after the process is provisioned to a target user or organization.

Tip: When you map all the adapter variables for a rule generator that is associated with a form, a quick way to see the form to which it is attached as well as the execution schedule of the rule generator, is by accessing the Usage Lookup tab of the Adapter Factory form.

After the rule generator is assigned to a process, and the process is provisioned, the rule generator will be executed by Oracle Identity Manager.

3.1.2 Associating Rule Generators with Processes

After you map the adapter variables of a rule generator to their proper locations, you must attach it to a provisioning process. Then, once the process is provisioned to a target user or organization, Oracle Identity Manager will trigger the associated rule generator.

Similarly, when a rule generator, which has been assigned to a provisioning process, is no longer needed for the process to be completed, you must remove the rule generator from the provisioning process.

To assign a rule generator to a provisioning process or remove a rule generator from a provisioning process, access the Event Handlers/Adapters tab in the Process Definition form. This form can be found in the Process Management folder.

3.1.3 Removing Rule Generators from Form Fields

Sometimes, after you attach a rule generator to a form field, you can connect a different rule generator to that form field. When this occurs, you must first remove the rule generator that is currently attached to the form field.

Caution: If you remove a rule generator from a form and if the class name of the form's data object matches the table name of a provisioning process, you will not be able to assign the rule generator to that provisioning process.

For example, suppose the `adpCONVERTTOLOWERCASE` rule generator is removed from the Solaris form. If the class name of the form's associated data object is `UD_SOLARIS`, the rule generator cannot be assigned to any provisioning process with a table name of `UD_SOLARIS`.

To remove a rule generator from a form field, perform the following steps:

1. Open the Data Object Manager form.

2. Select the form that contains a rule generator you want to remove.
3. The selected form, along with its rule generators, appear in the Data Object Manager form.
4. Click the rule generator that you want to remove from the form field.
5. Click **Delete**.

The selected rule generator no longer appears in the Data Object Manager form. This indicates that you have removed the rule generator from the form field.

Caution: If you attempt to remove a rule generator from a form field, and if an error box appears, the adapter has already been associated with a provisioning process. First, detach the rule generator from the process. Then, you can remove it from the form field.

3.2 Working with Entity Adapters

For information about working with entity adapters, see:

- Entity Adapters in "[Types of Adapters](#)" on page 2-3.
- The procedures in "[Working with Rule Generator Adapters](#)" on page 3-1 for details about mapping the variables of an entity adapter with Oracle Identity Manager forms and/or provisioning processes.

3.3 Working with Task Assignment Adapters

This section contains these topics:

- [Attaching Task Assignment Adapters to Process Tasks](#)
- [Removing Task Assignment Adapters from Process Tasks](#)

3.3.1 Attaching Task Assignment Adapters to Process Tasks

In [Chapter 2, "Developing Adapters"](#), you learned how to create a task assignment adapter. Once created, you must attach it to a process task so that Oracle Identity Manager can automate the assignment of the task to a user or role.

To connect a task assignment adapter to a process task, access the Assignment tab (from the Process Definition form). From this tab, you can also map any adapter variables to their proper locations.

The following procedure shows you how to attach a task assignment adapter to a process task.

1. Open the Process Definition form, which is located in the Process Management folder.

Within the Oracle Identity Manager workspace, the Process Definition form appears.

2. Select the process, which contains a task to which you want to attach an adapter.
The selected process, along with its tasks, appears in the Process Definition form.
3. Double-click the row header of the task to which you want to attach a task assignment adapter.

The Editing Task window appears, containing information about the task (for example, the Get Solaris UUID process task).

4. Click the **Assignment** tab. The Assignment dialog box is displayed.
5. From this tab, click **Add**.

A blank row appears within the Assignment tab.

The following table lists the relevant fields of the Assignment tab:

Field Name	Description
Priority	From this field, set the priority number for the associated task assignment rule.
Rule	From this lookup field, select the rule that will determine if the associated adapter will be used to automate the assignment of the process task to a user or role.
Target Type	From this lookup field, specify whether the task is to be assigned to an Oracle Identity Manager user or role.
Adapter	From this lookup field, select the adapter that is to be associated with the designated task assignment rule.
Adapter Status	This field displays the mapping status of the adapter's variables. See " Attaching Process Task Adapters to Process Tasks " on page 3-14 for information about the various mapping statuses for an adapter.

6. Double-click the **Priority** field. From this field, set the priority number for the associated task assignment rule.
7. Double-click the **Rule** lookup field. From the Lookup dialog box that is displayed, select the rule that will determine if the associated adapter will be used to automate the assignment of the process task to a user or role.
8. Double-click the **Target Type** lookup field. From the Lookup dialog box that is displayed, specify whether the task is to be assigned to an Oracle Identity Manager user or role.
9. Double-click the **Adapter** lookup field. From the Lookup dialog box that is displayed, specify the task assignment adapter that is to be associated with the rule you selected in Step 7 of this procedure.
10. On the toolbar that is displayed within the Assignment tab, click **Save**.

The mapping status of the task assignment adapter variables is displayed within the Adapter Status field. Use the following table to decide which action to perform, based on the adapter's mapping status.

Mapping Status	Action
Ready	The adapter does not have any variables that can be mapped. In other words, none of the adapter variables are return variables or have been designated as Resolve at Run time. So, proceed to Step 14 of this procedure.
Mapping Incomplete	At least one of the adapter's variable must be mapped. So, proceed to Step 11 of this procedure.
Adapter Unavailable	After the adapter had been compiled successfully, it was modified. As a result, you must recompile the adapter.

Note: To learn more about the various mapping statuses for an adapter, see [Section 3.5.2, "Attaching Process Task Adapters to Process Tasks"](#).

11. Click Map.

The Adapter Variables window appears. It displays the following information:

- The name of the task assignment adapter that is attached to the process task;
- The status of the adapter; and
- The mapping statuses, names, and descriptions of the adapter's variables.

12. Set the mappings for each variable that appears in the **Adapter Variables region of this window. To do so, double-click the row header of the variable you want to map (for example, UUID).**

The Edit Data Mapping for Variable dialog box is displayed.

[Table 3–2](#) lists the fields of the Edit Data Mapping for Variable dialog box is displayed.

Table 3–2 Fields of the Edit Data Mapping for Variable Dialog Box

Field Name	Description
Variable Name	This field displays the name of the adapter variable for which you are setting a mapping (for example, UUID).
Data Type	This field shows the data type of the adapter variable (for example, <i>String</i> is the data type for the UUID variable).
Map To	<p>This field contains the types of mappings that you can set for the adapter variable (for example, IT Resources).</p> <p>When you map the adapter variable to a location or a contact, Oracle Identity Manager enables the adjacent combo box. From this combo box, select the specific type of location or contact to which you are mapping the adapter variable.</p> <p>In addition, if you map the adapter variable to a custom process form, and this form contains child table(s), Oracle Identity Manager enables the adjacent combo box. From this combo box, select the child table to which you are mapping the adapter variable.</p> <p>If you are not mapping the adapter variable to a location, contact, or child table of a custom process form, this combo box is grayed out.</p>
Qualifier	This field contains the qualifiers for the mapping you selected in the Map To combo box (for example, IT Asset).
IT Asset Type	<p>This field enables you to select a specific IT Resource (for example, <i>Solaris</i>) when you map an adapter variable to an IT Resource, and this variable's data type is String.</p> <p>If you are not mapping the adapter variable to an IT Resource, or the variable's data type is not String, this field does not appear.</p>

Table 3–2 (Cont.) Fields of the Edit Data Mapping for Variable Dialog Box

Field Name	Description
IT Asset Property	<p>This field enables you to select a specific field that will receive the results of the mapping (for example, Unique ID), when you map an adapter variable to an IT Resource, and this variable's data type is <i>String</i>.</p> <p>If you are not mapping the adapter variable to an IT Resource, or if the variable's data type is not <i>String</i>, this field does not appear.</p> <p>Important: The IT Asset Type and IT Asset Property fields are included within this window for backward compatibility. The preferred way is to create an adapter variable with a data type of IT Resource, in which case these fields will not appear.</p>
Literal Value	<p>When you map the adapter variable to a literal, use this field to specify the specific literal value.</p> <p>If you are not mapping the adapter variable to a literal, this field does not appear.</p>
Old Value	<p>By selecting this check box, you map the adapter variable to the value that was originally in the selected Qualifier field before modification.</p> <p>Process task adapters associated with process tasks are conditionally triggered when some field on the process form is changed. If you click the Old Value option, and the process task is marked Conditional, the value that is passed to the adapter is the previous value of the field. This is useful in cases of fields that accept passwords.</p> <p>For example, if you want to disallow setting the password to the same value, you can use the old value for comparison.</p> <p>If you are not mapping the adapter variable to a field that belongs to a child table of a custom process form, this check box is grayed out.</p>

13. Complete the Map To, Qualifier, IT Asset Type, IT Asset Property, Literal Value, and Old Value fields.

See Also: "[Adapter Mapping Information](#)" on page 3-18 for more information about the mappings to select

14. On the toolbar, click **Save**. Then, click **Close**.

The Edit Data Mapping for Variable window disappears. The Adapter Variables dialog box is active again.

The contents in the Status field change from Mapping Incomplete to Ready. In addition, the mapping statuses for the adapter's variables change from No (N) to Yes (Y).

15. Click **Save**. Then, click **Close**.

The Adapter Variable dialog box disappears, and the Assignment tab is active once again.

The adapter that you assigned to the process task (for example, *Assign Solaris Task*) now has a status of Ready.

16. From the toolbar that appears within the Assignment tab, click **Save** and **Close**

The Assignment tab disappears, and the main screen is active once again. This signifies that the task assignment adapter is attached to the process task.

Note: Once you attach a task assignment adapter to a process task, a quick way to see the process and the task to which it is connected is by accessing the Usage Lookup tab of the Adapter Factory form.

3.3.2 Removing Task Assignment Adapters from Process Tasks

When a task assignment adapter either becomes invalid, or is no longer necessary for Oracle Identity Manager to allocate the process task to a user or role, you must remove the adapter from the task.

3.3.2.1 To Remove a Task Assignment Adapter from a Process Task

To detach a task assignment adapter from a process task, perform the following tasks:

1. Open the Process Definition form.

The Process Definition form appears in the Design Console workspace.

2. Select the process, which contains a task from which you want to remove an adapter (for example, the Solaris 8 process).

The selected process, along with its tasks, appears in the Process Definition form.

3. Double-click the row header of the process task from which you want to remove the adapter (for example, the Get Solaris UUID task).

The Editing Task dialog box is displayed, containing information about the process task.

4. Click the **Assignment** tab.

The Assignment tab appears, displaying information about the adapter that is attached to the process task.

5. Highlight the row, containing the adapter that you want to remove from the process task.

6. Click **Delete**. The adapter no longer appears within the Assignment tab.

7. Click **Save**. Then, click **Close**.

The Assignment tab disappears, and the Main Screen is active once again. This signifies that the task assignment adapter is removed from the process task.

3.4 Working with Prepopulate Adapters

This section contains these topics:

- [Attaching Prepopulate Adapters to Form Fields](#)
- [Removing Prepopulate Adapters from Form Fields](#)

3.4.1 Attaching Prepopulate Adapters to Form Fields

To attach a prepopulate adapter to a form field, perform the following steps:

1. Select the field to which a prepopulate adapter will be attached.
2. Select the rule that will determine if the adapter will be used to populate the designated field with information.
3. Select the adapter that will be associated with the designated field.

4. Set the priority number of the selected rule.
5. Map the adapter variables of the prepopulate adapter to their proper locations.

Note: To attach a prepopulate adapter to a form field, you must ensure the following:

- The form is not in an active state. Otherwise, create a new form version.
 - After attaching the adapter, you must activate the form to be able to use it.
-

6. Open the Form Designer form.
7. Query for the form to which you want to attach a prepopulate adapter (for example, Solaris).
8. Click the **prepopulate** tab.

The prepopulate adapters, which have already been attached to the form you queried, appear within this tab.

Note: If no adapters have been attached to a form field, the prepopulate tab will be empty.

9. Click **Add**.

The prepopulate Adapters dialog box is displayed.

[Table 3–3](#) lists and describes the fields of the prepopulate Adapters dialog box.

Table 3–3 *Fields of the Prepopulate Adapter Dialog Box*

Name	Description
Field Name	This combo box contains a list of all of the form fields to which a prepopulate adapter can be attached.
Rule	From this lookup field, select the rule that will determine if the associated adapter will be used to populate the designated form field with information.
Adapter	From this lookup field, select the adapter that will be associated with the designated field.
Order	From this field, set the priority number of the selected rule.
Adapter Status	This field displays the mapping status of the adapter variables. See " Attaching Process Task Adapters to Process Tasks " on page 3-14 for information about the various mapping statuses for an adapter.

Table 3–3 (Cont.) Fields of the Prepopulate Adapter Dialog Box

Name	Description
Adapter Variables	<p>This area displays the following:</p> <ul style="list-style-type: none"> ■ Mapped: The mapping statuses of the adapter's variables. "Y" indicates that an adapter variable has been mapped properly; "N" indicates that this variable has not been mapped correctly. ■ Name: The names of the adapter variables. ■ Mapped to: The form fields to which the variables are mapped. If an adapter variable is not yet mapped, the corresponding cell in this column will be empty.

10. From the **Field Name** combo box, select the form field, such as User ID, to which the prepopulate adapter will be attached.
11. Double-click the **Rule** lookup field. From the Lookup dialog box that is displayed, select the rule that will determine if the associated adapter will be used to populate the designated form field with information (for example, Rule for Lowercase User ID).
12. Double-click the **Adapter** lookup field. From the Lookup dialog box that is displayed, choose the adapter that will be associated with the field you selected in Step 10, for example, Display Lowercase Letters for User ID.
13. In the **Order** field, enter the priority number of the rule you selected in Step 11, for example, 2.
14. On the prepopulate Adapters window toolbar, click **Save**.
15. Mapping Incomplete appears within the Adapter Status field. This signifies that the adapter you selected contains variables that have not been mapped correctly. These variables can be mapped to their proper locations. Otherwise, the adapter will not work.
16. Set the mappings for each variable that appears in the Adapter Variables region of the prepopulate Adapters window. To do so, double-click the row header of the variable you want to map, for example, UserID.

The Map Adapter Variables window is displayed.

[Table 3–4](#) describes the fields of the Map Adapter Variables window.

Table 3–4 Fields of the Map Adapter Variables Window

Field Name	Description
Variable Name	This field displays the name of the adapter variable for which you are setting a mapping (for example, UserID).
Data Type	This field shows the data type of the adapter variable (for example, <i>String</i> is the data type for the UserID adapter variable).
Map To	<p>This field contains the types of mappings that you can set for the adapter variable (for example, Process Data).</p> <p>When you map the adapter variable to a location or a contact, Oracle Identity Manager enables the adjacent combo box. From this combo box, select the specific type of location or contact to which you are mapping the adapter variable.</p> <p>If you are not mapping the adapter variable to a location or contact, this combo box is grayed out.</p>

Table 3–4 (Cont.) Fields of the Map Adapter Variables Window

Field Name	Description
Qualifier	This field contains the qualifiers for the mapping you selected in the Map to combo box (for example, User ID).
IT Asset Type	This field enables you to select a specific IT Resource (for example, Solaris) when you map an adapter variable to an IT Resource, and this variable's data type is String. If you are not mapping the adapter variable to an IT Resource, or the variable's data type is not String, this field does not appear.
IT Asset Property	This field enables you to select a specific field that will receive the results of the mapping (for example, <i>User Name</i>), when you map an adapter variable to an IT Resource, and this variable's data type is String. If you are not mapping the adapter variable to an IT Resource, or the variable's data type is not String, this field does not appear. Important: The IT Asset Type and IT Asset Property fields are included within this window for backward compatibility. The preferred way is to create an adapter variable with a data type of IT Resource, in which case these fields will not appear.
Literal Value	When you map the adapter variable to a literal, use this field to specify the specific literal value. If you are not mapping the adapter variable to a literal, this field does not appear.

17. Complete the Map To, Qualifier, IT Asset Type, IT Asset Property, and Literal Value fields.

See Also: ["Adapter Mapping Information"](#) on page 3-18 for more information about the mappings to select

18. On the Map Adapter Variable window toolbar, click **Save**. Then, click **Close**.

The Map Adapter Variables window disappears. The prepopulate Adapters window is active again.

The text in the Adapter Status field changes from Mapping Incomplete to Ready. In addition, the mapping statuses for the adapter's variables change from No (N) to Yes (Y).

19. On the prepopulate Adapters window toolbar, click **Close**.

The prepopulate Adapters window disappears, and the Form Designer form is active again. The prepopulate adapter, which you attached to the User ID form field (Display Lowercase Letters for User ID), appears in the prepopulate tab of the Results of 1Q Sales 2003 form.

After a process, which references this form, is provisioned to a target user or organization, the form will appear. Oracle Identity Manager will check to see if the prepopulate rule, which has the highest priority, is valid. If so, Oracle Identity Manager will assign the associated prepopulate adapter to the designated field (User ID), and execute it. At this point, one of the following actions occur:

- If the Auto-prepopulate check box is selected for the provisioning process, Oracle Identity Manager will display the data that is generated by the prepopulate adapter automatically.

- If the Auto-prepopulate check box is cleared, an Oracle Identity Manager user must manually trigger the displaying of the data that is generated by the prepopulate adapter. To do this, the administrator must click the prepopulate button on the form section of the direct provisioning wizard in the Web client, while provisioning the form to a user.

Tip: Once you allocate a prepopulate adapter to a form field, and assign a prepopulate rule to the adapter, a quick way to see the association among the adapter, the form field, and the rule is by accessing the Usage Lookup tab of the Adapter Factory form.

3.4.2 Removing Prepopulate Adapters from Form Fields

If a prepopulate adapter, which has been associated with a form field, is no longer valid, you must remove the adapter from the field.

Note: Before removing the prepopulate adapter from a form field, you must create a new version of the form.

To remove a prepopulate adapter from a form field:

1. Select the prepopulate adapter that you want to remove.
2. Click **Delete**. The prepopulate adapter is removed from the form field. It cannot be triggered when the form is launched.
3. After removing the adapter, you must activate the form.

3.5 Working with Process Task Adapters

This section contains these topics:

- [Guidelines for Working with a Process Task Adapter](#)
- [Attaching Process Task Adapters to Process Tasks](#)
- [Removing Process Task Adapters from Process Tasks](#)

3.5.1 Guidelines for Working with a Process Task Adapter

After you create a process task adapter, you attach it to the appropriate process task by using the Integration tab of the Process Definition form. From this tab, you can also map any variables of the adapter to their proper locations, which were designated as either *Resolve at Run time* or as an adapter return variable.

For example, the adapter named *adpSOLARISPASSWORDUPDATED* is connected to the *Password Updated* task of the *Solaris* process.

After you attach an adapter to a process task, for the adapter to be functional, it might need data from fields of other forms. For this example, the *adpSOLARISPASSWORDUPDATED* adapter cannot work unless it obtains the following information:

- The user's Oracle Identity Manager ID and password.
- The user's Solaris ID and password.
- The IP address where Solaris is located.

Therefore, it must get this information from the *UserID*, *Passwd*, *SolarisUserID*, *SolarisUserPasswd*, and *ServerAddress adapter* variables respectively. These five variables are created by using the Adapter Factory form. The "Y" that precedes each adapter variable signifies that it has been mapped correctly.

The form that enables you to create process-specific fields, which will be used by a process to obtain the information it needs, is called the Form Designer. When you create these fields, Oracle Identity Manager stores them into a table. Then, by associating this table with a process (through the Table Name lookup field of the Process Definition form), the adapter, which you attach to a task of this process, will use the table to retrieve the appropriate data.

If you want to modify this table, you can do so through the Form Designer form.

The *UD_SOLARIS* table contains two fields: *UD_SOLARIS_USERID* and *UD_SOLARIS_PASSWD*. By accessing this record of the Form Designer form, you can edit the fields of the table.

Once you attach the process task adapter to a dependent process task, and the status of this process task is *Pending* (the status of the previous process task is *Completed*), Oracle Identity Manager will trigger the adapter automatically. When the process task is an independent task, Oracle Identity Manager will execute the adapter as soon as the process is requested.

The result of the adapter being executed represents the state of the process task. When the adapter is finished successfully, the process task to which this adapter is attached will have a status of *Completed*.

On the other hand, if the adapter cannot perform its designated function, the process task to which this adapter is attached will have a status of *Rejected*. By discovering the cause of the error, you can modify the process task and/or adapter so it can run successfully.

Note: To determine why a process task might have failed:

Find the process task. When the process task has not yet been provisioned to the target user or organization, it is located in the To Do List or Pending Approvals. To find the task:

1. Log in as the user.
 2. Select the To Do List link or the Pending Approvals links in the left side of the window.
-
-

3.5.2 Attaching Process Task Adapters to Process Tasks

In the previous chapter, you learned how to create a process task adapter. You must attach it to a process task to execute that process task automatically.

To connect an adapter to a process task, access the Integration tab (from the Process Definition form). From this tab, you can also map any adapter variables to their proper locations.

The following procedure shows you how to attach a process task adapter to a process task:

1. Open the Process Definition form, which is located in the Process Management folder.

In the Oracle Identity Manager Workspace, the Process Definition form appears.

2. Select the process, which contains a task to which you want to attach an adapter. The selected process, along with its tasks, appears in the Process Definition form. For this example, the Solaris process has been selected.
3. Double-click the row header of the task to which you want to attach an adapter. The Editing Task window appears, containing information about the task (for example, the *Password Updated* process task).
4. Click the **Integration** tab.
5. Click **Add**.
The Handler Selection window appears.
6. To access Oracle Identity Manager adapters, click the **Adapter** option.
The adapters appear, which you can attach to the process task.
7. From this region, select the adapter that you want to attach to the process task, for example, the *adpSOLARISPASSWORDUPDATED* adapter.

Tip: For classification purposes, the first three letters of each adapter's name are *adp*. For classification purposes, the first three letters of each adapter's name are *adp*.

8. From the Handler Selection window's toolbar, click **Save**.
A dialog box appears, stating that the adapter was successfully added to the process task.
9. Click **OK**.
The dialog box disappears, and the **Integration** tab is now active. This tab now displays the following:
 - The name of the adapter that is attached to the process task;
 - The status of the adapter; and
 - The names, descriptions, and mapping statuses of the adapter's variables.

Note: An adapter can have one of three mapping statuses:

Ready. This adapter has been successfully compiled, and all of its variables have been mapped correctly.

Mapping Incomplete. This adapter has been successfully compiled, but at least one of its variables have not been mapped correctly.

Adapter Unavailable. After this adapter had been compiled successfully, it was modified, and recompiled.

Note: If an adapter does not have any mappable variables, the Adapter Variables region is empty. In addition, the Status field will display either *Ready* or *Adapter Unavailable*, depending on whether the adapter has to be recompiled.

Note: A mappable adapter variable either has been designated as *Resolve at Run time* or it is an adapter return variable.

Note: Once you attach the adapter to the process task, any responses that you defined for the adapter appear in the Responses tab of the Editing Task window.

10. Set the mappings for each variable that appears in the Adapter Variables region of the Integration tab. To do so, double-click the row header of the variable you want to map (for example, *SolarisUserID*).

The Data Mapping for Variable window is displayed.

Table 3–5 describes the fields of the Data Mapping for Variable window.

Table 3–5 Fields of the Data Mapping for Variable Window

Field Name	Description
Variable Name	This field displays the name of the adapter variable for which you are setting a mapping (for example, <i>SolarisUserID</i>).
Data Type	This field shows the data type of the adapter variable (for example, <i>String</i> is the data type for the <i>SolarisUserID</i> variable).
Map To	This field contains the types of mappings that you can set for the adapter variable (for example, <i>IT Resources</i>). When you map the adapter variable to a location or a contact, Oracle Identity Manager enables the adjacent combo box. From this combo box, select the specific type of location or contact to which you are mapping the adapter variable. In addition, if you map the adapter variable to a custom process form, and this form contains child table(s), Oracle Identity Manager enables the adjacent combo box. From this combo box, select the child table to which you are mapping the adapter variable. If you are not mapping the adapter variable to a location, contact, or child table of a custom process form, this combo box is grayed out.
Qualifier	This field contains the qualifiers for the mapping you selected in the Map to combo box (for example, <i>IT Asset</i>).
IT Asset Type	This field enables you to select a specific IT Resource (for example, <i>Solaris</i>) when you map an adapter variable to an IT Resource, and this variable's data type is <i>String</i> . If you are not mapping the adapter variable to an IT Resource, or the variable's data type is not <i>String</i> , this field does not appear.
IT Asset Property	This field enables you to select a specific field that will receive the results of the mapping (for example, <i>User Name</i>), when you map an adapter variable to an IT Resource, and this variable's data type is <i>String</i> . If you are not mapping the adapter variable to an IT Resource, or the variable's data type is not <i>String</i> , this field does not appear. Important: The IT Asset Type and IT Asset Property fields are included within this window for backward compatibility. The preferred way is to create an adapter variable with a data type of <i>IT Resource</i> , in which case these fields will not appear.
Literal Value	When you map the adapter variable to a literal, use this field to specify the specific literal value. If you are not mapping the adapter variable to a literal, this field does not appear.

Table 3–5 (Cont.) Fields of the Data Mapping for Variable Window

Field Name	Description
Old Value	<p>By selecting this check box, you map the adapter variable to the value that was originally in the selected Qualifier field before modification.</p> <p>Process task adapters associated with process tasks are conditionally triggered when some field on the process form gets changed. If you click the Old Value option, and the process task is marked Conditional, the value that is passed to the adapter is the previous value of the field, before it got modified. This is useful in cases of fields that accept passwords. For example, if you want to disallow setting the password to the same value, you can use the old value for comparison.</p> <p>If you are not mapping the adapter variable to a field that belongs to a child table of a custom process form, this check box is grayed out.</p>

11. Complete the Map To, Qualifier, IT Asset Type, IT Asset Property, Literal Value, and Old Value fields.

See Also: "[Adapter Mapping Information](#)" on page 3-18 for more information about the mappings to select

12. On the toolbar, click **Save**. Then, click **Close**.

The Data Mapping for Variable window disappears. The **Integration** tab is active again.

13. On the Editing Task window toolbar, click **Save**.

The contents in the **Status** field change from *Mapping Incomplete* to *Ready*. In addition, the mapping statuses for the adapter's variables change from *No (N)* to *Yes (Y)*.

14. On the toolbar, click **Close**.

The Editing Task window disappears, and the main screen is active once again. The adapter you added to the *Password Updated* task (*adpSOLARISPASSWORDUPDATED*) appears in the **Process Definition** form.

This signifies that the *adpSOLARISPASSWORDUPDATED* process task adapter was attached to the *Password Updated* process task.

Tip: Once you attach a process task adapter to a process task, a quick way to see the process and task to which it is connected is by accessing the Usage Lookup tab of the Adapter Factory form.

3.5.3 Removing Process Task Adapters from Process Tasks

If a process task adapter is no longer necessary for Oracle Identity Manager to complete the process task automatically, or when you wish to attach a different adapter to a process task, you must first remove the adapter that is attached to the process task.

This procedure will show you how to remove a process task adapter from a process task.

3.5.3.1 To Remove a Process Task Adapter from a Process Task

1. Open the Process Definition form.

In the Design Console workspace, the **Process Definition** form appears.

2. Select the process, which contains a task from which you want to remove an adapter (for example, the *Solaris* process).

The selected process, along with its tasks, appears in the **Process Definition** form.

3. Double-click the row header of the process task from which you want to remove the adapter (for example, the *Password Updated* task).

The Editing Task window appears, containing information about the process task. Click the Integration tab.

4. Click the **Integration** tab.

The Integration tab displays information about the adapter that is attached to the process task.

5. Click **Remove**.

A dialog box appears, asking if you want to remove the adapter from the process task.

6. Click **OK**.

A dialog box appears, signifying that the adapter has been removed from the process task.

7. Click **OK**.

The contents of the adapter no longer appear in the Integration tab.

8. On the toolbar, click **Close**.

The Editing Task window disappears, and the main screen is active once again. The adapter that was once linked to the *Password Updated* task (*adpSOLARISPASSWORDUPDATED*) no longer appears in the child table of the **Process Definition** form.

This signifies that you have removed the adapter from the process task.

3.6 Adapter Mapping Information

An adapter is a Java class, generated by the Adapter Factory, which enables Oracle Identity Manager to interact with an external JAR file, a target IT resource (for example, a resource asset), or a user-defined form. The Adapter Factory is a code-generation tool provided by Oracle Identity Manager, which enables a User Administrator to create Java classes.

An adapter extends the internal logic and functionality of Oracle Identity Manager. It automates process tasks, and defines the rules for the auto-generation and validation of data in fields within Oracle Identity Manager. There are five types of adapters: task assignment adapters, task adapters, rule generator adapters, pre-populate adapters, and entity adapters.

The following topics are discussed in this section:

- [Adapter Task Mapping Information](#)
- [Adapter Variable Mapping Information](#)

3.6.1 Adapter Task Mapping Information

An adapter task is one of the several possible components within an adapter. And this is a logical step within an adapter, equivalent to calling a programming language method. The following types of adapter tasks are available: Functional Tasks (Java Task, Remote Task, and Stored Procedure Task), Utility Tasks (Utility Task and Oracle Identity Manager API Task), and Logic Tasks (Set Variable Task and Error Handler Task).

This section lists the mappings that you can set for the parameters of an adapter task, in the following topics:

- [Adapter Variables](#)
- [Adapter Task](#)
- [Literal](#)
- [Adapter References](#)
- [Organization Definition](#)
- [Process Definition](#)
- [User Definition](#)

3.6.1.1 Adapter Variables

The following table lists and describes the items of the Map To list box of the Data Mapping for Variable window and the Name list box to which you can map the parameters of an adapter variable for an adapter task.

Map To Combo Box	Name Combo Box	Description
Adapter Variables	A list of adapter variables are displayed	<p>You can map the parameter to the adapter variables that you created for this adapter.</p> <p>Note: When the adapter variable's classification type is Object, it cannot be used with process task adapters.</p> <p>Note: If the adapter variable's classification type is IT Resource, then an Attribute combo box is displayed. From this combo box, select the attribute of the IT resource to which you wish to map the parameter.</p>

3.6.1.2 Adapter Task

The following table lists and describes the items of the Map To, Name, and Output combo boxes of the Adapter Factory form to which you can map the parameters of an adapter task.

Map To Combo Box	Name Combo Box	Output combo Box	Description
Adapter Task	A list of adapter tasks are displayed.	A list of output variables pertaining to the selected adapter task is displayed.	You can map the parameter to the adapter tasks that you created for this adapter.

3.6.1.3 Literal

The following table lists and describes the items of the Map To and Type combo boxes, as well as the Value field of the Adapter Factory form, to which you can map the parameters of a constant (or literal) for an adapter task.

Map To Combo Box	Type Combo Box	Value Field	Description
Literal	String, Boolean, Character, Byte, Date, Integer, Float, Long, Short, Double	Enter the value of the literal into this field.	You can map the parameter to a String, Boolean, Character, Byte, Date, Integer, Float, Long, Short, or Double data type, respectively.

3.6.1.4 Adapter References

The following table lists and describes the items of the Map To and Type combo boxes of the Adapter Factory form to which you can map the parameters of an adapter reference for an adapter task.

Map To Combo Box	Type Combo Box	Description
Adapter References	Event Handler Name or Database Reference	You can map the parameter to the active adapter.

3.6.1.5 Organization Definition

The following table lists and describes the items of the Map To and Field combo boxes of the Adapter Factory form to which you can map the parameters of an organization definition for an adapter task.

Map To combo box	Field Combo Box	Description
Organization Definition	Organization Name	You can map the parameter to the Organization Name field of the Organizations form.
	Organization Type	You can map the parameter to the Type field of the Organizations form.
	Organization ID	You can map the parameter to the Organization # field of the Organizations form.
	Organization Parent	You can map the parameter to the Parent Organization field of the Organizations form.
	Organization Status	You can map the parameter to the Status field of the Organizations form.
	Organization Parent ID	You can map the parameter to the parent_key field in the ACT database table.
	Any fields that are displayed in the User Defined Fields tab of the Organizations form.	You can map the parameter to the selected user-defined field.

3.6.1.6 Process Definition

The following table lists and describes the items of the Map To and Field combo boxes of the Adapter Factory form to which you can map the parameters of a process definition for an adapter task.

Map To Combo Box	Field Combo Box	Description
Process Definition	Name	You can map the parameter to the Name field of the Process Definition form.
	Type	You can map the parameter to the Type field of the Process Definition form.

3.6.1.7 User Definition

The following table lists and describes the items of the Map To and Field combo boxes of the Adapter Factory form to which you can map the parameters of a user definition for an adapter task.

Map To Combo Box	Field Combo Box	Description
User Definition	User Key	You can map the parameter to a key, representing a unique record of the Users form.
	First Name	You can map the parameter to the First Name field of the Users form.
	Middle Initial	You can map the parameter to the Middle Name field of the Users form.
	Last Name	You can map the parameter to the Last Name field of the Users form.
	User Login	You can map the parameter to the User ID field of the Users form.
	Password	You can map the parameter to user password of the Users form.
	Type	You can map the parameter to the Xellerate Type field of the Users form.
	User Status	You can map the parameter to the Status field of the Users form.
	Role	You can map the parameter to the Role field of the Users form.
	Identity	You can map the parameter to the Identity field of the Users form.
	Disabled	You can map the parameter to the Disable User check box of the Users form.
	Organization	You can map the parameter to the Organization field of the Users form.
	Manager	You can map the parameter to the Manager field of the Users form.
	Start Date	You can map the parameter to the Start Date field of the Users form.
End Date	You can map the parameter to the End Date field of the Users form.	

Map To Combo Box	Field Combo Box	Description
	Email	You can map the parameter to the Email field of the Users form.
	Provisioning Date	You can map the parameter to the Provisioning Date field of the Users form.
	Provisioned Date	You can map the parameter to the Provisioned Date field of the Users form.
	Deprovisioning Date	You can map the parameter to the Deprovisioning Date field of the Users form.
	Deprovisioned Date	You can map the parameter to the Deprovisioned Date field of the Users form.
	Any fields that are displayed in the User Defined Fields tab of the Users form.	You can map the parameter to the selected user-defined field.

3.6.2 Adapter Variable Mapping Information

For a newly created adapter to work, you can map data to the parameters of the adapter's tasks. For this reason, you create placeholders, also known as adapter variables, to map the data at run time. Once an adapter variable is not needed for the adapter to run, you can remove it from the adapter. After you have deleted the adapter variable, recompile the adapter.

When an adapter variable is not the adapter return variable, or it is not designated as Resolve at Run time, it should be mapped within the Variable List tab of the Adapter Factory form. On the other hand, if the adapter variable is classified as an adapter return variable, or the adapter variable is set to Resolve at Run time, it can be mapped at another location within Oracle Identity Manager. This location is contingent upon the adapter's type. For example, the variables of a process task adapter will be mapped at a different place than the variables of a pre-populate adapter. The following table lists the variables of a particular type of adapter that can be mapped.

Adapter Type	Location
Process Task	The Integration tab of the Editing Task window
Task Assignment	The Assignment tab of the Editing Task window
Rule Generator	The Map Adapters tab of the Data Object Manager form
Pre-Populate	The Pre-Populate tab of the Form Designer form
Entity	The Map Adapters tab of the Data Object Manager form

The following topics are discussed in this section:

- [From the Variable List Tab](#)
- [Process Task Adapter Variable Mappings](#)
- [Task Assignment Adapter Variable Mappings](#)
- [Rule Generator and Entity Adapter Variable Mappings](#)
- [Prepopulate Adapter Variable Mappings](#)

3.6.2.1 From the Variable List Tab

The following table lists the mappings that you can set from the Variable List tab.

Variable Type	Map To	Qualifier/Resource Type
Object	Adapter References	Database References
		Data Object References
	Set at run time (for Task Assignment adapters only)	Database References
		Data Object References
IT Resource	Resolve at Run time	The IT Resource types that are displayed in the Table view of the IT Resources Type Definition form
String, Character, Byte, Integer, Float, Long, Short, Double	Literal	If you are mapping the adapter variable to a literal, a Literal Value field is displayed below the Resource Type combo box. Within this field, enter the value of this literal.
	Resolve at Run time	NA
	Adapter References	Event Handler Name Note: If the data type of the adapter variable is not String, Adapter References cannot be selected from the Map To combo box.
Boolean	Literal	Boolean. If you select this resource type, two Literal Value options are displayed below the Resource Type combo box: True and False. Select the option that corresponds to the value of the adapter variable.
	Resolve at Run time	NA
Date	Literal	If you are mapping the adapter variable to a literal, a Literal Value lookup field is displayed below the Resource Type combo box. Double-click this lookup field. From the Date & Time window that is displayed, select the date and time that will be the value of this literal.
	Resolve at Run time	NA
	System Date	NA Note: This variable's value will reflect Oracle Identity Manager's date and time. Hence, you do not map it.

3.6.2.2 Process Task Adapter Variable Mappings

The following table lists the process task adapter variable mappings.

Variable Type	Map To	Qualifier/Description
Object (Adapter Return Variable)	Process Data	You can map the parameter to a field of either the associated custom process form, or a child table that belongs to this form.
	Response Code	NA
	Task Information	Note. You can map the parameter to the Note tab of the Task List form. Reason. You can map the parameter to the Error Details window. To access this window, double-click a task that is displayed within the Task List form.
	Process Definition	Name. You can map the parameter to the Name field of the Process Definition form. Type. You can map the parameter to the Type lookup field of the Process Definition form.
Object (Adapter Return Variable)	Organization Definition	The fields of the Organizations form to which you can map the adapter variable. Note: Because the data type of the adapter variable is Object, you cannot select Organization ID from the Qualifier combo box.
	User Definition	The fields of the Users form to which you can map the adapter variable.
IT Resource	IT Resource	You can map the parameter to an IT resource. This IT resource is a member of the IT resource type that is displayed in parenthesis from within the Data Type field.
	Process Data	You can map the parameter to a field of the associated process-specific form. Note: The only field names that are displayed in this combo box are ones with a data type of IT Resource Lookup Field.
String, Boolean, Character, Byte, Date, Integer, Float, Long, Short, Double	Process Data	You can map the parameter to a field of either the associated custom process form, or a child table that belongs to this form.
	Task Information	Note. You can map the parameter to the Note tab of the Task List form. Reason. You can map the parameter to the Error Details window. To access this window, double-click a task that is displayed within the Task List form.
	Process Definition	Name. You can map the parameter to the Name field of the Process Definition form. Type. You can map the parameter to the Type lookup field of the Process Definition form.

Variable Type	Map To	Qualifier/Description
	Organization Definition	The fields of the Organizations form to which you can map the adapter variable.
String, Boolean, Character, Byte, Date, Integer, Float, Long, Short, Double	User Definition	The fields of the Users form to which you can map the adapter variable.
	Literal	<p>If you are mapping the adapter variable to a literal, and the variable's data type is String, Character, Byte, Integer, Float, Long, Short, or Double, a Literal Value field is displayed below the Qualifier combo box. Within the field, enter the value of this literal.</p> <p>When you are mapping the adapter variable to a literal, and the variable's data type is Boolean, two Literal Value options are displayed below the Qualifier combo box: True and False. Select the option that corresponds to the value of the adapter variable.</p> <p>If you are mapping the adapter variable to a literal, and the variable's data type is Date, a Literal Value lookup field is displayed below the Qualifier combo box. Double-click this lookup field. From the Date & Time window that is displayed, select the date and time that will be the value of this literal.</p>
String	IT Resources	<p>If you are mapping the adapter variable to an IT Resource, three combo boxes are displayed below the Map To combo box: Qualifier, IT Asset Type, and IT Asset Property. From these combo boxes, select the qualifier for the mapping, the specific name of the IT resource, and the field of the IT resource that will receive the results of the mapping.</p> <p>Note: If the data type of the adapter variable is not String, IT Resources cannot be selected from the Map To combo box.</p>

3.6.2.3 Task Assignment Adapter Variable Mappings

The following table lists the task assignment adapter variable mappings.

Variable Type	Map To	Qualifier/Description
IT Resource	Object Data	You can map the parameter to an IT resource's instance key. This IT resource is a member of the IT resource type that is displayed in parenthesis from within the Data Type field.

Variable Type	Map To	Qualifier/Description
Object (Adapter Return Value)	IT Resource	You can map the parameter to an IT resource.
	Object Data	You can map the parameter to a field of either the associated custom resource object form, or a child table that belongs to this form.
	Response Code	NA
	Task Information	The fields of the Task List form to which you can map the adapter variable.
	Process Definition	The fields of the Process Definition form to which you can map the adapter variable.
	Organization Definition	The fields of the Organizations form to which you can map the adapter variable.
	User Definition	The fields of the Users form to which you can map the adapter variable.
String, Boolean, Character, Byte, Date, Integer, Float, Long, Short, Double	Object Data	You can map the parameter to a resource object's instance key.
	Task Information	The fields of the Task List form to which you can map the adapter variable.
	Process Definition	The fields of the Process Definition form to which you can map the adapter variable.
	Organization Definition	The fields of the Organizations form to which you can map the adapter variable.
String, Boolean, Character, Byte, Date, Integer, Float, Long, Short, Double	User Definition	The fields of the Users form to which you can map the adapter variable.
	Request Info	Request ID. You can map the parameter to the Request ID field of the Requests form.
		Request Action. You can map the parameter to the Request Action field of the Requests form.
		Request Priority. You can map the parameter to the Request Priority field of the Requests form.
	Request Target User	The fields of the Users form to which you can map the adapter variable.
Request Target Organization	The fields of the Organizations form to which you can map the adapter variable.	
	Requester Info	The fields of the Users form to which you can map the adapter variable.

Variable Type	Map To	Qualifier/Description
	Literal	<p>If you are mapping the adapter variable to a literal, a Literal Value field is displayed below the Qualifier combo box. Within the field, enter the value of this literal.</p> <p>Note: If the data type of the adapter variable is Boolean, two options are displayed in place of the field: True and False. Select the option that reflects the value of the adapter variable.</p> <p>Note: If the data type of the adapter variable is Object, Literal cannot be selected from the Map To combo box.</p>
String	IT Resources	<p>Resource Instance. You can map the parameter to an IT resource's instance key. This IT resource is a member of the IT resource type that is displayed in parenthesis from within the Data Type field.</p> <p>IT Asset Type. You can map the parameter to an IT resource type.</p>
String	IT Resources	<p>IT Asset Property. You can map this parameter to one of the properties that comprise the selected IT resource type.</p>

3.6.2.4 Rule Generator and Entity Adapter Variable Mappings

The following table lists the rule generator and entity adapter variable mappings.

Variable Type	Map To	Qualifier/Description
Object (Adapter Return Variable), IT Resource, String, Boolean, Character, Byte, Date, Integer, Float, Long, Short	Literal	<p>If you are mapping the adapter variable to a literal, a Literal Value field is displayed below the Qualifier combo box. Within the field, enter the value of this literal.</p> <p>Note: If the data type of the adapter variable is Object, Literal cannot be selected from the Map To combo box.</p>
	Entity Field	<p>You can map the adapter variable to a field of the associated process form. The name of this form is displayed in the Form Description field of the Data Object Manager form.</p>
	Organization Definition	<p>The fields of the Organizations form to which you can map the adapter variable.</p> <p>Note: If the data type of the adapter variable is not Object, you cannot select Organization ID and Organization Parent ID from the Qualifier combo box.</p>
	User Definition	<p>The fields of the Users form to which you can map the adapter variable.</p>

3.6.2.5 Prepopulate Adapter Variable Mappings

The following table lists the prepopulate adapter variable mappings.

Variable Type	Map To	Qualifier/Description
IT Resource	IT Resource	You can map the parameter to an IT resource. This IT resource is a member of the IT resource type that is displayed in parenthesis from within the Data Type field.
	Process Data	You can map the parameter to a field of the associated process-specific form. Note: The only field names that are displayed in this combo box are ones with a data type of IT Resource Lookup Field.
	Object, String, Boolean, Character, Byte, Date, Integer, Float, Long, Short, Double	You can map the parameter to a field of the associated process-specific form.
	Organization Definition	The fields of the Organizations form to which you can map the adapter variable.
String, Boolean, Character, Byte, Date, Integer, Float, Long, Short, Double	User Definition	The fields of the Users form to which you can map the adapter variable.
	Literal	If you are mapping the adapter variable to a literal, and the variable's data type is String, Character, Byte, Integer, Float, Long, Short, or Double, a Literal Value field is displayed below the Qualifier combo box. Within the field, enter the value of this literal. When you are mapping the adapter variable to a literal, and the variable's data type is Boolean, two Literal Value options are displayed below the Qualifier combo box: True and False. Select the option that corresponds to the value of the adapter variable. If you are mapping the adapter variable to a literal, and the variable's data type is Date, a Literal Value lookup field is displayed below the Qualifier combo box. Double-click this lookup field. From the Date & Time window that is displayed, select the date and time that will be the value of this literal.

Variable Type	Map To	Qualifier/Description
String	IT Resources	<p>If you are mapping the adapter variable to an IT Resource, three combo boxes are displayed below the Map To combo box: Qualifier, IT Asset Type, and IT Asset Property. From these combo boxes, select the qualifier for the mapping, the specific name of the IT resource, and the field of the IT resource that will receive the results of the mapping.</p> <p>Note: If the data type of the adapter variable is not String, then IT Resources cannot be selected from the Map To combo box.</p>

Using the Callback Service

The callback service invokes deployment-specific logic at predetermined points during Oracle Identity Manager event processing. Callback service is described in the following sections:

- [Introducing the Callback Service](#)
- [Mapping Oracle Identity Manager Attributes](#)
- [Sending Event Callbacks](#)
- [Configuring the Callback Service](#)
- [Troubleshooting the Callback Service](#)

4.1 Introducing the Callback Service

The callback service triggers notifications and callbacks that allow external applications to perform some action as a part of Oracle Identity Manager event processing. For example, if a user is created by using Oracle Identity Manager, an external application that is registered to be apprised of that type of provisioning event will be notified, and therefore, the application can add the same user information to their own local data store. This is a *notification*. Notifications allow applications to sync changes and data.

Note: A Web service endpoint for the external application must be registered in the Callback Service configuration. See "[Configuring the Callback Service](#)" on page 4-8 for more information.

When an external application initiates the user provisioning event with Oracle Identity Manager, other external applications that are registered to be apprised of that type of provisioning event are contacted by Oracle Identity Manager. If Oracle Identity Manager itself is invoked by the external application as part of the provisioning process, then the notification is referred to as a *callback*. Therefore, a callback is simply notification of an external application followed by a callback to Oracle Identity Manager. Callbacks sync up information about an event between all registered parties and Oracle Identity Manager. A callback can also indicate success or failure status for the event.

Note: The callback service infrastructure handles both callbacks and notifications. In this document, *callbacks* is used to refer to both.

There are a number of callback types handled by the callback service. They include:

- Post-processing callbacks that are returned to external applications. For example, a request to provision a user profile in the configured Oracle Identity Manager data store is sent from an external application to Oracle Identity Manager as a Web service call. The profile is created and a post-processing callback is returned to the application to confirm the profile creation. These callbacks can be synchronous or asynchronous depending on the Oracle Identity Manager configuration.
- Status change callbacks are sent to interested parties when there is any change in the request status. For example, if the status of a user provisioning request is changed to completed or failed, or a request status change is recognized because of a Segregation of Duties (SoD) check, then a status change callback is sent.

Note: This functionality was previously referred to as a completion callback.

- Callback responses are generated by third parties and communicate to Oracle Identity Manager that a response to a post-processing callback will be returned. This Callback Response Service is manually configured as documented in ["Configuring the Callback Service"](#) on page 4-8

The callback service is built to invoke callbacks for events from any source including a user interface, Oracle Identity Manager request API or Service Provisioning Markup Language (SPML) client. It is a listener service that receives responses to asynchronous client callbacks. When a client Web service is invoked, Oracle Identity Manager generates a unique identifier and sends it along with the call. When the client responds back, it must use this identifier to correlate the response with the original Web service request. The callback service takes appropriate action based on the client response. The following sections contain additional information regarding the components of the Callback Service:

- [Using Callbacks](#)
- [Understanding Event Processing](#)
- [Retrying Callbacks](#)

4.1.1 Using Callbacks

This section describes a use case for Oracle Identity Manager callbacks. This use case is specific to Segregation of Duties (SoD) status change notifications. The use case concerns James North, an existing user and member of the POApprover role. James is requesting two additional roles: the POCreator and Buyer. The request goes through the following process:

1. Oracle Identity Manager receives the request and creates an Assign Roles request with an ID of 21.

A *Request ID* is a unique identifier generated by Oracle Identity Manager and used to correlate the request with future responses and communications.

2. Oracle Identity Manager initiates an SoD validation with Oracle Applications Access Control Governor (OAACG).
3. The Oracle Identity Manager request status changes and a callback is returned to the callbacks-registered Web services indicating SoD Check in Progress. The callback contains the request ID, the requested operation (MODIFY), the target type (IDENTITY), the targetGUID (JNORTH) and the locale (en_US).

4. OAACG returns an Approved with Conditions response indicating the Buyer role is acceptable but the POCreator role is in violation due to James' existing POApprover role.
5. The Oracle Identity Manager request status changes and a second callback is sent to the callbacks-registered Web services indicating an SoD Remediation In Progress. It contains the same identification information as the previous callback.
6. A TAG named CFOOVERRIDE is placed in the rejection notes, indicating the Chief Financial Office (CFO) can approve this violation in selective cases.

You must configure OBR rules to invoke an exception approval if any tag is placed with the rejection notes that uses 'OVERRIDE' at the end.

7. AMX Rules decipher the tag and determine that CFO approval is required so that the task is assigned to the person with title 'CFO'.
8. The CFO either approves the request, or rejects it.
 - a. If the request is rejected, then the Oracle Identity Manager request status changes and a third callback is returned as SoD Remediation Rejected. The Oracle Identity Manager request is then closed with the SoD Remediation Rejected status.

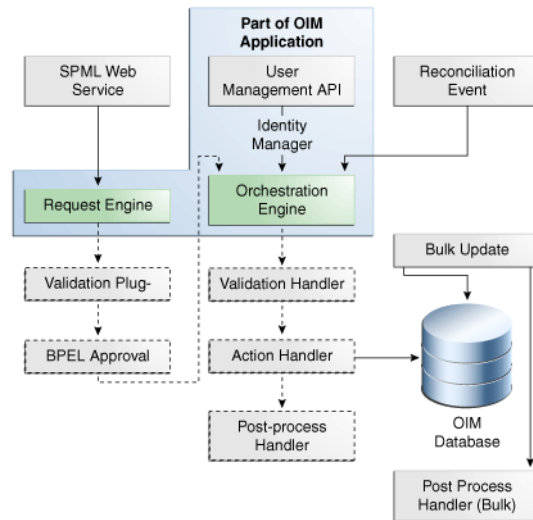
Note: If one portion of the request is rejected, then the entire request is rejected. Therefore, in this example, although the Buyer role does not have an SoD violation, it is not provisioned to the user.

- b. If the request is approved, then the Oracle Identity Manager request status changes and a third callback is returned as SOD Remediation Approved. The Oracle Identity Manager request is updated with the SOD Remediation Approved status. OAACG and requesting application is notified of the approval, and the roles are provisioned.
9. When the Oracle Identity Manager request status changes to Request Completed, then a final callback is sent to all the callbacks-registered Web services indicating that the request is completed with the Request Completed status.

4.1.2 Understanding Event Processing

Figure 4-1 illustrates how an event is processed. Oracle Identity Manager uses asynchronous invocation, giving the calling applications flexibility to process the event as needed, such as starting a human approval workflow.

Figure 4–1 Callback Service Process



For this release, the callback service supports a handler and a plugin event. They are:

- Post-processing Handler:** Invoked after the provisioning event is processed but before the request is marked as complete. The default post-processing handler will invoke multiple callbacks based on the callback service configuration.
- Status Change Plugin:** Invoked when there is any change in the request status; for example, if the status of a user provisioning request is changed to completed or failed, or a request status change is recognized due to a Segregation of Duties (SoD) check, then a status change callback will be sent. The default Status Change Plugin may invoke multiple callbacks based on the callback service configuration.

4.1.3 Retrying Callbacks

The callback service retries callbacks if there are network errors when invoking a service. Oracle Identity Manager retries the callback a fixed three times after a fixed five second interval. These values are not configurable.

4.2 Mapping Oracle Identity Manager Attributes

The names of attributes used by Oracle Identity Manager are proprietary. For example, a third-party application may refer to a user's last name as *surname* while Oracle Identity Manager uses *Last Name*. Because of this, attribute mapping is essential.

Note: The callback service uses the same attributes defined by the Service Provisioning Markup Language (SPML) layer of Oracle Identity Manager. These are referred to (singularly) as a SPML Provisioning Service Object (PSO). See [Appendix B, "SPML Attributes and LDAP Mappings, and Oracle Identity Manager Attributes"](#) for details.

Mapped attribute names are used in messages sent as well as in callback configuration (particularly, in the ConstraintAttribute parameter). [Table 4–1](#) defines how Oracle Identity Manager user type attributes are represented in callbacks using SPML PSOs.

Table 4–1 Oracle Identity Manager / Callback Service User Attribute Mapping

Callback Service Attribute (PSO)	Oracle Identity Manager User Attribute
activeEndDate	End Date
activeStartDate	Start Date
commonName	Common Name
countryName	Country
departmentNumber	Department Number
description	Description
displayName	Full Name
employeeNumber	Employee Number
employeeType	Role
facsimileTelephoneNumber	Fax
generationQualifier	Generation Qualifier
hireDate	Hire Date
homePhone	Home Phone
homePostalAddress	Home Postal Address
initials	Initials
localityName	Locality Name
mail	Email
middleName	Middle Name
mobile	Mobile
organization	LDAP Organization
organizationUnit	LDAP Organization Unit
pager	Pager
password	Password
postalAddress	Postal Address
postalCode	Postal Code
postOfficeBox	PO Box
preferredLangauge	Language
state	State
street	Street
surname	Last Name
telephoneNumber	Telephone Number
title	Title
userId	usr_key
userName	User Login

Table 4–2 defines how Oracle Identity Manager role type attributes are represented in callbacks using SPML PSOs.

Table 4–2 Oracle Identity Manager / Callback Service Role Attribute Mapping

Callback Service Attribute (PSO)	Oracle Identity Manager Role Attribute
commonName	Role Name
description	Role Description
displayName	Display Name

If the attribute name is not in either of these tables, it is referenced by its Oracle Identity Manager attribute.

4.3 Sending Event Callbacks

By default, callbacks are enabled (sent) for all Oracle Identity Manager events listed in EventHandlers.xml, the handler invoked by the Orchestration Engine during the post-processing stage of the provisioning process. Each event specifies the applicable entity type and operation. Specific callbacks may be disabled by changing the configuration. Table 4–3 summarize the user and role events for which Oracle Identity Manager makes callbacks and the information returned with the callback.

Table 4–3 Callback Initiated Events

Entity / Operation	Event Initiator	Returned in Post-Processing Handler	Returned in Status Change Plugin Callback
User Create	Third party requests only. No callbacks when a user is created using the console or through a reconciliation event.	<ul style="list-style-type: none"> ■ Target Type ■ Target GUID ■ Operation ■ Request ID ■ Provisioning Service Object (PSO) with created attributes and values (except password and challenge questions) <p>Note: PSO is used for create operations. Modification objects are used for modify operations.</p> <ul style="list-style-type: none"> ■ Roles assigned to the user 	<ul style="list-style-type: none"> ■ OIM Request Status ■ Target Type ■ Target GUID ■ Operation ■ Request ID
User Modify	All sources: third party requests, the console and through a reconciliation event.	<ul style="list-style-type: none"> ■ Target Type ■ Target GUID ■ Operation ■ Request ID ■ Modification object* with modified attributes and values (except password and challenge questions) 	<ul style="list-style-type: none"> ■ OIM Request Status ■ Target Type ■ Target GUID ■ Operation ■ Request ID

Table 4–3 (Cont.) Callback Initiated Events

Entity / Operation	Event Initiator	Returned in Post-Processing Handler	Returned in Status Change Plugin Callback
User Delete	All sources: third party requests, the console and through a reconciliation event.	<ul style="list-style-type: none"> ■ Target Type ■ Target GUID ■ Operation ■ Request ID 	<ul style="list-style-type: none"> ■ OIM Request Status ■ Target Type ■ Target GUID ■ Operation ■ Request ID
User Suspend (disable)	All sources: third party requests, the console and through a reconciliation event.	<ul style="list-style-type: none"> ■ Target Type ■ Target GUID ■ Operation ■ Request ID 	<ul style="list-style-type: none"> ■ OIM Request Status ■ Target Type ■ Target GUID ■ Operation ■ Request ID
User Resume (enable)	All sources: third party requests, the console and through a reconciliation event.	<ul style="list-style-type: none"> ■ Target Type ■ Target GUID ■ Operation ■ Request ID 	<ul style="list-style-type: none"> ■ OIM Request Status ■ Target Type ■ Target GUID ■ Operation ■ Request ID
User Assign Role - add memberOf	All sources: third party requests, the console and through a reconciliation event.	<ul style="list-style-type: none"> ■ Target Type ■ Target GUID ■ Operation ■ Request ID ■ Modification object* with GUIDs of assigned roles 	<ul style="list-style-type: none"> ■ OIM Request Status ■ Target Type ■ Target GUID ■ Operation ■ Request ID
User Revoke Role - delete memberOf	All sources: third party requests, the console and through a reconciliation event.	<ul style="list-style-type: none"> ■ Target Type ■ Target GUID ■ Operation ■ Request ID ■ Modification object* with GUIDs of assigned roles 	<ul style="list-style-type: none"> ■ OIM Request Status ■ Target Type ■ Target GUID ■ Operation ■ Request ID
Role Add	Third party requests only. No callbacks when a role is created using the console and through a reconciliation event.	<ul style="list-style-type: none"> ■ Target Type ■ Target GUID ■ Operation ■ Request ID ■ PSO* with created attributes and values 	<ul style="list-style-type: none"> ■ OIM Request Status ■ Target Type ■ Target GUID ■ Operation ■ Request ID
Role Modify	All sources: third party requests, the console and through a reconciliation event.	<ul style="list-style-type: none"> ■ Target Type ■ Target GUID ■ Operation ■ Request ID ■ Modification object* with modified attributes and values 	<ul style="list-style-type: none"> ■ OIM Request Status ■ Target Type ■ Target GUID ■ Operation ■ Request ID

Table 4–3 (Cont.) Callback Initiated Events

Entity / Operation	Event Initiator	Returned in Post-Processing Handler	Returned in Status Change Plugin Callback
Role Delete	All sources: third party requests, the console and through a reconciliation event.	<ul style="list-style-type: none"> ■ Target Type ■ Target GUID ■ Operation ■ Request ID 	<ul style="list-style-type: none"> ■ OIM Request Status ■ Target Type ■ Target GUID ■ Operation ■ Request ID

4.4 Configuring the Callback Service

Configuration of the callback service specifies how and when one or more callbacks are invoked. The following sections contain information on the configuration file and the procedure to import this file to the Metadata Services repository.

- [Understanding CallbackConfiguration.xml](#)
- [Importing CallbackConfiguration.xml](#)

4.4.1 Understanding CallbackConfiguration.xml

The configuration is stored in a single file called `CallbackConfiguration.xml`. This configuration file is located in the Oracle Identity Manager meta directory repository. It is used by the default event handlers and validation plug-in. The following parameters are configurable:

- **Policy name:** Defines the name of the callback policy. The value comes from the provisioning request. This is unique to Oracle Identity Manager and takes a string value.
- **Entity type:** Refers to the entity type for which the callback policy is applicable. It is a required, single value. Possible values are `User`, `Role`, and `RoleUser`.
- **Operation:** Refers to the database operations for which the callback policy is applicable. The required value may be either `Create` or `Delete`.
- **Description:** Takes a localized string that is a description of the policy.
- **ConstraintAttribute** and **ConstraintAttributeValue:** Fields specify a simple constraint that allows handlers and plug-in code to decide whether to invoke the particular callback for the given object. The attribute will be searched for in the entity data available to the handler, either in the form of orchestration data or `RequestData`. If the data does not exist, the constraint will not apply.
 - **ConstraintAttribute:** Takes as a value the name of an attribute on which the constraint is specified. The name must be the attribute name as defined on the application side as opposed to the name defined on the Oracle Identity Manager side. See "[Mapping Oracle Identity Manager Attributes](#)" on page 4-4 for more information.
 - **ConstraintAttributeValue:** Takes a value equal to the value the `ConstraintAttribute` must have. The value here must be the same as the value of the `ConstraintAttribute` present in the orchestration or request data. If the data has multiple values, at least one must match. This parameter is relevant only when `ConstraintAttribute` itself has a value.
- **Provisioning Steps:** Specifies the orchestration step for which this callback policy should be used. Possible values are:

- validation
- preProcessing
- approval
- postProcessing
- completion
- **stepName:** Refers to a Web service endpoint for the external application.
- **description:** Takes a localized string that is a description of the Web service endpoint.
- **InvokeOnChange:** Takes as a value one or more attribute names and is applicable only to modify operations. The callback will be invoked only when one of the attributes listed as a value of this parameter has changed. The value must be the attribute name as defined on the application side as opposed to the name defined on the Oracle Identity Manager side. See "[Mapping Oracle Identity Manager Attributes](#)" on page 4-4 for more information.
- **CallbackOnly:** Specifies whether Oracle Identity Manager should wait for a response from the external application. Possible values are true or false. If true, then Oracle Identity Manager will wait for a response from the application and, until a response is received, the orchestration process will be waiting. If false, then Oracle Identity Manager will not wait for a callback response and the orchestration process will continue.
- **targetIDAttribute:** Takes as a value an attribute that should be used as the target GUID in the message. The value must be the attribute name as defined on the application side as opposed to the name defined on the Oracle Identity Manager side. See "[Mapping Oracle Identity Manager Attributes](#)" on page 4-4 for more information. The default value is LDAP GUID.
- **roleIDAttribute:** Takes as a value the role attribute that should be used as role GUID in the message. The value must be the attribute name as defined on the application side as opposed to the name defined on the Oracle Identity Manager side. See "[Mapping Oracle Identity Manager Attributes](#)" on page 4-4 for more information. The default value is LDAP GUID.

[Example 4-1](#) is a sample configuration file.

Example 4-1 Sample CallbackConfiguration.xml

```
<?xml version='1.0' encoding='UTF-8'?>
<callbackConfiguration xmlns="http://www.oracle.com/schema/oim/callback_config"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.oracle.com/schema/oim/callback_config">
<policy>
<policyName>User Creation1</policyName>
<entityType>User</entityType>
<operation>Create</operation>
<description>Policy to create a user in the store</description>
<provisioningSteps>
<postProcessing>
<asyncSteps>
<stepName>http://myhost.mycompany.com:7001/testCallbackService/
  PostProcessingPluginRequestPortImplTest</stepName>
<description>Webservice url for this policy</description>
</asyncSteps>
</postProcessing>
</provisioningSteps>
```

```

</policy>
<policy>
<policyName>User Enable1</policyName>
<entityType>User</entityType>
<operation>Enable</operation>
<description>Policy to enable a user in the store</description>
<provisioningSteps>
<postProcessing>
<asyncSteps>
<stepName>http://myhost.mycompany.com:7001/testCallbackService/
  PostProcessingPluginRequestPortImplTest</stepName>
<description>Webservice url for this policy</description>
</asyncSteps>
</postProcessing>
</provisioningSteps>
</policy>
<policy>
<policyName>User Delete1</policyName>
<entityType>User</entityType>
<operation>Delete</operation>
<description>Policy to delete a user in the store</description>
<provisioningSteps>
<postProcessing>
<asyncSteps>
<stepName>http://myhost.mycompany.com:7001/testCallbackService/
  PostProcessingPluginRequestPortImplTest</stepName>
<description>Webservice url for this policy</description>
</asyncSteps>
</postProcessing>
</provisioningSteps>
</policy>
<policy>
<policyName>User Disable1</policyName>
<entityType>User</entityType>
<operation>Disable</operation>
<description>Policy to disable a user in the store</description>
<provisioningSteps>
<postProcessing>
<asyncSteps>
<stepName>http://myhost.mycompany.com:7001/testCallbackService/
  PostProcessingPluginRequestPortImplTest</stepName>
<description>Webservice url for this policy</description>
</asyncSteps>
</postProcessing>
</provisioningSteps>
</policy>
<policy>
<policyName>User Modify1</policyName>
<entityType>User</entityType>
<operation>Modify</operation>
<description>First Policy to modify a user in the store</description>
<provisioningSteps>
<postProcessing>
<asyncSteps>
<stepName>http://myhost.mycompany.com:7001/testCallbackService/
  PostProcessingPluginRequestPortImplTest</stepName>
<description>Webservice url for this policy</description>
</asyncSteps>
</postProcessing>
</provisioningSteps>

```



```

</policy>
<policy>
<policyName>Role Assign1</policyName>
<entityType>RoleUser</entityType>
<operation>CREATE</operation>
<description>Policy to assign roles to the user</description>
<provisioningSteps>
<postProcessing>
<asyncSteps>
<stepName>http://myhost.mycompany.com:7001/testCallbackService/
  PostProcessingPluginRequestPortImplTest</stepName>
<description>Webservice url for this policy</description>
</asyncSteps>
</postProcessing>
</provisioningSteps>
</policy>
<policy>
<policyName>Role Revoke1</policyName>
<entityType>RoleUser</entityType>
<operation>Delete</operation>
<description>Policy to revoke role from the user</description>
<provisioningSteps>
<postProcessing>
<asyncSteps>
<stepName>http://myhost.mycompany.com:7001/testCallbackService/
  PostProcessingPluginRequestPortImplTest</stepName>
<description>Webservice url for this policy</description>
</asyncSteps>
</postProcessing>
</provisioningSteps>
</policy>
<policy>
<policyName>Role Creation1</policyName>
<entityType>Role</entityType>
<operation>Create</operation>
<description>Policy to create a role in the store</description>
<provisioningSteps>
<postProcessing>
<asyncSteps>
<stepName>http://myhost.mycompany.com:7001/testCallbackService/
  PostProcessingPluginRequestPortImplTest</stepName>
<description>Webservice url for this policy</description>
</asyncSteps>
</postProcessing>
</provisioningSteps>
</policy>
<policy>
<policyName>Role Delete1</policyName>
<entityType>Role</entityType>
<operation>Delete</operation>
<description>Policy to delete a role in the store</description>
<provisioningSteps>
<postProcessing>
<asyncSteps>
<stepName>http://myhost.mycompany.com:7001/testCallbackService/
  PostProcessingPluginRequestPortImplTest</stepName>
<description>Webservice url for this policy</description>
</asyncSteps>
</postProcessing>
</provisioningSteps>

```

```
</policy>
<policy>
<policyName>Role Modify1</policyName>
<entityType>Role</entityType>
<operation>Modify</operation>
<description>Policy to modify a role in the store</description>
<provisioningSteps>
<postProcessing>
<asyncSteps>
<stepName>http://myhost.mycompany.com:7001/testCallbackService/
  PostProcessingWebPluginRequestPortImplTest</stepName>
<description>Webservice url for this policy</description>
</asyncSteps>
</postProcessing>
</provisioningSteps>
</policy>
</callbackConfiguration>
```

4.4.2 Importing CallbackConfiguration.xml

Following is the procedure to configure the callback service. It entails importing the CallbackConfiguration.xml file. This file contains list of callback policies for a given entity type or operation.

1. Create a credential entry in the Credential Store Framework (CSF) with the map name oim and key appid.keycredentials.

This entry stores the user name and password that Oracle Identity Manager will use to identify itself for callbacks. The CSF is available as part of domain creation when Oracle Identity Manager is installed.

See Also: *Oracle Fusion Middleware Security Guide* for information about the Credential Store Framework

2. Import CallbackConfiguration.xml to the Metadata Services (MDS) repository using the weblogicImportMetadata.sh MDS utility.

It should be loaded under the /metadata/iam-features-callbacks/sample_data/ namespace.

The following should be considered when importing CallbackConfiguration.xml:

- Ensure that the CallbackConfiguration.xml file is imported to the MDS repository under the exact metadata namespace:
/metadata/iam-features-callbacks/sample_data/
- Remove old CallbackConfiguration.xml files in other metadata namespaces. For example,
/metadata/iam-features-callbacks/old_config/CallbackConfiguration_st3b16.xml
and /metadata/iam-features-callbacks/old_config/CallbackConfiguration.xml
are not valid. Remove any invalid entries found in the MDS repository using weblogicDeleteMetadata.sh.
- If modifications are made to CallbackConfiguration.xml after it has been imported, then reimport the modified file as documented and purge the Oracle Identity Manager cache by using the PurgeCache.sh utility located in the *OIM_HOME*/server/bin/ directory.

4.5 Troubleshooting the Callback Service

Table 4–4 lists the troubleshooting steps that you can perform if you encounter callback service errors:

Table 4–4 *Troubleshooting Callback Service*

Problem	Solution
Not able to submit SPML request (Failed Authentication).	<p>Make sure that the SPML request is submitted with a valid SPML user (member of SPML_App_Role group) with its correct credentials.</p> <p>If the request is submitted from client APIs, then note that compatible client policy must be applied.</p> <p>The following is the sample error response displayed when a SPML request is submitted with incorrect credentials:</p> <pre data-bbox="699 611 1455 869"><env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"><env:Header/><env:Body><env:Fault xmlns:ns0="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"><faultcode>ns0:FailedAuthentication</faultcode><faultstring>FailedAuthentication : The security token cannot be authenticated.</faultstring><faultactor/></env:Fault></env:Body></env:Envelope></pre>

Table 4–4 (Cont.) Troubleshooting Callback Service

Problem	Solution
<p>For a given request type, for example Assign Role, Oracle Identity Manager is making callbacks to more than one callback Web service although the policyName matches with one callback service.</p>	<p>This is because when callbackOnly is set to false for all the eligible entity type and operation, for example Assign Role request types, the callbacks are triggered for all matching entity types and operations.</p> <p>PolicyName matching is ignored when callbackOnly attribute is set to false.</p> <p>If callbackOnly Attribute is set to true, then it checks for the policy name. All the callback Web service URLs present in that policy are triggered when the entity type and operation condition is also met.</p> <p>All the callback Web service URLs present in that policy are triggered when the entity type and operation condition is also met.</p>
<p>The policy reference URI 'oracle/wss_saml_or_username_token_service_policy' is not valid.</p>	<p>Make sure that WSM Policy Manager is deployed and targeted to the interacting servers such as Oracle Identity Manager and SPML request starting server.</p> <p>In addition, make sure that WSM Policy Manager is in active mode and is ready for serving the requests.</p>
<p>Not sure what is SPML APPID and Oracle Identity Manager APPID, and where these APPIDs are to be created.</p>	<p>SPML APPID is used for submitting SPML requests to Oracle Identity Manager. Any client that seeks user provisioning service with Oracle Identity Manager must contain SPML APPID in their repository. For example, when Fusion Applications is the client to Oracle Identity Manager, Fusion Applications typically use LDAP directory as their repository.</p> <p>Oracle Identity Manager APPID is used for sending callbacks to all the Web services registered in the CallbackConfiguration.xml file for a given SPML request type.</p> <p>Oracle Identity Manager repository or database contains only SPML APPID. Oracle Identity Manager APPID is not present in Oracle Identity Manager repository but is present in the Credentials Store Framework (CSF) under map name oim and with key appid.credentials.</p> <p>SPML repository or LDAP contains both SPML APPID and Oracle Identity Manager APPID.</p> <p>When Fusion Applications sends a SPML request to Oracle Identity Manager, it uses SPML APPID to communicate to Oracle Identity Manager. This SPML APPID is present in the SPML repository or LDAP. This user is authenticated at Oracle Identity Manager side against the database. Therefore, Oracle Identity Manager database contains SPML APPID in it.</p> <p>When Oracle Identity Manager communicates with Fusion Applications, it uses Oracle Identity Manager APPID to communicate to Fusion Applications. This Oracle Identity Manager APPID is present in the CSF. This user is authenticated at Fusion Applications side against LDAP by checking the Oracle Identity Manager APPID in LDAP repository. Therefore, LDAP contains Oracle Identity Manager APPID in it.</p>

Developing Rules

This chapter describes the Business Rule Definition of the Design Console. It contains the following topics:

- [Overview of Business Rule Definition](#)
- [Event Handler Manager Form](#)
- [Data Object Manager Form](#)
- [Reconciliation Rules Form](#)

5.1 Overview of Business Rule Definition

The Development Tools/Business Rule Definition folder provides system administrators and developers with tools to manage the event handlers and data objects of Oracle Identity Manager.

This folder contains the following forms:

- **Event Handler Manager:** This form lets you create and manage the event handlers that are used with Oracle Identity Manager.
- **Data Object Manager:** This form lets you define a data object, assign event handlers and adapters to it, and map any adapter variables associated with it.

5.2 Event Handler Manager Form

This form is displayed in the Development Tools/Business Rule Definition folder. You use this form to manage the Java classes that process user-defined or system-generated actions (or events). These classes are known as event handlers. When you add a new event handler to Oracle Identity Manager, you must first register it here so that Oracle Identity Manager can recognize it.

There are two types of event handlers:

- Event handlers that are created through the Adapter Factory form. These begin with the letters `adp`. They are known as adapters.
- Event handlers that are created internally in Oracle Identity Manager. These begin with the letters `tc`. They are referred to as system event handlers.

By using the Event Handler Manager form, you can specify when you want Oracle Identity Manager to trigger an event handler. An event handler can be scheduled to run as follows:

- **Pre-Insert:** Before information is added to the database

- **Pre-Update:** Before information is modified in the database
- **Pre-Delete:** Before information is removed from the database
- **Post-Insert:** After information is added to the database
- **Post-Update:** After information is modified in the database
- **Post-Delete:** After information is removed from the database

Figure 5–1 shows the Event Handler Manager form.

Figure 5–1 Event Handler Manager Form

Table 5–1 describes the fields of the Event Handler Manager form.

Table 5–1 Fields of the Event Handler Manager Form

Field Name	Descriptions
Event Handler Name	The name of the event handler.
Package	The Java package to which the event handler belongs.
Pre-Insert	If you select this check box, Oracle Identity Manager will trigger the event handler before information is added to the database.
Pre-Update	If you select this check box, Oracle Identity Manager will trigger the event handler before information is modified in the database.
Pre-Delete	If you select this check box, Oracle Identity Manager will trigger the event handler before information is removed from the database.

Table 5–1 (Cont.) Fields of the Event Handler Manager Form

Field Name	Descriptions
Post-Insert	If you select this check box, Oracle Identity Manager will trigger the event handler after information is added to the database.
Post-Update	If you select this check box, Oracle Identity Manager can trigger the event handler after information is modified in the database.
Post-Delete	If you select this check box, Oracle Identity Manager will trigger the event handler after information is removed from the database.
Notes	Additional information about the event handler.

The following sections describe how to create and modify event handlers.

Note: To use an event handler, you must attach it to a data object by using the Data Object Manager form. For more information about assigning event handlers to data objects, see [Section 5.3, "Data Object Manager Form"](#).

Caution: Any event handler that begins with the letters `adp` is associated with adapters, and should not be modified. However, you can modify system event handlers. These event handlers begin with the letters `tc`.

Adding or Modifying an Event Handler

To add or modify an event handler:

1. Open the Event Handler Manager form.
2. To add an event handler to Oracle Identity Manager, enter the name of the event handler into the **Event Handler Name lookup** field.
To modify an event handler, double-click the **Event Handler Name lookup** field. From the Lookup dialog box that is displayed, select the event handler that you want to edit.
3. In the **Package** field, add or edit the name of the Java package of which the event handler is a member.
4. Select the check boxes that correspond to when you want Oracle Identity Manager to trigger the event handler.

You can schedule an event handler to run on `preinsert`, `preupdate`, `predelete`, `postinsert`, `postupdate`, and `postdelete`.

Note: Selecting a check box does not mean that the event handler is triggered at that time, for example, on `preinsert`. It signifies that the event handler can run at that time.

5. In the **Notes** area, add or edit explanatory information about the event handler.
6. Click **Save**.

The event handler is added or modified.

5.3 Data Object Manager Form

The Data Object Manager form is displayed in the Development Tools/Business Rule Definition folder. You use this form to:

- Assign a rule generator adapter, entity adapter, or an event handler to an object that can add, modify, or delete data in the database. This type of object is known as a data object.
- Schedule the adapter or event handler to run according to a schedule (pre-insert, pre-update, pre-delete, post-insert, post-update, or post-delete).
- Organize the order in which Oracle Identity Manager triggers adapters or event handlers that belong to the same execution schedule.
- View the user groups that can add, modify, and delete the current data object.
- Map the variables of an adapter to their proper source and target locations.

See Also: [Chapter 2, "Developing Adapters"](#) for more information about adapter variables, rule generator adapters, and entity adapters

Figure 5–2 shows the Data Object Manager form.

Figure 5–2 Data Object Manager Form

Table 5–2 describes the fields of the Data Object Manager form.

Table 5–2 Fields of the Data Object Manager Form

Fields	Description
Form Description	The name of the form that is associated with the data object.
Data Object	The name of the data object to which you are assigning event handlers rule generator adapters, or entity adapters.

The following section describes how to select the target data object to which a rule generator adapter, entity adapter, or event handler will be assigned.

Selecting a Target Data Object

To select a target data object:

1. Open the Data Object Manager form.
2. Double-click the **Form Description** field.

From the Lookup dialog box displayed, select the name of the form that is associated with the data object to which you want to assign an event handler, rule generator adapter, or entity adapter.

After you select a form, the name of the corresponding data object is displayed in the **Data Object** field.

3. Click **Save**.

The target data object is selected. You can now assign rule generator adapters, entity adapters, and event handlers to it.

5.3.1 Tabs of the Data Object Manager Form

After you start the Data Object Manager form and select a target data object, the tabs of this form become functional.

The Data Object Manager form contains the following tabs:

- Attach Handlers
- Map Adapters

Each of these tabs is described in the following sections:

- [Attach Handlers Tab](#)
- [Map Adapters Tab](#)

5.3.1.1 Attach Handlers Tab

You use this tab to select the rule generator adapters, entity adapters, or event handlers that will be assigned to or removed from a data object. This includes the following:

- Specifying when Oracle Identity Manager triggers the assigned event handlers or adapters (on pre-insert, pre-update, pre-delete, post-insert, post-update, or post-delete).
- Setting the order in which Oracle Identity Manager triggers the adapters or event handlers that belong to the same execution schedule.

When an event handler, rule generator adapter, or entity adapter must no longer be triggered by Oracle Identity Manager, you must remove it from the data object.

For example, Oracle Identity Manager can trigger the `adpCONVERTTOLOWERCASE`, `adpSOLARISHMDSTRINGGEN`, `adpSETSOLARISASSET`, and `adpSETPASSWORDFROMMAIN` adapters on pre-insert. Based on the sequence numbers of these adapters, Oracle Identity Manager triggers the `adpCONVERTTOLOWERCASE` adapter first, followed by the `adpSOLARISHMDSTRINGGEN`, `adpSETSOLARISASSET`, and `adpSETPASSWORDFROMMAIN` adapters, respectively.

Note: To see the user groups that can add, modify, and delete the current data object, click the **Insert Permissions**, **Update Permissions**, or **Delete Permissions** tabs, respectively.

The following sections discuss these procedures:

- Assigning an event handler, rule generator adapter, or entity adapter to a data object
- Organizing the execution schedule of event handlers or adapters
- Removing an event handler, rule generator adapter, or entity adapter from a data object

5.3.1.1.1 Assigning an Event Handler or Adapter to a Data Object

To assign an event handler or adapter:

1. Select the tab of the Data Object Manager form that represents when you want the adapter or event handler to be triggered.

For example, if you want Oracle Identity Manager to activate an adapter on pre-insert, select the **Pre-Insert** tab.

2. From the selected tab, click **Assign**.

The Assignment dialog box is displayed.

3. Select the event handler or adapter, and assign it to the data object.

4. Click **OK**.

The event handler or adapter is assigned to the data object.

5.3.1.1.2 Organizing the Execution Schedule of Event Handlers or Adapters

To organize the execution schedule:

1. Select the event handler or adapter whose execution schedule you want to change.

2. Click **Assign**.

The Assignment dialog box is displayed.

3. Select the event handler or adapter.

4. If you click **Up**, the selected event handler or adapter will switch places and sequence numbers with the event handler or adapter that precedes it.

If you click **Down**, the selected event handler or adapter will switch places and sequence numbers with the event handler or adapter that follows it.

5. Repeat Steps 3 and 4 until all event handlers and adapters have the appropriate sequence numbers.

6. Click **OK**.

The event handlers and adapters will now be triggered in the correct order for the execution schedule or schedules that you organized.

5.3.1.1.3 Removing an Event Handler or Adapter from a Data Object

To remove an event handler or adapter:

1. Select the desired event handler or adapter.

2. Click **Delete**.

The event handler or adapter is removed.

5.3.1.2 Map Adapters Tab

The Map Adapters tab becomes operational only after you assign a rule generator adapter or entity adapter to the data object.

You use this tab to map the variables of a rule generator or entity adapter to their proper source and target locations. For example, suppose the `adpSOLARISUSERIDGENERATOR` adapter has three variables: `firstname`, `Adapter` return value, and `lastname`. If a `Y` is displayed in the Mapped column for each adapter variable, this signifies that all three variables are mapped to the correct locations, and the adapter's status will change to Ready.

Note: An adapter can have any one of the following three statuses:

- **Ready:** This adapter has successfully compiled, and all of its variables are mapped correctly.
 - **Mapping Incomplete:** This adapter has successfully compiled, but at least one of its variables has been not mapped correctly.
 - **Mapping Incomplete:** This adapter has successfully compiled, but at least one of its variables has not been mapped correctly.
-
-

For more information about compiling adapters and mapping its variables, see [Chapter 2, "Developing Adapters"](#).

Note: If no adapters are assigned to a data object, the Map Adapters tab is grayed out.

5.4 Reconciliation Rules Form

This form is located in the Development Tools folder.

Figure 5–3 Reconciliation Rules Form

You use this form to define rules that are invoked at the following times:

- When Oracle Identity Manager tries to determine which user or organization record is associated with a change on a trusted source. These rules are evaluated as soon as all required fields in the reconciliation event are processed on the Reconciliation Data tab of the Reconciliation Manager form.
- When Oracle Identity Manager attempts to determine which user or organization record is the owner of an account discovered on a target resource, for example, as a result of a change detected on that system. These rules are evaluated only when all required fields in the reconciliation event are processed on the Reconciliation Data tab of the Reconciliation Manager form, and no processes were matched to the event on the Processes Matched Tree tab of the same form.

As mentioned, rules defined by using this form are used to match either users or organizations associated with a change on a trusted source or target resource. Rules of these types are referred to as user-matching or organization-matching rules, respectively. These rules are similar to the ones you can define by using the Rule Designer form except that the rules created by using the Reconciliation Rules form are specific to the resource object (because they relate to a single target resource) and only affect reconciliation-related functions.

Topics in working with reconciliation rules include:

- [Defining a Reconciliation Rule](#)
- [Adding a Rule Element](#)
- [Nesting a Rule Within a Rule](#)
- [Deleting a Rule Element or Rule](#)

5.4.1 Defining a Reconciliation Rule

The following procedure describes how to define a reconciliation rule.

Note: In the following procedure, you must ensure that the **Active** check box is selected. If this check box is not selected, the rule will not be evaluated by Oracle Identity Manager's reconciliation engine when processing reconciliation events related to the resource. However, you can only select this check box after Oracle Identity Manager has selected the **Valid** system check box. The **Valid** check box can only be selected after you have created at least one rule element, and Oracle Identity Manager has determined that the logic of this rule element is valid.

To define reconciliation rules for user or organization matching:

1. Go to the Reconciliation Rules form.
2. Enter a name for the rule in the **Name** field.
3. Select the target resource with which this rule is to be associated in the **Object** field
4. Enter a description for the rule in the **Description** field.

Select the **And** or **Or** operator for the rule. If **And** is selected, all elements (and rules if they are nested) of the rule must be satisfied for the rule to be evaluated to true. If **Or** is selected, the rule will be evaluated to true if any element (or rule if one has been nested) of the rule is satisfied.

5. Click **Save**.

The rule definition will be saved. Rule elements must now be created for the rule.

5.4.2 Adding a Rule Element

To define individual elements in a reconciliation rule:

1. Go to the Rule definition to which you want to add elements.
2. Click **Add Rule Element** on the **Rule Elements** tab.
The Add Rule Element dialog box is displayed.
3. Click the **Rule Element** tab.
4. Select a user-related data item from the **User Data** menu.

This will be the user data element that Oracle Identity Manager examines when evaluating the rule element. The menu will display all fields on the Oracle Users form (including any user-defined fields you have created).

Note: If the rule being defined is for organization matching, both the data available and the name of the menus will be related to organizations, rather than users.

5. Select an operator from the **Operator** menu.

This will be the criteria that Oracle Identity Manager applies to the attribute for data item you selected when evaluating the rule element. The following are valid operators:

- **Equals:** If you select this option, the user or organization record's data element must exactly match the attribute you select.

Note:

- If you configure trusted source reconciliation of users, you must ensure that the `User ID` field of the Oracle Identity Manager User account is used in the reconciliation matching rule.
 - If you configure trusted source reconciliation of organizations, you must ensure that the `Organization Name` field of the Oracle Identity Manager User account is used in the reconciliation matching rule.
-
-

- **Contains:** If you select this option, the user or organization record's data element must only contain (not be an exact match with) the attribute you select.
 - **Start with:** If you select this option, the user or organization record's data element must begin with the attribute you select.
 - **End with:** If you select this option, the user or organization record's data element must end with the attribute you select.
6. Select a value from the **Attribute** menu. The values in this menu are the fields that were defined on the Reconciliation Fields tab for the resource associated with the rule. If the reconciliation fields have not yet been designated for the resource, no values will be available.

Note: When defining a rule element for a target resource (as opposed to a trusted source), only fields associated with parent tables of the resource's custom process form are available for selection in the **Attribute** field.

7. If you want Oracle Identity Manager to perform a particular transformation on the data in the **Attribute** field (before applying the operator), select the desired transformation from the **Transform** menu.

Note: If you select a value other than None from this menu, after you click **Save**, you must also select the tab and set the appropriate properties so that Oracle Identity Manager is able to perform the transformation correctly.

The possible transformations are described in [Table 5-3](#).

Table 5-3 Transformation Properties

Transformation	Properties to Be Set on the Rule Element Properties tab
Substring	Start Point, End Point
Endstring	Start Point
Tokenize	Delimiters, Token Number, Space Delimiter

8. Select the **Case-Sensitive** check box.

For the rule element to be met, if this check box is selected, the value selected in the **Attribute** field must match the capitalization of the value being evaluated in

the reconciliation event record. If this check box is deselected, the value selected in the **Attribute** field is not required to match the capitalization used in the value being evaluated in the reconciliation event record.

9. Click **Save**.

10. If you select a value (other than None) in the **Transform** menu and have not yet set the properties for the transformation, the **Properties Set** check box will not be selected.

You must select the **Rule Element Properties** tab, set the appropriate properties, and click **Save** again.

The rule element will be added to the rule.

11. Repeat this entire procedure for each rule element you wish to add to the rule.

Note: Ensure that the **Active** check box is selected.

5.4.3 Nesting a Rule Within a Rule

You can nest an existing rule within a rule. Oracle Identity Manager evaluates the criteria of the nested rule in the same way as any other element of the rule.

Note: Only reconciliation-related rules that are associated with the same resource object are available for selection in the dialog box.

To nest a rule within a rule:

1. Go to the rule to which you want to add another rule.
2. Click **Add Rule** on the **Rule Elements** tab.
3. The Rule Choice lookup dialog box is displayed.

Locate and select the desired rule.

4. Click **OK**.

The selected reconciliation rule is added to rule.

5. Repeat steps 2 through 4 for each rule you want to nest in the rule.

5.4.4 Deleting a Rule Element or Rule

To delete a rule element or a rule:

1. Go to the rule from which you want to delete an element.
2. Select the rule element or rule to be deleted on the **Rule Elements** tab.
3. Click **Delete**.

Developing Scheduled Tasks

Oracle Identity Manager contains a set of predefined tasks that can be scheduled as job runs. An example is a password warning task that sends email to users for password expiration.

Oracle Identity Manager also provides the capability of creating your own scheduled tasks. You can create scheduled tasks according to your requirements if none of the predefined scheduled tasks fit your needs.

For example, you can configure a reconciliation run using a scheduled task that checks for new information on target systems periodically and replicates the data in Oracle Identity Manager.

This chapter explains how to create and implement your custom scheduled tasks. It contains these topics:

- [Overview of Task Creation](#)
- [Define the Metadata for the Scheduled Task](#)
- [Configure the Scheduled Task XML File](#)
- [Develop the Scheduled Task Class](#)
- [Configure the Plug-in XML File](#)
- [Create the Directory Structure for the Scheduled Task](#)

6.1 Overview of Task Creation

This section outlines the essential steps in creating scheduled tasks, and presents an example to illustrate the process. Subsequent sections provide details on each step.

- [Steps in Task Creation](#)
- [Example of Scheduled Task](#)

6.1.1 Steps in Task Creation

The basic steps for configuring new scheduled tasks are as follows:

1. Review Oracle Identity Manager's predefined scheduled tasks to determine whether a custom task is necessary.

For details about the predefined tasks, see "Managing Scheduled Tasks" in the *Oracle Fusion Middleware System Administrator's Guide for Oracle Identity Manager*.

2. Determine key features of the scheduled task, such as the task name and the parameters that control the actions performed by the task.

For details, see [Section 6.2, "Define the Metadata for the Scheduled Task"](#).

3. Add the task metadata to the scheduled task XML file.

For details, see [Section 6.3, "Configure the Scheduled Task XML File"](#).

4. Develop the scheduled task Java class.

For details, see [Section 6.4, "Develop the Scheduled Task Class"](#).

5. Declare the new scheduled task as a plug-in.

For details, see [Section 6.5, "Configure the Plug-in XML File"](#).

6. Package the task files so that Oracle Identity Manager can locate the files and make the task available for jobs.

For details, see [Section 6.6, "Create the Directory Structure for the Scheduled Task"](#).

6.1.2 Example of Scheduled Task

To illustrate the steps in developing a scheduled task, we use an example scheduled task that retrieves employee records belonging to the given department from a given IT resource.

In addition, our scheduled task should allow the user to specify the number of records to be retrieved and whether to include disabled records in the retrieval.

6.2 Define the Metadata for the Scheduled Task

Each scheduled task contains the following metadata information:

- Name of the scheduled task
- Name of the Java class that implements the scheduled task
- Description
- Retry Interval
- (Optional) Parameters that the scheduled task accepts. Each parameter contains the following additional information:
 - Parameter Name
 - Parameter Data Type
 - Required/ Optional Parameter
 - Help Text

6.3 Configure the Scheduled Task XML File

Configuring the scheduled task XML file involves updating the XML file that contains the definitions of custom scheduled tasks. This section describes how to update the task XML file with the details of the new custom scheduled task.

You can modify the task.xml file located in the /db namespace of Oracle Identity Manager MDS schema, or you can create a custom scheduled task file. If you create a custom file, then the file name must be the same as the scheduled task name, with the .xml extension. You must import the custom scheduled task file to the /db namespace of Oracle Identity Manager MDS schema.

See Also: [Chapter 7, "Developing Plug-ins"](#) for examples of plug-ins.

Note: The scheduled task XML file can be imported into MDS using an Oracle WebLogic Server import utility. In a clustered environment, having the file in MDS avoids the need to copy the file on each node of the cluster.

For details about importing files into MDS, see [Chapter 33, "MDS Utilities and User Modifiable Metadata Files"](#).

The elements in the XML file reflect the task parameters that you described in [Section 6.2, "Define the Metadata for the Scheduled Task"](#).

[Example 6–1](#) shows a sample XML code for the scheduled task described in the preceding paragraph. Note that all the parameters are declared to be required parameters in this example.

Example 6–1 Sample XML for a Scheduled Task

```
<scheduledTasks xmlns="http://xmlns.oracle.com/oim/scheduler">
  <task>
    <name>Test_scheduled_task</name>
    <class>oracle.iam.scheduler.TestScheduler</class>
    <description>Retrieve Employee Records For Given Department</description>
    <retry>5</retry>
    <parameters>
      <string-param required="true" helpText="Name of the
department">Department Name</string-param>
      <string-param required="true" encrypted="false" helpText="Name of the
department">Department Name</string-param>
      <number-param required="true" helpText="Number of Records to Be
Retrieved">Number of Records</number-param>
      <boolean-param required="false" helpText="Retrieve disabled employee
records?">Get Disabled Employees</boolean-param>
    </parameters>
  </task>
</scheduledTasks>
```

See Also: [Appendix A, "Scheduled Task Configuration File"](#) for details about the elements in the scheduled task configuration file.

This is basically exporting the task.xml from MDS and then adding the required tags to it and importing it back into MDS.

You must export the task.xml file from MDS, add the required tags to the file, and then import it back to MDS. See [Chapter 33, "MDS Utilities and User Modifiable Metadata Files"](#) for information about exporting and importing MDS files.

6.4 Develop the Scheduled Task Class

The next step is to create a Java class to execute the task whose metadata was defined in the XML file. The Java class that implements a scheduled task is known as a **scheduled task class**.

To develop a Java class for the scheduled task:

1. Create a Java class file that extends the `oracle.iam.scheduler.vo.TaskSupport` class and overrides the `execute()` method with processing logic based on your requirements.

2. Create a JAR file for the Java class that you created. Name the JAR such that you can readily associate this JAR with your custom scheduled task.

The JAR file can contain the dependent classes of the Java class. You can also create a separate JAR file for the dependent classes and place it in the lib/directory.

3. Copy the JAR file into the lib/ directory.
4. Repeat Steps 1 through 3 for every Java class that you want to create.

6.5 Configure the Plug-in XML File

You must configure the plugin.xml file in order to declare the scheduled task as a plug-in. See [Chapter 7, "Developing Plug-ins"](#) for more information about plug-ins.

Note: Oracle recommends creating one plugin.xml file for one scheduled task. This is because when the plugin is unregistered, the corresponding package is deleted.

To configure the plugin.xml file:

1. Create the plugin.xml file by using any text editor.

Note: Create the plugin.xml file only if no such file exists. If there are existing plugins, then add a new plugin element for the new plugin.

2. Specify the plug-in point for the scheduled task by changing the value of the pluginpoint attribute of the plugins element to oracle.iam.scheduler.vo.TaskSupport.

The following XML code block from the plugin.xml file shows the value entered within the plugins element:

```
<plugins pluginpoint="oracle.iam.scheduler.vo.TaskSupport">
```

Note: For scheduled tasks, the <plugins> element remains the same for all scheduled tasks.

3. Add a <plugin> element for each scheduled task that you are adding.

To specify the class that implements the plug-in (in this case, the scheduled task), change the value of the pluginclass attribute of the plugin element to the name of the Java class that implements the scheduled task. The following XML code block from the plugin.xml file shows sample values entered within the plugin element:

```
<plugin pluginclass="oracle.iam.scheduler.TestScheduler" version="1.0.1"
name="scheduler element"/>
```

After modification, the plugin.xml file looks similar to the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<oimplugins xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<plugins pluginpoint="oracle.iam.scheduler.vo.TaskSupport">
```

```

<plugin pluginclass= "oracle.iam.scheduler.TestScheduler"
version="1.0.1" name="scheduler element">
</plugin>
</plugins>
</oimplugins>

```

4. Save and close the plugin.xml file.

6.6 Create the Directory Structure for the Scheduled Task

The final step in configuring the scheduled task is to create a plugin.zip file with the directory structure specified in [Example 6-2](#). In this example, a single plug-in is being added, but there can be multiple pugins in the plugin.zip file. Scheduler requires that files be zipped in a particular structure and named according to a particular naming convention. This ensures that Oracle Identity Manager identifies the custom scheduled tasks and makes it available in Oracle Identity Manager Administrative and User Console while creating jobs

Example 6-2 Directory Structure for the Scheduled Task

```

plugin/
  plugin.xml
  lib/
    CLASS_NAME1.jar

```

Note that:

- The XML file for the plug-in must be named plugin.xml.
- The lib/ directory must contain only .JAR files. The lib/ directory consists of JAR files that contains the classes implementing the plug-in logic and the dependent library JAR files. In most instances, this directory consists of a single JAR file with the implementation of all the plug-ins that are specified in plugin.xml. See ["Developing Plug-ins"](#) on page 7-5 for for information about the directory structure.
- The directory for the scheduled task must contain the following files:
 - XML for the plug-in
 - JAR files
- There is one plugin.zip file for all the plug-ins that you create.

In the preceding example, *CLASS_NAME.JAR* is the JAR file that you create in [Section 6.4, "Develop the Scheduled Task Class"](#).

After you create the plugin.zip file, if deploying in a clustered environment, register the plug-in to the database by using appropriate APIs. See ["Registering and Unregistering Plug-ins By Using APIs"](#) on page 7-6 for details about registering plug-ins to Oracle Identity Manager by using APIs.

Note: The XML for the plug-in must be named plugin.xml. Ensure that the lib directory contains only JAR files.

Developing Plug-ins

The Service Framework in Oracle Identity Manager enables you to extend the platform by adding new services. Sometimes, the existing services can be extended or modified to suit new requirements. Oracle Identity Manager supports this requirement by means of the Plug-in Framework.

This chapter explains basic features of the plug-in framework and how to perform common tasks to customize your environment using plug-ins. The topics include:

- [Background of the Plug-in Framework](#)
- [Configuring Plug-ins](#)
- [Defining and Using Plug-ins](#)
- [Registering Plug-ins](#)
- [About Mapped Values](#)
- [Plug-in Points](#)

7.1 Background of the Plug-in Framework

A **plug-in** is software that extends the functionality of features provided by Oracle Identity Manager. The **plug-in framework** enables you to define, register, and configure plug-ins, which extend the functionality provided by features. Plug-ins can be predefined or custom-developed, and they are utilized at **plug-in points**.

A plug-in point is a specific point in the business logic where extensibility can be provided. An interface definition called the **plug-in interface** accompanies such a point. Users can extend the plug-in interface based on the business requirements and register them as plug-ins.

This section describes the plug-in framework in the following topics:

- [About the Plug-in Framework](#)
- [About Plug-in Stores](#)
- [Steps for Developing Plug-ins](#)

7.1.1 About the Plug-in Framework

The plug-in framework provides Oracle Identity Manager with a number of added capabilities. For example:

- Services can define plug-in points where functionality can be extended. For example, the notification engine uses a default provider to send e-mail

notifications. This provider is set up as a plug-in point so that a custom e-mail provider can be implemented.

- Services can discover all the plug-ins defined for a particular plug-in point.
- Plug-ins can be loaded from multiple sources. Feature developers do not need to be concerned about where the plug-ins are stored or with how they are loaded.
- Developers can change the plug-in code without needing to re-register the plug-in.
Note: This is possible only when the plug-in is stored on the filesystem.

7.1.2 About Plug-in Stores

The plug-in framework can store plug-ins in two types of stores:

- File system
- The Oracle Identity Manager database

When looking for plug-ins, the framework first examines plug-ins registered in the database, and looks in the file system.

7.1.2.1 File Store

The File Store consists of one or more directories on the Oracle Identity Manager host and is primarily used in development environments. This type of store is not appropriate for a production environment. File storage is convenient for the developer since there is no need to explicitly register the developed plug-ins with a file store. Users can just drop in the plug-in zips or exploded plug-in directory to the designated location(s).

By default, Plug-in framework looks for the plug-ins under the `OIM_HOME/plugins` directory. Additional plug-in directories can also be specified.

If a monitoring thread is enabled, then the plug-in framework monitors all the additions, modification, and deletions of plug-in zip files under the registered plug-in directories in the file system, and automatically reloads the plug-ins. Plug-in metadata such as name, version, and ID is read from the plug-in zip and is maintained in memory. This metadata is updated based on any file changes. The latest plug-in zip file is considered to be the current version of the plug-in.

For details about how to configure the file store, see [Section 7.2, "Configuring Plug-ins"](#).

7.1.2.2 Database Store

Plug-ins can be stored in the Oracle Identity Manager database, so that they are accessible from any node in a cluster. The Plug-in Framework uses Operation DB as the database store. This type of store is appropriate for a production environment.

You must explicitly register any plug-ins that are stored in the database. You can use the Plugin Registration Utility, which is a command-line tool, to register and deregister plug-ins. You can also use the `registerPlugin` API for this purpose. See [Section 7.4, "Registering Plug-ins"](#) for more information about registering plug-ins.

Note: After registering a plug-in, the server must be restarted. However, restarting the server might also depend on the feature that defines the plug-in point.

7.1.3 Steps for Developing Plug-ins

The basic aspects of developing and implementing plug-ins are as follows:

- Declare a plug-in instance where you may specify the type of value mapping for the plug-in point.
- Specify the properties that will comprise the plug-in metadata.
- Develop the plug-in code and distribute the plug-in.

For details, see [Section 7.3, "Defining and Using Plug-ins"](#).

7.2 Configuring Plug-ins

You use the oim-config.xml file in the MDS to configure the following:

See Also: "Configuring the oim-config.xml File" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager* for information about configuring the oim-config.xml file

- The directory or directories in which the files store will look for plug-ins.
- Whether to activate a thread that monitors the file store for any changes; the thread checks the zip files or exploded files in all the plug-in directories.

The monitoring thread is typically activated in a dynamic development environment since plug-ins are being added or modified in such an environment; it can be inactive in a production system which contains a set of plug-ins . This is tracked by the reloadingEnabled attribute.

- The time interval at which the monitoring thread wakes up and looks for any changes.

The following is a code snippet from the oim-config.xml file:

```
<pluginConfig storeType="common">
    <storeConfig reloadingEnabled="true"
        reloadingInterval="20">
        <!--
            Plugins present in the OIM_HOME/plugins directory are added by default.
            For adding more plugins, specify the plugin directory as below:
            <registeredDirs>/scratch/oimplugins</registeredDirs>
            <registeredDirs>/scratch/custom</registeredDirs>
        -->
    </storeConfig>
</pluginConfig>
```

In this example:

- The common store designation tells the framework to monitor both database and file stores

Note: Do not modify the `Store` value; `common` is appropriate in all environments.

- One directory is configured; additional directories can be configured by simply adding more `<registeredDirs>` tags.
- The monitoring thread is active and looks for plug-in changes every 20 seconds by default.

Monitoring is typically active in development environments only. If you switch between active and inactive, you must restart the application server for the change to take effect.

Note: Restarting the application server is required for any changes made to plug-in data in the `oim-config.xml` file.

7.3 Defining and Using Plug-ins

This section provides details about the key aspects of defining and setting up plug-ins:

Note: Although these topics are presented in separate sections, they are not necessarily sequential steps. Tasks such as declaring plug-in points, their plug-ins and the plug-in metadata can be performed together in a single step.

- [Declaring Plug-ins](#)
- [Specifying Plug-in Metadata](#)
- [Developing Plug-ins](#)

7.3.1 Declaring Plug-ins

To extend the functionality provided by Oracle Identity Manager, you can declare the plug-ins for the application.

A plug-in has a Java class that implements the plug-in point interface. Be sure to assign unique names to all the plug-ins associated with a specific plug-in point. If the plug-in names are non-unique, an exception will be thrown during plug-in registration.

Declare the plug-ins in the `plugin.xml` file. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<oimplugins>
  ....
  <plugins pluginpoint="oracle.iam.sample.passwdmgmt.service.PasswordElement">
    <plugin pluginclass=
      "oracle.iam.sample.passwdmgmt.custom.NumCustomPasswordElement"
      version="1.0.1" name="num pwd element"/>
    <plugin pluginclass=
      "oracle.iam.sample.passwdmgmt.custom.DictionaryPasswordElement"
      version="1.0.1" name="Dictionary password element" />
  </plugins>
  ....
</oimplugins>
```

Note: You can have multiple versions of the plug-in stored and the feature can request a specific version of the plug-in from the plug-in framework.

The XML shows two plug-in declarations. Both the plug-ins extend from the same plug-in point.

7.3.2 Specifying Plug-in Metadata

Along with each plug-in that is defined in `plugin.xml`, you can specify a list of properties that comprise the plug-in metadata. For an example of using the plug-in metadata, see ["Extending Request Management Operations"](#) on page 23-30.

In this example, the metadata consists of a single property known as `PasswordElementNum`, with the value 1:

```
<?xml version="1.0" encoding="UTF-8"?>
<oimplugins>
    ....
    <plugins pluginpoint="oracle.iam.sample.passwdmgmt.service.PasswordElement">
        <plugin pluginclass=
"oracle.iam.sample.passwdmgmt.custom.NumCustomPasswordElement"
            version="1.0.1" name="num pwd element">
            <metadata name="PasswordElementNum"><value>1</value></metadata>
        </plugin>
        <plugin pluginclass=
"oracle.iam.sample.passwdmgmt.custom.DictionaryPasswordElement"
            version="1.0.1" name="Dictionary password element" >
            <metadata name="PasswordElementNum"><value>2</value></metadata>
        </plugin>
    </plugins>
    ....
</oimplugins>
```

7.3.3 Developing Plug-ins

To develop a plug-in:

1. Identify the plug-in point to extend.
2. Identify the Java class that implements the plug-in point interface. Package the Java class and other dependent classes into a JAR file. Put the JAR file in the `lib/` directory.
3. Create the `plugin.xml` file. See ["Declaring Plug-ins"](#) on page 7-4 for details.
4. Identify the resource files required by the plug-in, such as property files, resource bundles, and image files.
5. Zip the entire package.

An Oracle Identity Manager plug-in is distributed as a ZIP file with a specified directory structure. The directory structure is as follows:

- **The `plugin.xml` file:** The XML file contains the metadata associated with all the plug-ins such as the plug-in point it extends, the class implementing the plug-in, name, and the version number. All the fields in the XML are

mandatory except the name. If the name is not given, then plugin class name is used as the name.

- **The lib/ directory:** The lib/ directory consists of JAR files that contains the classes implementing the plug-in logic and the dependent library JAR files. In most instances, this directory consists of a single JAR file with the implementation of all the plug-ins that are specified in plugin.xml.
- **The resources/ directory:** Contains resource files required by the plug-in, such as property files, resource bundles, and image files. These resources given in the resources directory of the plug-in zip can be accessed as follows:

```
this.getClass().getClassLoader().getResourceAsStream(<resource_name>);
```

Multiple plug-ins implementing the same plug-in point can be part of the same ZIP file.

A plug-in has a Java class that implements the plug-in point interface. The plug-in library (JAR) can contain other dependent classes as well, but the class implementing the plug-in is the only one that is exposed to the feature. This class must be specified in plugin.xml.

6. Place the ZIP file in the file store (lib/ directory) or database store. See ["About Plug-in Stores"](#) on page 7-2 for details.
7. If the ZIP is placed in the database store, then register the plug-in by using the Plug-in Registration Utility, as described in ["Registering Plug-ins"](#) on page 7-6.

7.4 Registering Plug-ins

You can register the plug-ins by using APIs and Plugin Registration Utility.

- [Registering and Unregistering Plug-ins By Using APIs](#)
- [Registering and Unregistering Plug-ins By Using the Plugin Registration Utility](#)

7.4.1 Registering and Unregistering Plug-ins By Using APIs

You can use the following APIs for registration-related tasks:

- PlatformService.registerPlugin
- PlatformService.unregisterPlugin

Here is an example:

```
ClientPlatform platform = OIMClient.getInstance();
platform.login("username", "password");
PlatformService service = platform.getService(PlatformService.class);
File zipFile = new File(fileName);
FileInputStream fis = new FileInputStream(zipFile);
int size = (int) zipFile.length();
byte[] b = new byte[size];
int bytesRead = fis.read(b, 0, size);
while (bytesRead < size) {
    bytesRead += fis.read(b, bytesRead, size - bytesRead);
}
fis.close();
service.registerPlugin(b);
service.unregisterPlugin(pluginID, version);
```

7.4.2 Registering and Unregistering Plug-ins By Using the Plugin Registration Utility

You can use the Plugin Registration Utility for registering and unregistering plug-ins. The utility uses the following files:

- pluginregistration.xml
- ant.properties

These files are located in the *OIM_HOME/plugin_utility/* directory.

Note: Plug-in registration utilities require Apache Ant version 1.7 or later.

Before using the utility, perform the following:

1. Set the values for *WLS_HOME* and *OIM_HOME* in ant.properties.

For example:

```
WLS_HOME = ../middleware/wlserver_10.3
OIM_HOME = ../middleware/Oracle_IDM1/server
```

2. Build the wfullclient.jar in Oracle WebLogic server:

- a. Change directories to *WLS_HOME/server/lib*.
- b. Run the following command:

```
java -jar ../../../../modules/com.bea.core.jarbuilder_1.3.0.0.jar
```

Note: The exact JAR file version can be different based on the WLS. Use the corresponding file with the name as *com.bea.core.jarbuilder* at the *WLS_HOME/../../modules/* directory.

Registering a Plug-in

To register a plug-in:

1. Execute the ant target "register":

```
ant -f pluginregistration.xml register
```

2. This will prompt for the Oracle Identity Manager username and password along with the server information and the location of the plugin zip file. Enter the complete path of the zip file location.

Unregister a Plug-in

To unregister a plug-in:

1. Execute the ant target "unregister":

```
ant -f pluginregistration.xml unregister
```

2. This will prompt for the Oracle Identity Manager username and password along with the server information and the classname of the plug-in class. Enter the classname with the complete package.

7.5 About Mapped Values

You can use mapped values to setup and provide contextual information on the environment. With simple mapped values, a hashed table is built in which you can store a one-to-one mapping of values.

The plug-in framework uses the following logic to determine a simple mapped value:

- If the attribute has a mapping specified for a given method, that mapping is used.
- If the attribute uses the same mapping for all methods, that mapping is always used.
- Otherwise, if no mapping is specified, a `NoSuchMappingException` is thrown.
- If the mapping's attribute is declared, it is used as the key to look up the value in the provided `HashMap`.
- If the mapping's attribute is not declared but the mapping's value is declared, that value is returned.

Here is an example that shows how the hash map is built and used:

```
public boolean evaluate(String password) {
    Map<String, Object> ro = new HashMap<String, Object>();
    ro.put("smvalue1", "value");
    ro.put("smattr2", "value2");
    // Using the mappings declared in Example 22-1, attr1 would
    // have the value of "smvalue1".
    String attr1 = PluginFramework.getMappedValue(this, ro,
        "attr1", "evaluate");
    // Using the mappings declared in Example 22-1, attr1 would
    // have the value of "value2"
    String attr2 = PluginFramework.getMappedValue(this, ro, "attr2", "evaluate");
}
```

Here, Example 22-1 refers to an example `plugin.xml`, as shown in [Example 7-1](#):

Example 7-1 Example Plugin.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<oimplugininstances>
  <plugininstances name="CustomPasswordElementInstance" plugin="num pwd element"
    pluginpoint="oracle.iam.sample.passwdmgmt.service.PasswordElement"
    version="1.0.1">
    <metadata name="meta1">
      <value>1</value>
    </metadata>
    <mapping method="evaluate">
      <simple-mapping name="attr1" entity-type="User" value="smvalue1" />
      <runtime-mapping name="phone" entity-type="User" attribute="phonenum"
value="N/A" />
    </mapping>
    <mapping>
      <simple-mapping name="attr1" entity-type="User" value="value1" />
      <simple-mapping name="attr2" entity-type="User" attribute="smattr2" />
      <runtime-mapping name="fname" entity-type="User" attribute="firstname" />
      <runtime-mapping name="lname" entity-type="User" attribute="lastname" />
    </mapping>
    <description>Test class for plugin mapped value</description>
  </plugininstances>
</oimplugininstances>
```

7.5.1 Accessing Mapped Values

The plug-in framework uses the following logic to determine a simple mapped value:

- If the given attribute name has a mapping specified for the given method, then that mapping is used. Otherwise, if the given attribute name has a mapping specified for all methods, then that mapping is used. Otherwise, `NoSuchMappingException` is generated.
- If the attribute for the mapping is declared, then the attribute is used as the key to lookup the value in the provided `HashMap`.
- If the attribute for the mapping is not declared but the value of the mapping is declared, then that value is returned. For example:

```
Map<String, Object> ro = new HashMap<String, Object>();
ro.put("smvalue1", "value");
ro.put("smattr2", "value2");
String attr1 = PluginFramework.getMappedValue(this, ro, "attr1", "evaluate");
String attr2 = PluginFramework.getMappedValue(this, ro, "attr2", "evaluate");
```

7.6 Plug-in Points

[Table 7-1](#) lists the Java interfaces that act as plug-in points in Oracle Identity Manager:

Table 7-1 Plug-in Points

Plug-in Point	Description
oracle.iam.ldapsync.LDAPContainerMapper	This is used by LDAP synchronization to determine which user/role container should be used to create the user/role in LDAP.
oracle.iam.platform.kernel.spi.EventHandler	This is the kernel event handler. See Chapter 8, "Developing Event Handlers for Extending User Management Operations" for information about kernel event handlers.
oracle.iam.platform.auth.api.LoginMapper	<p>This is an implementation of a LoginMapper maps the JAAS user principal name to the corresponding Oracle Identity Manager username. This plug-in point is used to override the default mapping of JAAS user principal name to Oracle Identity Manager username for SSO scenarios. The default implementation returns the same value as the JAAS user principal name.</p> <p>This plug-in point is typically used in SSO scenarios where the JAAS user principal name and the Oracle Identity Manager username might be different. For example, the SSO system might set the email as the JAAS username but no user with that username exist in Oracle Identity Manager. For Oracle Identity Manager to recognize that user, the JAAS user principal name must be mapped to the Oracle Identity Manager username. This can be done by implementing a plug-in for LoginMapper, as shown:</p> <pre> public class CustomLoginMapper implements LoginMapper{ public String getOIMUserID(String jaasPrincipal) throws MappingException { return getUserName(jaasPrincipal); } private String getUserName(String emailID){ String userName = null; //Use usermgmt APIs to get the username corresponding to this email id return userName; } } </pre>
oracle.iam.identity.usermgmt.api.PasswordVerifier	This is used for verification of old password while changing the user's password. The class that is to be used for this validation is configured in the OIM.OldPasswordValidator system property. By default, use the container based authentication for verifying old password.
oracle.iam.request.plugins.StatusChangeEvent	This allows running of custom code during request status change.
oracle.iam.request.plugins.RequestDataValidator	This is used for custom validation of request data after submission.
oracle.iam.request.plugins.PrePopulationAdapter	This is used to prepopulate an attribute value by running custom code during request creation.

Table 7-1 (Cont.) Plug-in Points

Plug-in Point	Description
oracle.iam.scheduler.vo.TaskSupport	This is used to run the job in context. Execute method of the task is retrieved through the plug-in and is loaded.
oracle.iam.identity.usermgmt.api.UserNamePolicy	This is an implementation of username policies that are used to generate/validate username.
oracle.iam.identity.usermgmt.api.ReservationInLDAP	This is an implementation for reservation of user attributes in LDAP.

Developing Event Handlers for Extending User Management Operations

This chapter describes the asynchronous implementation of the post-processing functions involved in user management operations. It contains the following topics:

- [An Overview of User Management Operations](#)
- [Extending User Management Operations with Event Handlers](#)
- [Troubleshooting an Event Handler](#)

8.1 An Overview of User Management Operations

In an Identity Management system, any action performed by a user or system is called an operation. Examples of operations are creating users and updating users. Each operation goes through pre- and post-processing stages.

What happens at each stage is determined by branching and by the event handler, if any, that is deployed at that stage. If a stage has a branch, responses from the event handlers decide which branch to take. If a stage has no event handlers, or event handlers respond with no recommendation, the operation simply follows the default path and moves to the next stage.

Each operation performed in an identity management environment can have consequences for users or other entities. For example, creating a user might result in provisioning of resources to that user, updating the history results in changes to the reporting tables, and creating a new password policy might make certain user passwords invalid and require changes during next login.

Operations specific to a user, such as creation, modification, deletion, enable, disable, and so on are referred to as *user management operations*. The lifecycle of an operation consists of these stages:

- **validation:** Stage to perform validation on the orchestration, such as validity of orchestration parameters. Orchestration parameter is the data that is required to carry out the orchestration operation.
- **pre-processing:** Stage to perform orchestration parameter manipulations or get approvals or perform Segregation of Duties (SoD) checks.
- **audit:** Stage in which the auditing of operation is performed.
- **action:** Stage in which the action takes place.
- **post-processing:** Stage in which consequent operations related to the current operation takes place. Examples of consequent operations are auto role membership and policy evaluation on a user creation.

- **compensation:** Process is moved to this stage if the operation is rolled back by calling the `compensate` method.
- **finalization:** Last stage in the process to perform any clean up.

You can customize the consequences of user management operations such as create, update, delete, enable, disable, lock, unlock, and change password - also referred to as the post-processing functions of user management operations - by writing event handlers.

Tip: See ["Writing Custom Event Handlers"](#) on page 8-4 for details about how to write event handlers.

in 11g Release 1 (11.1.1), Oracle Identity Manager supports asynchronous execution of post-processing functions associated with user management operations. This significantly improves the perceived performance of user management operations.

Post-processing functions associated with user management operations can be triggered programmatically on a given set of users. This capability becomes useful in situations where the users are managed directly on the data store, specifically in a reconciliation or bulk load scenario.

8.2 Extending User Management Operations with Event Handlers

The 11g Release 1 (11.1.1) kernel exposes Service Provider Interfaces (SPIs) that are implemented to customize the functionality of user management operations. Currently, you can customize only pre-process, post-process, and validation stages of an operation for an entity.

Note: Customizations of pre-process and validation functions are synchronous with the operation. However, customizations of the post-process functions are asynchronous.

This section describes extending user management operations with event handlers in the following topics:

- [Understanding Elements in Event Handlers XML Files](#)
- [Writing Custom Event Handlers](#)

8.2.1 Understanding Elements in Event Handlers XML Files

Event Handlers XML files are comprised of elements and element attributes. This section describes some of the elements and element attributes within Event Handlers XML files. It also describes a mandatory namespace for the event handler XML definitions.

Elements

The top-level (or parent) element in Event Handlers XML files is `eventhandlers`. [Table 8-1](#) lists and describes sub-elements that are typically defined within the `eventhandlers` parent element.

Table 8–1 Typical Sub-elements within the eventhandlers Element

Sub-element	Description
validation-handler	Identifies the validations that will be performed on the orchestration.
action-handler	Identifies the operations that will be performed at preprocess, postprocess, and action stages.
failed-handler	Identifies the event handlers that will be executed if an event handler fails.
finalization-handler	Identifies the event handlers to execute at the end of the orchestration. Finalization is the last stage of any orchestration.
change-failed	Identifies event handlers to be executed upon consequence orchestration failures.
out-of-band-handler	<p>Defines event handlers for out-of-the-band orchestration flows such as veto and cancel. A process can move to some out-of-the-band stages if the event handlers are invalid or canceled. These stages are:</p> <ul style="list-style-type: none"> ■ Invalid: Process is moved to this stage if orchestration validation fails. ■ Veto: Process is moved to this stage if any of the preprocess event handlers are vetoed. For example, if approvals are rejected by the approver, then orchestration is vetoed. ■ Cancel: Process is moved to this stage if the operation is stopped by calling the cancel method. ■ Compensation: Process is moved to this stage if the operation is rolled back by calling the compensate method.
compensate-handler	Identifies the event handlers that will be executed in the compensation flow of the orchestration.

Element Attributes

The elements within Event Handlers XML files contain attributes. [Table 8–2](#) lists and describes attributes that are typically defined within elements.

Table 8–2 Typical Attributes of Sub-elements within the eventhandlers Element

Element Attribute	Description
Name	The name of the event handler.
class	Full package name of the Java class that implements the event handler.
entity-type	Identifies the type of entity the event handler is executed on. A value of ANY sets the event handler to execute on any entity.
operation	Identifies the type of operation the event handler is executed on. A value of ANY sets the event handler to execute on any operation.
order	Identifies the order (or sequence) in which the event handler is executed. Supported values are FIRST, LAST, or a numeral.

Table 8–2 (Cont.) Typical Attributes of Sub-elements within the eventhandlers Element

Element Attribute	Description
orch-target	<p>Identifies the type of orchestration, such as entity orchestration, Toplink orchestration, and so on. The following is a list of supported values:</p> <ul style="list-style-type: none"> ■ oracle.iam.platform.kernel.vo.EntityOrchestration ■ oracle.iam.platform.kernel.vo.MDSOrchestration ■ oracle.iam.platform.kernel.vo.RelationOrchestration ■ oracle.iam.platform.kernel.vo.ToplinkOrchestration <p>The default value is oracle.iam.platform.kernel.vo.EntityOrchestration.</p>
sync	<p>This attribute is operational in only the action-handler and change-failed elements. The sync attribute indicates whether the event handler is synchronous or asynchronous. Supported values are TRUE or FALSE. If set to TRUE (synchronous), the kernel expects the event handler to return an EventResult. If set to FALSE (asynchronous), you must return null as the event result and notify the kernel about the event result later.</p> <p>Note: The sync attribute must be set to TRUE for validation-handler elements.</p>
stage	<p>This attribute is operational in only the out-of-band-handler, action-handler, and failed-handler elements. The stage attribute indicates the stage at which the event handler gets executed. The following is a list of supported values:</p> <ul style="list-style-type: none"> ■ preprocess ■ action ■ audit ■ postprocess ■ veto ■ cancelled
tx	<p>This attribute is operational in only the out-of-band-handler, action-handler, compensate-handler, and finalization-handler elements. The tx attribute indicates whether or not the event handler should run in its own transaction. Supported values are TRUE or FALSE.</p>

Namespace Requirement in <eventhandlers> Element

Example 8–1 is the namespace that MUST be defined in the <eventhandlers> root element for all custom event handler XML definitions.

Example 8–1 Mandatory Namespace Definition for Custom Event Handlers

```
<eventhandlers xmlns="http://www.oracle.com/schema/oim/platform/kernel"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.oracle.com/schema/oim/platform/kernel
  orchestration-handlers.xsd">
```

8.2.2 Writing Custom Event Handlers

All customizations to user management operations need to be implemented as event handlers.

Perform the following steps to write a custom event handler:

1. Implement the custom event handler:

See [Section 8.2.2.1, "Implementing Custom Event Handlers"](#)

2. Create a plug-in containing the custom event handler:
See [Section 8.2.2.2, "Creating Plug-ins for Custom Event Handlers"](#)
3. Define custom events:
See [Section 8.2.2.3, "Defining Custom Events"](#)

8.2.2.1 Implementing Custom Event Handlers

To implement custom event handlers:

1. Implement one of the SPIs mentioned in [Table 8–3](#) to write a custom pre-process, post-process, or validation handler.

Table 8–3 SPIs to Write Custom Event Handlers

Stage	SPI to implement
Pre-Process	<code>oracle.iam.platform.kernel.spi.PreProcessHandler</code>
Post-Process	<code>oracle.iam.platform.kernel.spi.PostProcessHandler</code>
Validation	<code>oracle.iam.platform.kernel.spi.ValidationHandler</code>

2. Include the following JAR files in the class path to compile a custom class:

From `OIM_INSTALL_HOME/server/platform`

- `iam-platform-kernel.jar`
- `iam-platform-utils.jar`
- `iam-platform-context.jar`
- `iam-plaftorm-authz-service.jar`

From `OIM_INSTALL_HOME/designconsole/lib`

- `oimclient.jar`
- `xlAPI.jar`

From `OIM_INSTALL_HOME/designconsole/lib` and `OIM_INSTALL_HOME/server`:

All other JAR files

3. Create a library of JAR files containing the custom classes.

The following code samples illustrate how to invoke Oracle Identity Manager 9.1.x APIs and 11g APIs to customize user management operations. See SPI Javadocs for more information.

Example 1: Custom Password Validation

[Example 8–2](#) shows a sample custom validation handler code fragment that checks to ensure that \$ is not used in a password.

Example 8–2 Sample Custom Validation Handler

```
throws ValidationException, ValidationFailedException {
    HashMap<String, Serializable> parameters = orchestration.getParameters();
    String password = (parameters.get("usr_password") instanceof ContextAware)
        ? (String) ((ContextAware) parameters.get("usr_password")).getObjectValue()
        : (String) parameters.get("usr_password");
```

```

        if (password.contains("$")) {
            throw new ValidationFailedException();
        }
    }
}

```

Example 2: Custom Pre-process Event Handler to Set Middle Initial

[Example 8-3](#) shows a sample custom pre process event handler code fragment that sets the middle initial to the first letter of the first name if the user does not have a middle name.

Note: Bulk reconciliation does not execute preprocess event handlers.

Example 8-3 Sample Custom Pre Process Event Handler

```

// This custom preprocess event handler sets the first letter of the first name as
// the middle initial when the user doesn't have a middle name
public EventResult execute
    (long processId, long eventId, Orchestration orchestration) {
    HashMap<String, Serializable> parameters = orchestration.getParameters();
    // If the middle name is empty set the first letter of the first name
    // as the middle initial
    String middleName = getParamaterValue(parameters, "Middle Name");
    if (isNullOrEmpty(middleName)) {
        String firstName = getParamaterValue(parameters, "First Name");
        middleName = firstName.substring(0,1);
        orchestration.addParameter("Middle Name", middleName);
    }
    return new EventResult();
}
private String getParamaterValue(HashMap<String, Serializable> parameters,
    String key) {
    String value = (parameters.get(key) instanceof ContextAware)
        ? (String) ((ContextAware) parameters.get(key)).getObjectValue()
        : (String) parameters.get(key);
    return value;
}

```

Example 3: Custom Post-process Event Handler to Provision Resource Object

[Example 8-4](#) shows a sample custom post process event handler code fragment that provisions a resource object OBJ005 to a user whose role is ROLE 00.5:

Example 8-4 Sample Custom Post Process Event Handler

```

// This custom post process event handler provisions resource object 'OBJ005'
// to a user who has role 'ROLE 005'
public EventResult execute(long processId, long eventId,
    Orchestration orchestration) {
    tcUserOperationsIntf userOperationsService =
        Platform.getService(tcUserOperationsIntf.class);
    try {
        String userKey = getUserKey(processId, orchestration);
        if (hasRole(userKey, "ROLE 005")) {
            long objKey = findObject("OBJ001");
            userOperationsService.provisionResource(Long.getLong(userKey), objKey);
        }
    }
}

```



```

    } catch (Exception e) {
    throw new EventFailedException("Error occurred ", e);
    }

    return new EventResult();
    }

    // This method retrieves the key of the user entity on which an operation
    // is performed
    // This method shows how to retrieve the operation being performed, entity type
    // and the associated value objects
    private String getUserKey (long processID, Orchestration orchestration) {
        String userKey;
        String entityType = orchestration.getTarget().getType();
        EventResult result = new EventResult();

        if (!orchestration.getOperation().equals("CREATE")) {
            userKey = orchestration.getTarget().getEntityId();
        } else {
            OrchestrationEngine orchEngine = Platform.getService(OrchestrationEngine.class);
            userKey = (String) orchEngine.getActionResult(processID);
        }
        return userKey;
    }

    // This method checks if a given user has a given role.
    // It demonstrates how to invoke a OIM 11g API from a custom event handler
    private boolean hasRole(String userKey, String roleName)
        throws Exception {
        RoleManager roleManager = Platform.getService(RoleManager.class);
        List<Identity> roles = roleManager.getUserMemberships(userKey);

        for (Iterator iterator = roles.iterator(); iterator.hasNext();) {
            Role role = (Role) iterator.next();
            if (roleName.equals((String)role.getAttribute("Role Name"))) {
                return true;
            }
        }
        return false;
    }

    // This method finds details about a resource object with the given name.
    // It demonstrates how to invoke a 9.1.x API from a custom event handler
    private long findObject(String objName) throws Exception {
        long objKey = 0;
        tcObjectOperationsIntf objectOperationsService =
            Platform.getService(tcObjectOperationsIntf.class);
        HashMap params = new HashMap();
        params.put("Objects.Name", objName);
        tcResultSet objects = objectOperationsService.findObjects(params);
        for (int i = 0; i < objects.getRowCount(); i++) {
            objects.goToRow(i);
            if (objects.getStringValue("Objects.Name").equals(objName)) {
                objKey = objects.getLongValue("Objects.Key");
            }
        }
        return objKey;
    }

```

8.2.2.2 Creating Plug-ins for Custom Event Handlers

To create plug-ins containing custom event handlers, you need to develop the appropriate event handler classes. See [Chapter 7, "Developing Plug-ins"](#) for details.

Note: Ensure that plug-in point used in the plug-in definition is set to `oracle.iam.platform.kernel.spi.EventHandler`.

Note: The plug-ins can be packaged as required, just like the JAR files, as long as they adhere to the packaging guidelines.

Here is an example of a plug-in XML file:

```
<?xml version="1.0" encoding="UTF-8"?>
<implugins>
  <plugins pluginpoint="oracle.iam.platform.kernel.spi.EventHandler">
    <plugin pluginclass=
      "oracle.oim.extensions.preprocess.SamplePreprocessExtension"
      version="1.0"
      name="SamplePreprocessExtension">
    </plugin>
    <plugin pluginclass=
      "oracle.oim.extensions.postprocess.SamplePostprocessExtension"
      version="1.0"
      name="SamplePostprocessExtension">
    </plugin>
    <plugin pluginclass=
      "oracle.oim.extensions.validation.SampleValidationExtension"
      version="1.0"
      name="SampleValidationExtension">
    </plugin>
  </plugins>
</implugins>
```

8.2.2.3 Defining Custom Events

Take these steps to define custom events:

1. Create the metadata XML file containing definitions of all the custom events.

[Example 8-5](#) shows what a metadata file looks like:

Example 8-5 Sample Metadata XML File for Custom Event Definitions

```
<?xml version='1.0' encoding='utf-8'?>
<eventhandlers>
  <!-- Custom preprocess event handlers -->
  <action-handler
    class="oracle.oim.extensions.preprocess.SamplePreprocessExtension"
    entity-type="User"
    operation="CREATE"
    name="SamplePreprocessExtension"
    stage="preprocess"
    order="1000"
    sync="TRUE" />

  <!-- Custom postprocess event handlers -->
```

```

<action-handler
  class="oracle.oim.extensions.postprocess.SamplePostprocessExtension"
  entity-type="User"
  operation="CREATE"
  name="SamplePostprocessExtension"
  stage="postprocess"
  order="1000"
  sync="TRUE" />

<action-handler
  class="oracle.oim.extensions.postprocess.SamplePostprocessExtension"
  entity-type="User"
  operation="MODIFY"
  name="SamplePostprocessExtension"
  stage="postprocess"
  order="1000"
  sync="TRUE" />

<!-- Custom validation event handlers -->
<validation-handler
  class="oracle.oim.extensions.validation.SampleValidationExtension"
  entity-type="User"
  operation="CREATE"
  name="SampleValidationExtension"
  order="1000" />

<validation-handler
  class="oracle.oim.extensions.validation.SampleValidationExtension"
  entity-type="User"
  operation="MODIFY"
  name="SampleValidationExtension"
  order="1000" />
</eventhandlers>

```

2. Import these event definitions into MDS. See [Chapter 33, "MDS Utilities and User Modifiable Metadata Files"](#) for more information. For a shiphome-based install the scripts necessary to import the event definitions are located in the following directory:

OIM_HOME/common/wlst

Note: You need not define all the custom event definitions in the same file. They could be split across multiple new event definition files or they could be added to any existing event definition file. Exporting the desired event definition file, modifying it, and importing it back implements the latter. For information about exporting and importing event definition files, see [Chapter 33, "MDS Utilities and User Modifiable Metadata Files"](#).

8.3 Troubleshooting an Event Handler

The following list contains information to help troubleshoot your event handler if it appears not to start. Common causes of this might be:

- `EventHandlers.xml` has a syntax error.
- `EventHandlers.xml` was imported into the wrong place in MDS. It must be in a top-level directory recognized by Oracle Identity Manager; for example, `/db/`,

`/file/` or `/custom/` `/mystuff/`(for example) and the root directory (`/`) are not recognized.

- A preprocess event handler has been implemented using reconciliation. Preprocess event handlers do not start on reconciliation.
- A preprocess event handler has been implemented using reconciliation but not bulk orchestration execute. Both bulk and non-bulk methods must be implemented to work with reconciliation.

Configuring LDAP Container Rules

In earlier releases of Oracle Identity Manager, role name (UGP.UGP_NAME in the database) is unique. This is a limitation because a lot of roles can exist in large enterprises, and as a result, it is possible that administrators need to create two or more roles in Oracle Identity Manager with the same name but for different purpose.

Oracle Identity Manager can be installed with LDAP synchronization enabled. When roles are coming from LDAP via reconciliation, it is possible that two or more roles have the same name. LDAP supports two roles with the same name if the roles are located under two different Organization Units (OUs).

In Oracle Identity Manager 11g Release 1 (11.1.1), namespace is introduced to handle two roles with the same name. Roles with the same name are supported if the roles are in different namespaces. However, two or more roles with the same name in the same namespace is not supported.

When LDAP is integrated with Oracle Identity Manager, the namespace maps to an OU. By the default configuration, there is only one default namespace called Default, and therefore, role names are unique. To configure multiple namespaces, you must create an XML file called LDAPContainerRules.xml and load it in the metadata store (MDS). The LDAPContainerRules.xml also specifies the namespace of a role based on the role attributes.

When LDAP synchronization is enabled, and a user is to be created, then a plug-in determines in which container the user is to be created. Similarly, if a role is to be created, then this plug-in determines the container in which the role is to be created. For this, Oracle Identity Manager calls a plug-in that implements the `oracle.iam.ldapsync.LDAPContainerMapper` interface. All the attributes of the user/role are passed to the plug-in, and it returns the Domain Name (DN) of the LDAP container. You can write your own plug-in, register the plug-in to Oracle Identity Manager, and then configure Oracle Identity Manager to use the plug-in by setting the `LDAPContainerMapperPlugin` system property. See "System Properties in Oracle Identity Manager" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager* for information about this system property.

Oracle Identity Manager provides a default plug-in for determining the LDAP container for user/role based on user or role attributes that are synchronized to LDAP. The default plug-in reads the rules from a XML file to determine the LDAP container. The XML file must be deployed to MDS as `/db/LDAPContainerRules.xml`. When Oracle Identity Manager is installed with LDAP synchronization enabled, the installer asks for user and role container values. These values are stored in the `/db/LDAPContainerRules.xml` file at containers for which the expression is Default. The following is an example:

```
<container-rules>
  <user>
```

```

<rule>
  <expression>Country=US, Locality Name=AMER</expression>
  <container>l=amer,dc=oracle,dc=com</container>
</rule>
<rule>
  <expression >Country=IN, Locality Name=APAC</expression>
  <container>l=apac,dc=oracle,dc=com</container>
</rule>
<rule>
  <expression>Default</expression>
  <container>l=users,dc=oracle,dc=com</container>
</rule>
</user>
<role>
  <rule>
    <expression>Role Description=AMER</expression>
    <description>AMER</description>
    <container>l=amer,ou=role,dc=oracle,dc=com</container>
  </rule>
  <rule>
    <expression >Role Description=APAC</expression >
    <description>APAC</description>
    <container>l=apac,ou=role,dc=oracle,dc=com</container>
  </rule>
  <rule>
    <expression>Default</expression>
    <description>Default</description>
    <container>l=roles,dc=oracle,dc=com</container>
  </rule>
</role>
</container-rules>

```

In the LDAPContainerRules.xml file, each rule contains the following sections:

- **Expression:** This specifies the actual rule that you use to find the namespace and the OU for LDAP.

The <expression> tag must be defined based on user/role attributes. Only the equal to (=) operator is supported in the <expression> tag. The expression can be based on multiple attributes, as shown in the example, and the LDAP container is determined based on an AND operation of all the defined attributes. If none of the rules satisfy, then the users or roles are put in the container for which expression is Default.
- **Description:** This is the namespace that is used for the Role Namespace attribute.

The description (namespace) associated to the default expression will always use Default. By default, roles do not have many attributes for creating meaningful expressions. Therefore, you need to add a new User Defined Field (UDF) attribute, for instance, the Role Location attribute. In this example, the Role Description attribute is used to define the rule.
- **Container:** This is the OU that is used to figure out where to create the user or role in LDAP.

Suppose a user is to be created with the attributes Country=US and Locality Name=AMER. This user would be created in the container l=amer,dc=oracle,dc=com. If a user is to be created in Country=France and Locality Name=FR, then it would be created in the container l=users,dc=oracle,dc=com because no expression matches these two attributes, and therefore, the default container is selected.

Understanding Context

A context is the environment in which an Oracle Identity Manager operation is performed. For example, a user creation operation performed on the Oracle Identity Manager Administrative and User Console is carried out in the Web context. The following information constitutes the context or environment in which this operation is performed:

- User performing the operation
- IP address of the computer from which the user creation request originated
- Date and time at which the request is submitted
- Proxy that is used to reach the application server

For example, if the user is created by running the bulk load utility, the context includes the user who started the bulk load utility, the computer from which the operation is being performed, and so on.

A context is maintained in main-memory. It consists of a set of context variables where each context variable has both a name and value. Each functional component involved in an operation, such as request management, reconciliation, or notification, can add values to the context. Context values can only be set, they cannot be modified. The context values act as a means of communication across components involved in an operation.

Context variable values are loaded into memory only when they are required. This enhances performance. A context also acts as a cache of the typical values required by event handlers. This helps reduce the need to fetch values from the repository each time the values are required.

- [Child Context](#)
- [Context Types](#)

10.1 Child Context

A child context is a subcontext that is initiated while an operation is in progress. For example, if user creation operation involves provisioning of resource through access policies, resource provisioning runs in the access policy context, which is the child context of the one in which user is being created. This means that contexts can be nested, and there can be a stack of contexts. New contexts can be created by functional components, and further processing starts using the newly created context.

10.2 Context Types

Context Manager supports the following context types:

- **SELF:** Operation is initiated through Oracle Identity Manager Self Service.
- **ADMIN:** Operation is initiated through Identity Administration or Advanced Administration. This is the default context.
- **RECON:** Operation is performed by reconciliation.
- **REQUEST:** Operation is performed by a request.
- **POLICY:** Operation is performed because of access policy.

Calling `ContextManager.getContextType()` should tell the type of context. Some of the information that you can retrieve under various contexts are:

- Reconciliation context: The profile from which the reconciliation event has been created can be retrieved by `ContextManager.getValue("profileName")` method call.
- Scheduled tasks run in ADMIN context: Some of the information that can be retrieved are:
 - Job name: `ContextManager.getValue("JOBNAME")`
 - Task name: `ContextManager.getValue("TASKNAME")`
- Request context: You can retrieve the request key by using the following code:

```
HashMap<String, ContextAware> requestContext = (HashMap<String, ContextAware>)
ContextManager.getValue("requestData", true);
requestContext.get("requestKey");
```
- Policy context: `ContextManager.getContextKey()` provides the policy that is evaluated. If multiple policies are applicable, then this returns the highest priority policy key.

Part II

Application-Specific Connectors

This part describes how to develop application-specific connectors.

It contains the following chapters:

- [Chapter 11, "Developing Resource Objects"](#)
- [Chapter 12, "Developing Provisioning Processes"](#)
- [Chapter 13, "Developing Process Forms"](#)
- [Chapter 15, "Developing Lookup Definitions, UDFs, and Remote Manager"](#)

Developing Resource Objects

The Resource Management features of the Administrative and User Console enable you to manage resource objects for an organization or individual user. Managing resources includes the following activities:

- Searching for and viewing the details of a resource
- Provisioning, disabling, enabling, and revoking a resource from users or organizations
- Managing resource administrator and authorizer roles
- Viewing, creating, and modifying workflows
- Creating and managing IT resources

This chapter includes the following topics related to managing resources:

- [Viewing Resource Details](#)
- [Working with Users and Organizations Associated with Resources](#)
- [Using the Resource Administrator Option](#)
- [Using the Resource Authorizers Option](#)
- [Using the Resource Workflows Option to View Workflows](#)
- [Using the Resource Workflows Option to Create and Modify Workflows](#)
- [Creating IT Resources](#)
- [Managing IT Resources](#)
- [Managing Resources By Using the Design Console](#)

11.1 Viewing Resource Details

To view the details of a resource:

1. Login to the Administrative and User Console, and then click **Advanced**.
2. In the Welcome page, under Configuration, click **Manage Resource**.

Alternatively, click the **Configuration** tab, click **Resource Management**, and then select **Manage Resource**.

The Resource Search page is displayed.

3. Use the fields at the top of the page to select the search criteria, and enter the corresponding search value in the adjoining field or use the asterisk (*) wildcard

character. To use the Resource Type and Target criteria, select a value from the corresponding box.

4. From the Resource Audit Objective list, select the required option.

The Resource Audit Objective list lets you group resources by any data type. You can select multiple values for the same resource. You can also add audit schedule values for quarterly, semiannual, and annual reviews in the list of values of the field, and select a combination, such as SOX and quarterly, as audit requirements.

The predefined values in the Resource Audit Objective list are as follows:

- SOX (Hosts Financially Significant Information)
- HIPAA (Hosts Private Healthcare Information)
- GLB (Hosts Non-Public Information)
- Requires Quarterly Review
- Requires Annual Review

5. Click **Search**.

The results table is displayed.

6. Click the name of a resource. For example, you can select a resource named Active Directory.

The Resource Detail page is displayed.

7. To view detailed information about the resource, use the menu.

The detailed information depends on the type of object, such as user or organization. For example, the detailed information that you can view for the organization object includes the following:

- Organization Associated With This Resource
- Resource Administrators
- Resource Authorizers

11.2 Working with Users and Organizations Associated with Resources

You can enable, delete, and revoke resources that are associated with an user or organization. You can also determine mapping categories for resources that are provisioned more than once to a user or organization.

To work with an organization that is associated with a resource:

Note: The procedure in this section is for working with an organization associated with a resource. You can also find the users associated with a given resource in the exact same way, as described for organizations in this section.

1. Perform Steps 1 through 3 of the procedure described in the ["Viewing Resource Details"](#) on page 11-1.
2. Select the **Organization Associated For the Resource** option.
The Organization Associated For the Resource page is displayed.
3. Use the options to filter the list of associated organizations.

Selecting the **All** option lists all the organizations. The By Status option filters the organizations on the basis of values in the Resource Status column. The organizations associated with the resource are listed under the Organization Name column. The resource status in this case, indicates that the resource is provisioned for each of the organizations listed. To modify the resource for the organization, select one of the following:

- Enable
- Disable
- Revoke

The value in the Identifier column corresponds with a field type that you can map from the Process Definition form in the Design Console by using the Map Descriptive Field. This value lets you distinguish which mapping category is defined, such as Process Type, Organization Name, or Request Key, when the same resource has been provisioned several times to the same organization.

11.3 Using the Resource Administrator Option

On the Resource Detail page, select **Resource Administrator**. The Resource Administrators page displays the names of roles that are assigned as administrators to this resource. This page also displays the Write Access and Delete Access permissions. These are permissions that the administrator roles have on the resource, but not with resource parameters. Write access allows the role to make changes to the resource. Delete access allows the role to delete the resource.

Note: Make sure that resource you want to delete is not used, for example, the resource is not provisioned or does not have a form associated to it.

You can perform the following operations:

- [Assigning Roles as Administrators for Resources](#)
- [Updating Permissions of an Administrative Role](#)

11.3.1 Assigning Roles as Administrators for Resources

To assign a role as administrator for resources:

1. Click **Assign.**

The Assign Administrators page is displayed.

This page displays all role names that can be assigned to this resource. Select the options to activate the write and delete access and assign the role to this resource.

2. Click **Assign.**

The Confirm Assign page is displayed. This page displays the new roles that are to be assigned as administrators for the resource.

3. Click **Confirm Assign or click **Cancel**.**

The Resource Administrators page is displayed with a list of all role names associated with this resource. You can modify this information.

11.3.2 Updating Permissions of an Administrative Role

You can update the permissions of an administrative role.

To update the permissions:

1. Click **Update Permissions**.

The Update Administrators page is displayed.

2. To change the permission setting for an administrative role, click the options for write and delete access.
3. Click **Update** to make the modifications, otherwise, click **Cancel**.

The Confirmation page is displayed. It displays the administrative role names that you updated.

4. If these are the correct names, click **Confirm Update**, otherwise, click **Cancel**.

11.4 Using the Resource Authorizers Option

You can determine which roles are authorized to provision the resource.

To determine the resource authorizer:

1. On the Resource Detail page, select **Resource Authorizer** from the menu.

The Resource Authorizers page is displayed.

2. To set the level of priority for authorizing this resource, select **Increase/Decrease Priority**.
3. To delete the authorizer of this resource, select the appropriate **Role Name** option, and then click **Delete**.
4. To add additional roles to authorize resources, click **Assign**.

The Assign Authorizers page is displayed.

5. Select the appropriate role name option and click **Assign**, otherwise, click **Cancel**.

The Confirmation page is displayed.

6. If the information is correct, click **Confirm Assign**, otherwise, click **Cancel**.

The Resource Authorizers page is displayed. Note that the role name that you assigned to this resource is added to the results table.

11.5 Using the Resource Workflows Option to View Workflows

The Resource Workflows option in the Administrative and User Console consists of the Workflow Visualizer and the Workflow Designer. Using the Workflow Visualizer, you can view workflows. Using the Workflow Designer, you can create and edit workflows. This section discusses the Workflow Visualizer.

See Also: ["Using the Resource Workflows Option to Create and Modify Workflows"](#) on page 11-16

The Workflow Visualizer tool provides a visual representation of task sequences, dependencies, and other components of a workflow definition. The visual representation provides an overview of the workflow, its relationships, and the task components that constitute the flow. You can also print the workflow view.

The Workflow Visualizer tool displays processes of type Provisioning. The Provisioning type process is used to provision Oracle Identity Manager resources to users or organizations.

Note: To access the Workflow Visualizer, the Nexaweb applet requires your Web browser configuration to use Java Virtual Machine 1.4.2.x.x.

This section includes the following topics:

- [Opening the Workflow Visualizer](#)
- [Elements of the Workflow Visualizer](#)
- [Operations on the Workflow Visualizer](#)

11.5.1 Opening the Workflow Visualizer

To open the Workflow Visualizer:

1. On the Resource Detail page, select **Resource Workflows** from the list.

The Resource Workflows page is displayed. This page displays the resource name and a table that lists the names of the workflow definitions for this resource.

2. To render the workflow definition into a graphic flowchart, select the required workflow.

A graphical representation of the workflow definition is displayed in a new window.

11.5.2 Elements of the Workflow Visualizer

For provisioning workflows, multiple tabs are displayed on the Workflow Designer page. Provisioning workflows can have forms associated with them, and the workflow details header shows the form name.

[Table 11–1](#) lists the information fields in the Workflow Visualizer.

Table 11–1 Information Fields in the Workflow Visualizer

Field	Description
Workflow Name	The name of the Process Definition.
For Resource	The name of the Object (resource object that is provisioned).
Workflow Type	The Process Definition type, which is Provisioning. The type also indicates whether or not the workflow is the default for the resource.
Form Name	The name of the form associated with a provisioning workflow.

[Table 11–2](#) describes the toolbar menu items in the Workflow Visualizer.

Table 11–2 *Toolbar Menu items in the Workflow Visualizer*

Field	Description
Display Option	<p>This option lets you view the elements on the page. You can show or hide the elements on the page, which helps in keeping the page uncluttered.</p> <p>Display Unknown Response Code: The Unknown Response Code is defined for every task in the workflow. It is not used in the logic of the workflow. However, you can use this option to display the Unknown Response Code.</p> <p>Display Adapter Name On-Screen: You can display the name of the automated adapter.</p> <p>Display Undo Tasks: You can display the undo tasks for the tasks.</p> <p>Display Recovery Tasks: You can display the recovery tasks for the tasks.</p>

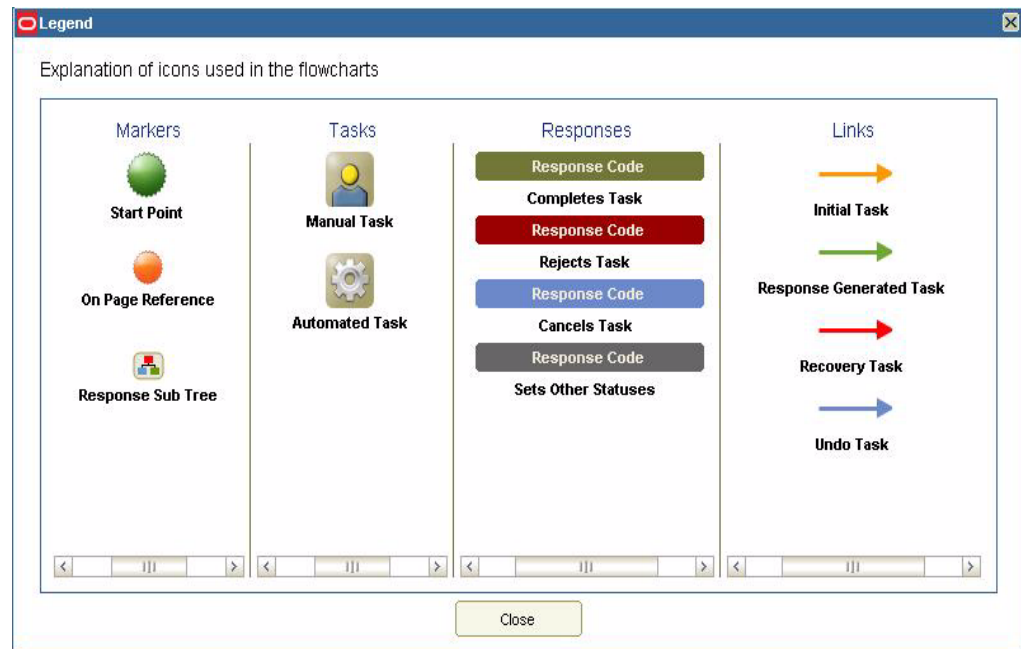
Table 11-2 (Cont.) Toolbar Menu items in the Workflow Visualizer

Field	Description
Generate Image	This option enables you to save the workflow view as an image that can be printed. When you click this menu item, a new browser window opens and it displays a JPEG formatted image. The entire workflow is displayed, even parts of the flowchart that are hidden due to scrolling limitations of the display area. You can then use the standard Web browser features to save the image on your computer.
Reload Workflow	This option refreshes the workflow view and rearranges the different items on the page based on a predefined graph algorithm.

Table 11–2 (Cont.) Toolbar Menu items in the Workflow Visualizer

Field	Description
Legend	<p>This option provides an explanation of all the visual components that are used to create the flowchart of the workflow definition. Figure 11–1 shows the Legend page.</p> <p>Markers</p> <p>The Markers nodes represent position markers for special conditions. These conditions are:</p> <p>Start Point: This marker represents the logical start point within the workflow. It is not an actual task within the workflow definition.</p> <p>On-Page Reference: This marker represents a task node that has already been drawn somewhere else in the workflow chart. It is used to show connectivity to other tasks without crowding the workflow view with crossing links.</p> <p>Response Sub-Tree: The Response Sub-Tree (Expansion Nodes) helps keep the workflow uncluttered by hiding significant subtrees of response nodes. You can double-click the Expansion Node marker to redraw the flowchart with the responses.</p> <p>Tasks</p> <p>The Tasks nodes represent the tasks in the workflow. They are:</p> <p>Manual Tasks: These tasks require user action in order to be completed.</p> <p>Automated Tasks: These tasks do not require user interaction in order to be completed. Automated tasks always require a process task adapter. Provisioning processes generally consist of automated tasks.</p> <p>Responses</p> <p>The Response nodes represent the response codes that are defined on the tasks. The Response node shows the actual response code within it. The response code is based on the status that the response has set on the task.</p> <p>Completes Task: The process task has been completed, and this is indicated in green color.</p> <p>Rejects Task: The process task has been rejected, and this is indicated in red color.</p> <p>Cancels Task: The process task has been canceled, and this is indicated in blue color.</p> <p>Links</p> <p>Direction arrow lines connect the task and response nodes and indicate the flow of the workflow. The color of the link indicates the type of relationship between two nodes that it connects. The types of links are:</p> <p>Initial Task: The Initial Task is the first process task in the workflow definition.</p> <p>Response Generated Task: The Response Generate Task is defined as a process task that is triggered when the current task has the Completed status. In general, a new process task can be triggered when the conditional task receives a particular response code in conjunction with the running of the process task.</p> <p>Recovery Task: The Recovery Task is defined as a process task that is triggered when the current process task has the Rejected status.</p> <p>Undo Task: The Undo Task is defined as a process task that is triggered when the current process task has the Canceled status.</p> <p>Dependent Task: The Dependent Task is defined as a process task that is dependent on another process. Oracle Identity Manager can start this type of task only when the process task on which it is dependent is completed.</p>

[Figure 11–1](#) shows the Legend page.

Figure 11–1 Legend Page

In addition to the Information Fields and Toolbar Menu Items of the Workflow Visualizer, the UI elements of the workflow are tasks and responses. For information about tasks and responses, see [Table 11–1](#) and the "Creating and Configuring Tasks and Responses" on page 11-31.

11.5.2.1 Using the Provisioning Workflow Definition Event Tabs

The Provisioning Workflow Definition is displayed with associated event tabs in the logical flow of the way tasks get executed based on their responses. The event tabs represent the various task sequences for a specific event in the workflow definition. When you click an event tab, it displays the appropriate tasks for the workflow event of the process. You can arrange the flowchart to meet your requirements. If there is no task defined for the workflow event, then the tab displays a blank view. If there is more than one task sequence for the workflow event type, then the tab displays a menu from which you can select the process flowchart that you want to view.

11.5.2.1.1 Provisioning Tab The Provisioning tab shows the tasks that will provision a resource. When the workflow type is Provisioning, the workflow shows all the tasks needed to provision a resource.

11.5.2.1.2 Reconciliation Tab The Reconciliation tab shows the reconciliation event for the provisioning process with marker tasks inserted into it: either Reconciliation Insert Received, Reconciliation Update Received, or Reconciliation Delete Received. These tasks can have adapters attached to them to start a provisioning action. If a task has no adapters attached to it, then a response code of Event Processed is assigned to the task. Additional provisioning process tasks can be generated based on this response code to start a provisioning flow due to the reconciliation event.

11.5.2.1.3 Service Account Tab The Service Account tab shows all the provisioning processes of service accounts for users (administrators). When a user is provisioned with a service account, Oracle Identity Manager manages a mapping from the user's identity to the service account. When the resource is revoked or the user is deleted, the

provisioning process for the service account is not canceled. Instead, a task is inserted into the provisioning process to remove the mapping from the user to the service account. The provisioning processes of the service account are: `Service Account Changed`, `Service Account Alert`, and `Service Account Moved`.

11.5.2.1.4 User Event Tab The User Event tab shows the workflows that respond to changes to a user record, for example, updating the password or user ID.

11.5.2.1.5 Org Event Tab The Org Event tab shows workflows that respond to changes to an organization record (for example, updating the name or parent name) that the resource is provisioned to or the organization of the user that the resource is provisioned to.

11.5.2.1.6 Resource Event Tab The Resource Event tab shows workflows that respond to state changes of the provisioned resource instance, for example, being enabled or disabled.

11.5.2.1.7 Form Event Tab The Form Event tab shows workflows that respond to data changes in the process form of the provisioned resource instance.

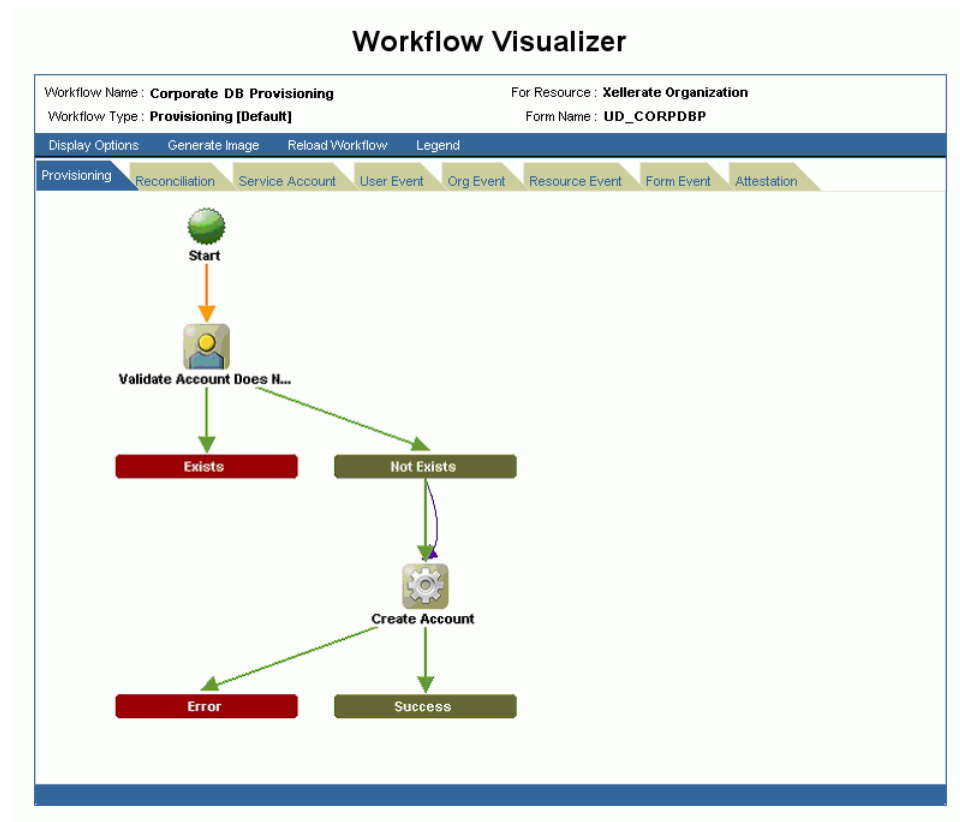
11.5.2.1.8 Attestation Tab The Attestation Event tab shows the workflows that respond to data changes in an attestation process.

11.5.3 Operations on the Workflow Visualizer

This section discusses the various operations that you can perform by using the Workflow Visualizer:

- [Rearranging Elements](#)
- [Using the Expansion Nodes](#)
- [Accessing the Task Details](#)

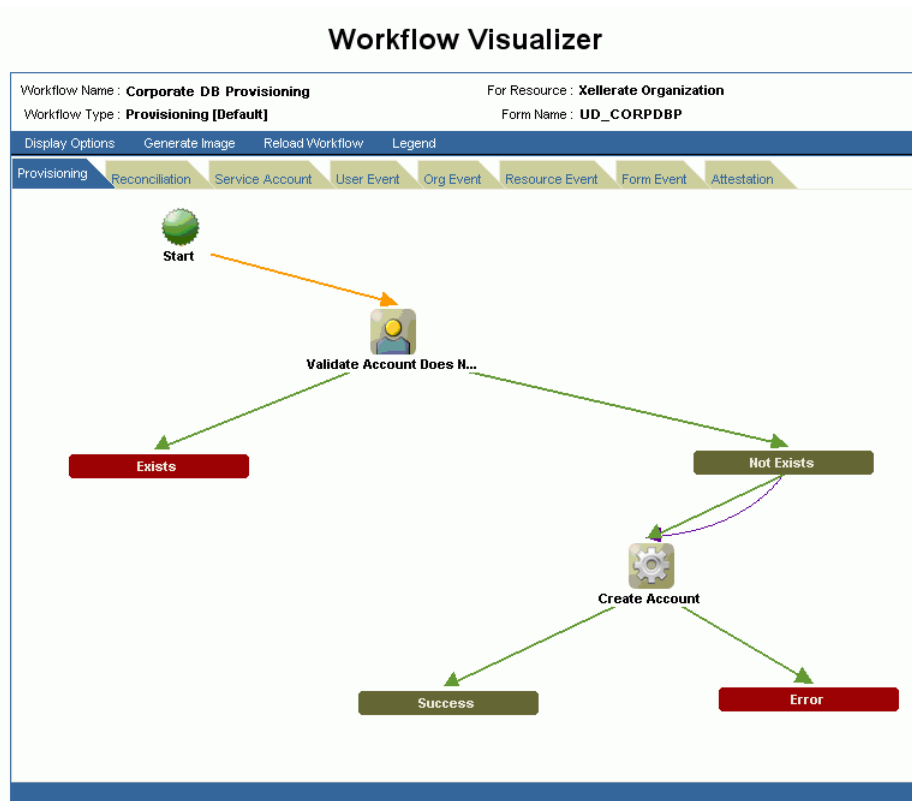
Suppose the Corporate DB Provisioning workflow definition is shown. Selecting an event tab displays the appropriate sequence of tasks for that event. These event tabs are discussed in the "[Using the Provisioning Workflow Definition Event Tabs](#)" on page 11-9. [Figure 11-2](#) shows a sample workflow in the Workflow Visualizer.

Figure 11–2 Sample Workflow Displayed in the Workflow Visualizer

11.5.3.1 Rearranging Elements

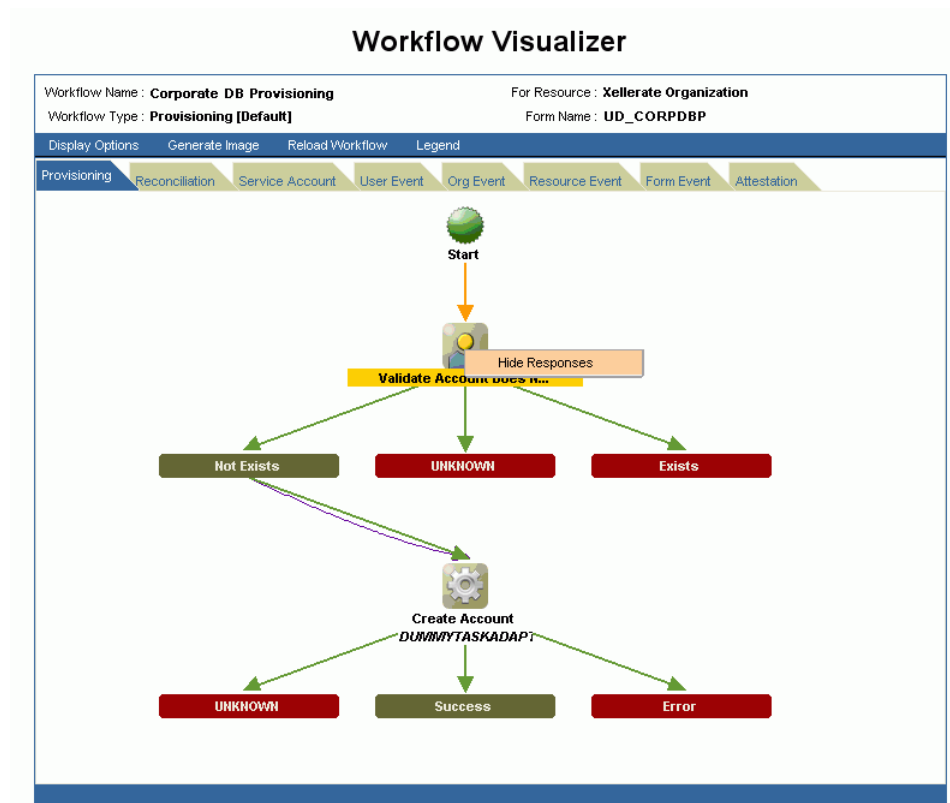
You can rearrange the graphical workflow by moving the icons that constitute the workflow definition to any location in the workflow view. As you move an icon component, the direction arrow continues to be associated with the link. The drag-and-drop functionality of the components in a workflow is illustrated in [Figure 11–3](#).

Figure 11–3 Using Drag-and-Drop in the Workflow Visualizer



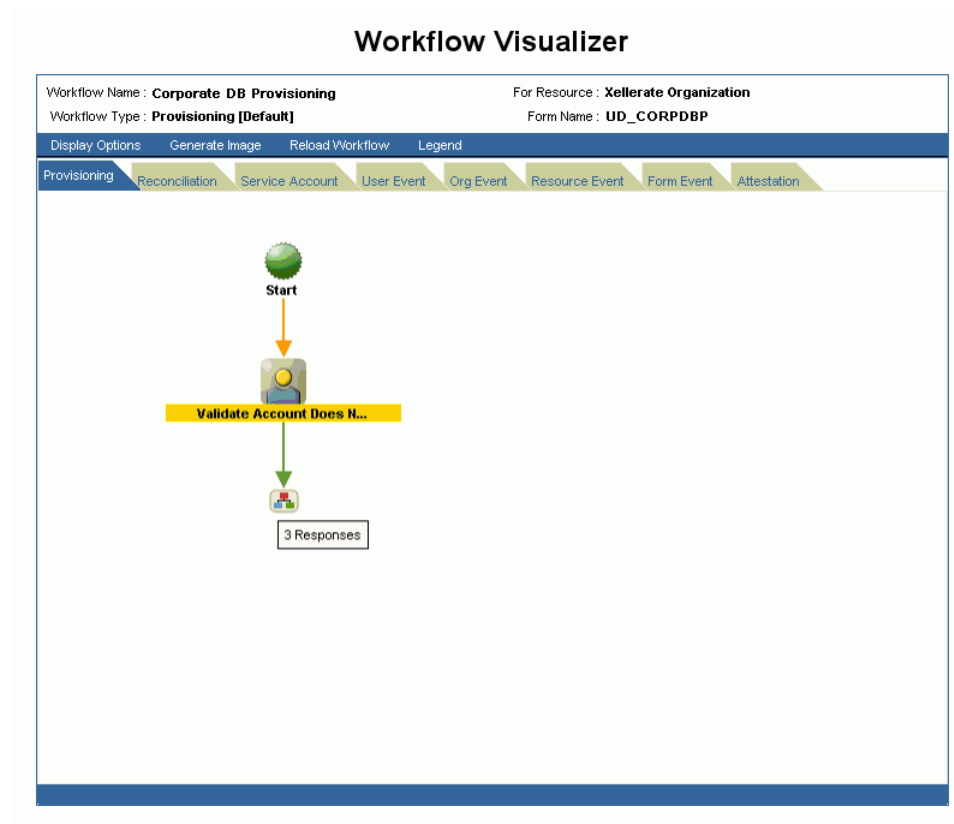
You can also use the **Display Options** toolbar menu item to display or hide Unknown Response Code, Adapter Name, Undo Tasks, and Recovery Tasks. The workflow automatically refreshes and redraws the workflow based on the changes that you made.

When you right-click a task node, the Hide Responses option is displayed. When you click this option, the response subtree collapses and is replaced with an expansion node. The task node label is highlighted in yellow to denote that it was collapsed. If the node is collapsed, then the Hide Responses option does not appear. [Figure 11–4](#) shows the task node.

Figure 11–4 Using the Task Node (Shortcut Menu)

11.5.3.2 Using the Expansion Nodes

Task Nodes with more than five response codes, not including the Unknown Response code, are not to be drawn with their responses in the flowchart. Instead, an expansion node replaces the entire response subtree. When you double-click the expansion node, the flowchart is redrawn to display the response subtree for the parent task (node). The label of the task node is highlighted in yellow. [Figure 11–5](#) shows a collapsed response subtree.

Figure 11–5 Collapsed Response Subtree in the Workflow Visualizer

Note: When you place the cursor over the expansion node, a tooltip indicates how many response codes are associated with it. Unknown Response Codes are hidden, by default.

11.5.3.3 Accessing the Task Details

To view detailed information about a particular task, double-click the task icon. The Task Detail page displays information about the task definition on the following tabs:

- **General:** This tab displays task information, for example, the name and description.
- **Automation:** This tab provides information about any adapter automating the task, its status, and variable mappings.
- **Task Assignment:** This tab displays information about how the task is assigned and all associated information.
- **Depends On:** This tab lists all tasks that the selected task depends on.
- **Resource Status Management:** This tab shows the mapping between the task status and the resource status.

11.5.3.3.1 **General Tab** [Table 11–3](#) describes the fields on the General tab:

Table 11–3 Fields on the General Tab

Field	Description
Task Name	This field displays the name of the process task.
Task Description	This field displays explanatory information about the process task.
Task Effect	This field indicates the process action for this task. It can be <code>ENABLED</code> , <code>DISABLED</code> , or <code>NONE</code> . A process is enabled or disabled for a user's access to a resource. A disabled action will also disable all associated tasks. The <code>NONE</code> action indicates that this task is not associated with a particular process action.
Retry Interval	This field indicates the time in minutes, for which you want to wait before adding this process task instance.
Retry Attempt Limit	This field indicates the number of times Oracle Identity Manager will retry a rejected task.
Conditional Task	This field specifies any condition that must be met for the process task.
Complete On Recovery	This field indicates that Oracle Identity Manager will change the status of the current process task from <code>Rejected</code> to <code>Unsuccessfully Completed</code> on completion of all recovery tasks that are generated. This flag triggers other dependent process tasks.
Allow Cancellation While Pending	This field indicates whether or not the process task can be canceled if its status is <code>Pending</code> .
Allow Multiple	This field indicates whether or not the task is allowed to be inserted multiple times within a single process instance.
Required For Workflow Completion	This field indicates that the process cannot be completed if the process task does not have a <code>Completed</code> status.
Manual Insert	This field indicates whether or not a user can manually add the current process task to the process.

11.5.3.3.2 Automation Tab Tasks belonging to provisioning processes are usually automated. [Table 11–4](#) describes the fields on the Automation tab.

Note: If the task is not automated, then this tab is not displayed.

Table 11–4 Fields on the Automation Tab

Field	Description
Adapter Name	This field shows the name of the adapter.
Adapter Status	This field indicates whether or not the adapter is completely mapped.
Adapter Variable	This field contains a user-defined placeholder within the adapter that contains run-time application data used by its adapter tasks.
Mapped?	This field indicates whether or not the adapter variable is mapped.

11.5.3.3.3 Task Assignment Tab This tab specifies the assignment rules for the process task. These rules determine how the process task is assigned.

Tasks belonging to provisioning processes are usually automated. As a result, they do not need task assignment rules.

11.5.3.3.4 Depends On Tab This tab displays the task name that the current task is dependent on.

11.5.3.3.5 Resource Status Management Tab A resource is provided with predefined provisioning statuses that represent the various statuses of the resource object throughout its lifecycle as it is provisioned to the target user or organization. This tab displays the link between the status of a process task (Task Status) and the provisioning status of the resource (Resource Status) to which it is assigned. [Table 11-5](#) describes the fields on the Resource Status Management tab.

Table 11-5 Fields on the Resource Status Management Tab

Field	Description
Task Status	The status can be one of the predefined provisioning status types.
Resource Status	The status can be one of the following: <code>Waiting</code> , <code>Provisioning</code> , <code>None</code> , <code>Ready</code> , <code>Enabled</code> , <code>Disabled</code> , <code>Revoked</code> , <code>Provisioned</code> , and <code>Provide Information</code> .

11.6 Using the Resource Workflows Option to Create and Modify Workflows

The Workflow Designer provides the ability to create and modify workflows. While the Workflow Visualizer provides a graphical view of the workflows, the Workflow Designer provides the ability to create workflows and to edit them.

See Also: ["Process Definition Form"](#) on page 12-5 for information about the Process Definition form

This section discusses the following topics:

- [Opening the Workflow Designer](#)
- [Creating a Workflow](#)
- [Workflow Designer Main Page](#)
- [Creating and Configuring Tasks and Responses](#)
- [Configuring Data Flows](#)

11.6.1 Opening the Workflow Designer

To open the Workflow Designer:

1. In the welcome page of Oracle Identity Manager Advanced Administration, click **Manage Resource**. Alternatively, you can click **Configuration**, and from the Resource Management list, select **Manage Resource**.

The Resource Search page is displayed.

Note: To open the Workflow Designer by using Mozilla Firefox Web browser, an additional authentication dialog box might be displayed. Providing authentication in this dialog box allows access to the Workflow Designer. To avoid this additional authentication:

1. In Mozilla Firefox, from the Tools menu, select **Options**. The Options dialog box is displayed.
2. Click **Privacy**.
3. Select the **Accept third-party cookies** option.
4. Click **OK**.

The additional authentication is not required when the Workflow Designer is opened by using Microsoft Internet Explorer Web browser.

2. Search for a resource.
3. Select a resource by clicking the resource name. The Resource Detail page is displayed.
4. Select **Resource Workflows** from the additional details list. The Resource Workflows page is displayed.
5. Click **Create New Workflow** to open the Workflow Designer and create a new workflow. Alternatively, click **Edit** in the Edit Workflow column of the results table to open the Workflow Designer and edit an existing workflow.

11.6.2 Creating a Workflow

On the Resource Workflows page, when you click **Create New Workflow**, the Workflow Designer opens with the Create Workflow dialog box, as shown in [Figure 11-6](#).

Figure 11-6 Create Workflow Dialog Box

The screenshot shows a dialog box titled "Create Workflow" with the subtitle "Create New Provisioning Workflow". Inside the dialog, there are three input fields: "Workflow Name" (with a red asterisk indicating it is required), "Workflow Form", and "Default Workflow" (with a checked checkbox). At the bottom left, there is a legend: "* Indicates Required Field". At the bottom right, there is a button labeled "Create Workflow".

In this dialog box, you must specify the values that are required to create a new workflow. [Table 11-6](#) describes the fields in the Create Workflow dialog box.

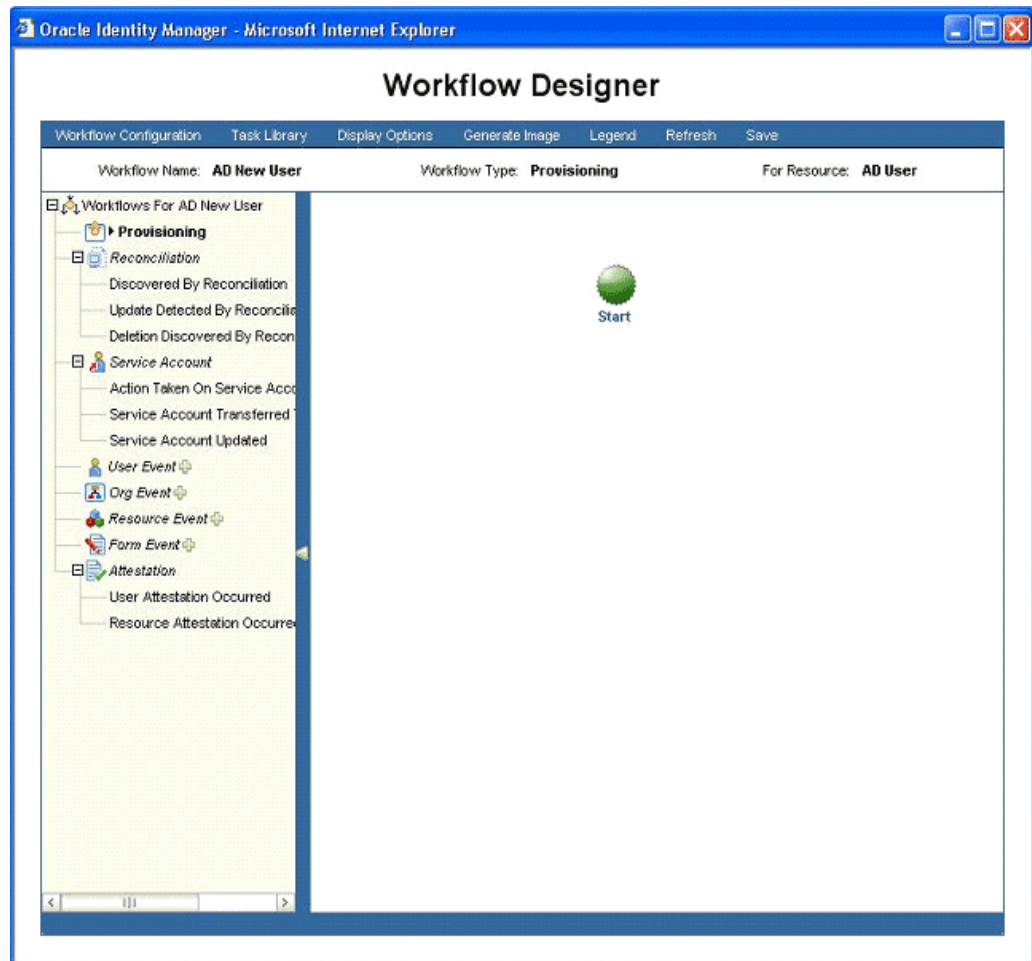
Table 11–6 Fields in the Create Workflow Dialog Box

Field	Description
Workflow Name	The name of the new workflow.
Workflow Form	The form associated with the resource for which the workflow is defined. The forms can be: <ul style="list-style-type: none"> ■ All the process forms that are not yet assigned to any processes ■ All the process forms assigned to the other processes defined for the current resource, for which this workflow is being defined
Default Workflow	This check box specifies whether or not the current Business Workflow is to be designated as the default provisioning Business Workflow for the resource object with which it is associated. If this check box is selected, then the Business Workflow will be set as the default provisioning Business Workflow for the resource object to which it is assigned. If this check box is not selected, then the process will start only if a process selection rule causes it to be selected.
Create Workflow	The button to create the workflow.

11.6.3 Workflow Designer Main Page

After you click **Create Workflow** in the Create Workflow dialog box by selecting the Provisioning option, the Workflow Designer main page is displayed as shown in [Figure 11–7](#).

Figure 11–7 Workflow Designer Main Page



This page has different sections, with each section giving more information or options to extend the new workflow.

The Workflow Designer main page consists of the following sections:

- [Information](#)
- [Toolbar](#)
- [Designer Page](#)
- [Menu Section](#)

11.6.3.1 Information

This section displays the following labels that provide global information about the current workflow:

- **Workflow Name:** The name of the current workflow
- **Workflow Type:** The type of the current workflow, Provisioning or Approval
- **For Resource:** The resource to which the current workflow is attached

11.6.3.2 Toolbar

The toolbar provides features to manage and view the workflow designer pages. This includes options to configure the global workflow information such as the name, form name, auto save, auto prepopulate, generating an image of the graphical workflow view, reloading the workflow, a popup legend, saving the workflow, and providing display options.

This section discusses the functions of the following toolbar buttons:

- [Workflow Configuration](#)
- [Task Library](#)
- [Display Options](#)
- [Generate Image](#)
- [Legend](#)
- [Refresh](#)
- [Save](#)

11.6.3.2.1 Workflow Configuration Clicking **Workflow Configuration** opens the Workflow Configuration dialog box, as shown in [Figure 11–8](#). This dialog box provides options for configuring the current workflow.

Figure 11–8 Workflow Configuration Dialog Box

The screenshot shows a dialog box titled "Workflow Configuration". Inside the dialog, there are several configuration options:

- Workflow Name:** A text input field.
- Default Workflow:** A checkbox that is currently checked.
- Descriptive Field:** A text input field with a "Clear" button next to it.
- Form Name:** A text input field with a "Clear" button next to it.
- Auto Save Form:** An unchecked checkbox.
- Auto Prepopulate Form:** An unchecked checkbox.

At the bottom of the dialog are two buttons: "OK" and "Cancel".

[Table 11–7](#) describes the fields in the Workflow Configuration dialog box.

Table 11–7 Fields in the Workflow Configuration Dialog Box

Field	Description
Workflow Name	The name of the current workflow.
Default Workflow	This check box specifies whether or not the current process is to be designated as the default provisioning process for the resource object with which it is associated. Note: For more information about this check box, see " Creating a Workflow " on page 11-17.
Descriptive Field	This is used to map any of the following to a particular instance of the provisioned resource: <ul style="list-style-type: none"> ■ Request Key ■ User Login ■ Organization Name ■ Process Type ■ Data From Workflow Form
Form Name	The form assigned to the current workflow.
Auto Save Form	This check box is used to set autosave for the form during provisioning without prompting the user for form data. This helps in setting default values for form fields either through predetermined set default values or through data flows.
Auto Prepopulate Form	This check box is used to prepopulate the fields during provisioning, with data either from default values or from data flows. Setting this option lets you see the forms while provisioning, along with the data on the fields that you can modify.

11.6.3.2.2 Task Library Clicking **Task Library** opens the Task Library page. The Task Library page displays a list of all the tasks in the workflow across all subworkflows. This page also shows a few parameters related to each task, such as in which subworkflows it is present (for provisioning workflows), whether or not multiple instances are allowed, whether or not cancellation while pending is allowed, retry period, and retry count. In addition, you can edit and delete tasks on this page. [Figure 11–9](#) shows the Task Library page.

Figure 11–9 Task Library Page

The screenshot shows the 'Task Library' window with a search section and a table of tasks. The search section includes fields for 'Task Name', 'Used in Workflows', 'Allow Multiple Instances', and 'Allow Cancellation While Pending', along with a 'Search' button. The table below lists various tasks with their 'Used in Workflows' status and 'Allow Multiple Instances' status.

Task Name	Used in Workflows	Allow Multiple Instances	All
Reconciliation Delete Received	Show List...	X	
System Validation	None	X	
Service Account Changed	Show List...	✓	
Reconciliation Update Received	Show List...	✓	
Service Account Alert	Show List...	✓	
Resource Attestation Event Occurred	Show List...	✓	
User Attestation Event Occurred	Show List...	✓	
Service Account Moved	Show List...	✓	
Reconciliation Insert Received	Show List...	X	

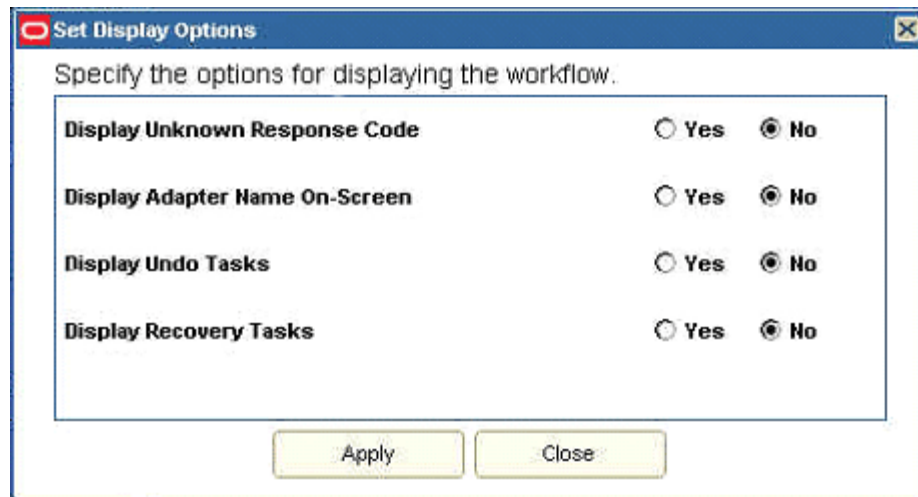
At the bottom of the window, there are three buttons: 'Remove Selected Task', 'Edit Selected Task', and 'Close'.

You can delete a task only after both the following conditions are met:

- The task is removed from all workflows. This implies that the task is deleted by right-clicking the task on any subworkflow and clicking **Remove Task and Subflow**.
- No instance of the task is present in the system. For instance, if a workflow is created with a task and if the resource for that workflow is provisioned to a user and the workflow is started resulting in the task being run, then an instance of that task is created in the system. In that case, the task cannot be deleted.

The Task Library page has search criteria on the top that you can use to search for tasks. The main section lists the tasks with various parameters. You can click a row to highlight it. If a task can be deleted, then the Remove Selected Task button is enabled along with the Edit Selected Task button.

11.6.3.2.3 Display Options Clicking **Display Options** opens the Set Display Options dialog box that provides options to specify how the workflow is displayed when you are designing the workflow. Figure 11–10 shows the Set Display Options dialog box.

Figure 11–10 Set Display Options Dialog Box

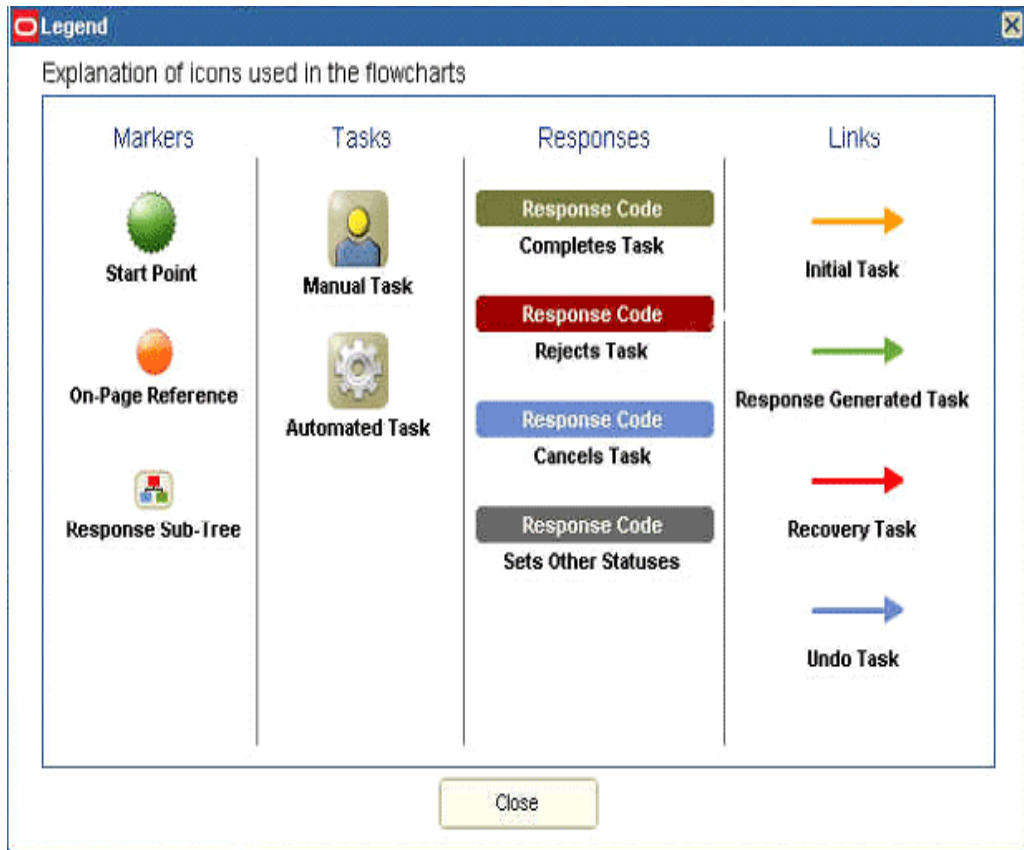
You can use this dialog box to enable or disable the following options:

- **Display Unknown Response Code:** Display or hide unknown response codes.
- **Display Adapter Name On-Screen:** Display or hide adapter names attached to the tasks.
- **Display Undo Tasks:** Display or hide undo tasks.
- **Display Recovery Tasks:** Display or hide recovery tasks.

11.6.3.2.4 Generate Image Clicking **Generate Image** saves the current view of the workflow as a JPEG image. The image opens in a new browser window.

11.6.3.2.5 Legend Clicking **Legend** opens the Legend dialog box, which is shown in [Figure 11–11](#). This dialog box shows the following types of elements:

Figure 11–11 Legend Dialog Box



- Markers:** These elements represent a particular marking or place in the workflow. For example, the starting point, an on-page reference, or a place representing an extended workflow with more elements underneath can be represented with a marker.

You can right-click a Task element and select the option to hide the responses. When you hide a response, the icon for the Response subtree is displayed to indicate that there are hidden responses. The on-page reference marker refers to other elements on the page whose relationship is not shown with links. An example of this is a response code defined for a task and for that response a response-generated task is defined. If this response-generated task has its response referring to the original task in a circular manner, then an on-page reference marker makes it easier to show the relationship.

- Tasks:** These icons are used to indicate manual and automated tasks. If a task has an event handler or an adapter attached to it for autocompletion, then it is an automated task. Otherwise, it remains a manual task.
- Responses:** These are the different color codes used for different types of response codes, such as Completes, Rejects, and Cancels. Any user-defined response code is shown with a different color code.
- Links:** These are the different color codes used for links that display the relationship or linkage between elements. Depending on the type of task the link refers to, the color code for the link is different. For example, the color code indicates whether or not the task is undo or recovery. The different types of links are: Initial Task, Response Generated Task, Recovery Task, and Undo Task.

11.6.3.2.6 Refresh Clicking **Refresh** reloads the workflow to display it with default indentations and locations for the labels and icons. It regenerates the topology to arrange the elements on the workflow by using the JGraph algorithm.

11.6.3.2.7 Save Clicking **Save** saves all changes made to the workflow, including all the additions and modifications to the Oracle Identity Manager database.

Caution: You must click **Save** to commit the changes. If you close the Workflow Designer main page without saving the workflow, then all the changes will be lost.

11.6.3.3 Designer Page

The designer page displays the workflow with all the elements and their positions in the process flow with the help of links. This is similar to a drawing board in which the components, such as tasks and responses, can be created by using appropriate options. These components on the designer page can be further configured. On this page, the different entities of the workflow can be graphically shown along with their relationship with each other. For a newly created workflow, this page displays a start marker that indicates the starting point for the workflow process. All the objects that are added to this page are relative to this marker, which acts as a reference point.

11.6.3.4 Menu Section

The menu section consists of the menu items that represent a particular subsection of the workflow. This section is available only for Provisioning workflows. The menu items available are the following:

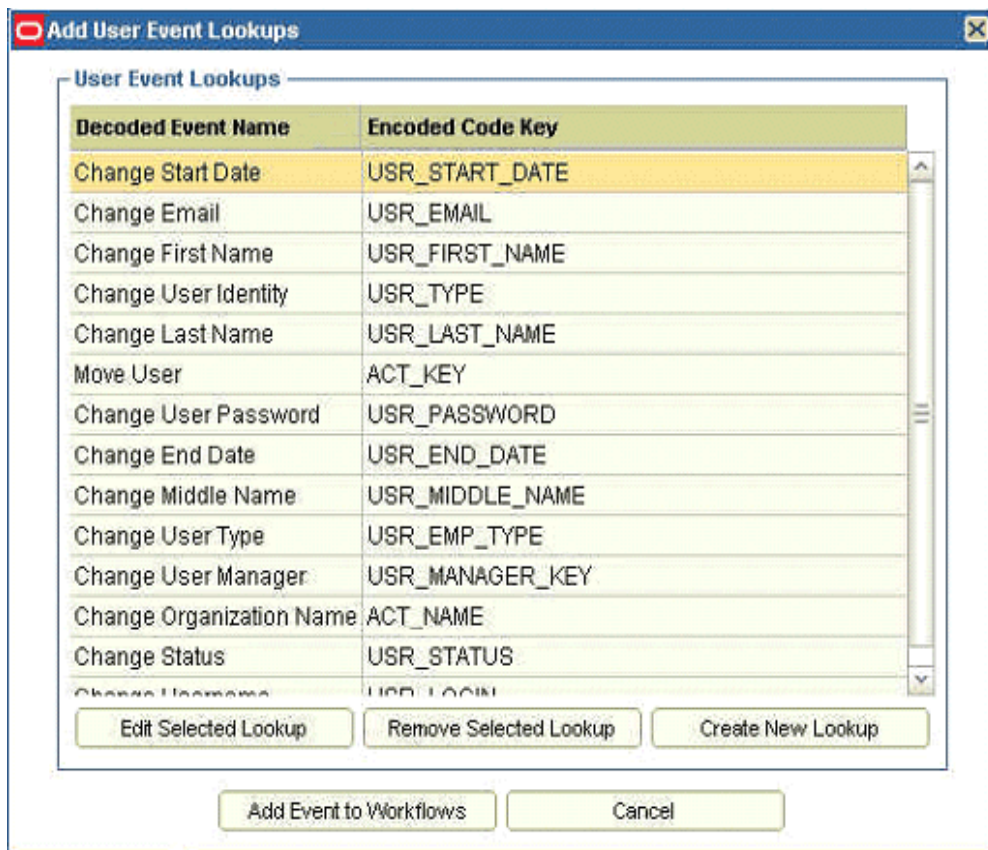
- **Provisioning:** This is the default page displayed when the Workflow Designer application is started.
- **Reconciliation:** This provides a list of tasks that are run on reconciliation events, such as Reconciliation Insert Received, Reconciliation Update Received, and Reconciliation Delete Received. These tasks are submenu items under the Reconciliation menu item.
- **Service Account:** Service accounts are general administrator accounts, such as admin1, admin2, and admin3, that are used for maintenance purposes. Usually, these accounts are used to allow one system, rather than a user, to interact with another system. The model for managing and provisioning service accounts is different from standard provisioning. Service accounts are requested, provisioned, and managed in the same manner as regular accounts. Service accounts use the same resource objects, provisioning processes, and process forms as regular accounts. A service account is distinguished from a regular account by an internal flag. When a user is provisioned with a service account, Oracle Identity Manager manages a mapping from the user's identity to the service account. This user is considered the owner of the service account. The tasks that are available under the Service Account menu item are Service Account Change, Service Account Alert, and Service Account Moved.
- **User Event:** This provides a list of tasks that are run based on the events on users. They have the following default names:
 - Change User Location
 - Move User
 - Change User Type

- Change User Password
- Change User Manager
- Change Username
- Change First Name
- Change Last Name
- Change User Identity

Note: These names are derived from the decoded values of `Lookup.USR_PROCESS_TRIGGERS` in the design console Lookup Definition form. If the values are modified, then these names will be different accordingly.

A user event can be inserted into the workflow by clicking the plus sign (+) icon next to the User Event menu item. Clicking the + icon opens the Add User Event Lookups dialog box with a list of currently available event tasks, as shown in [Figure 11-12](#). Selecting a task and clicking **Add Event to Workflows** will create a new menu item under the User Event menu and open the page for that workflow.

Figure 11-12 Add User Event Lookups Dialog Box



The Add User Event Lookups dialog box also provides the following options to create new lookup events and edit or remove existing lookup events:

- Create New Lookup: When you click this button, the Create Lookup Event dialog box is displayed, as shown in [Figure 11-13](#).

Figure 11-13 Create Lookup Event Dialog Box

- Edit Selected Lookup: When you click this button, the Edit Lookup Event dialog box is displayed, as shown in [Figure 11-14](#).

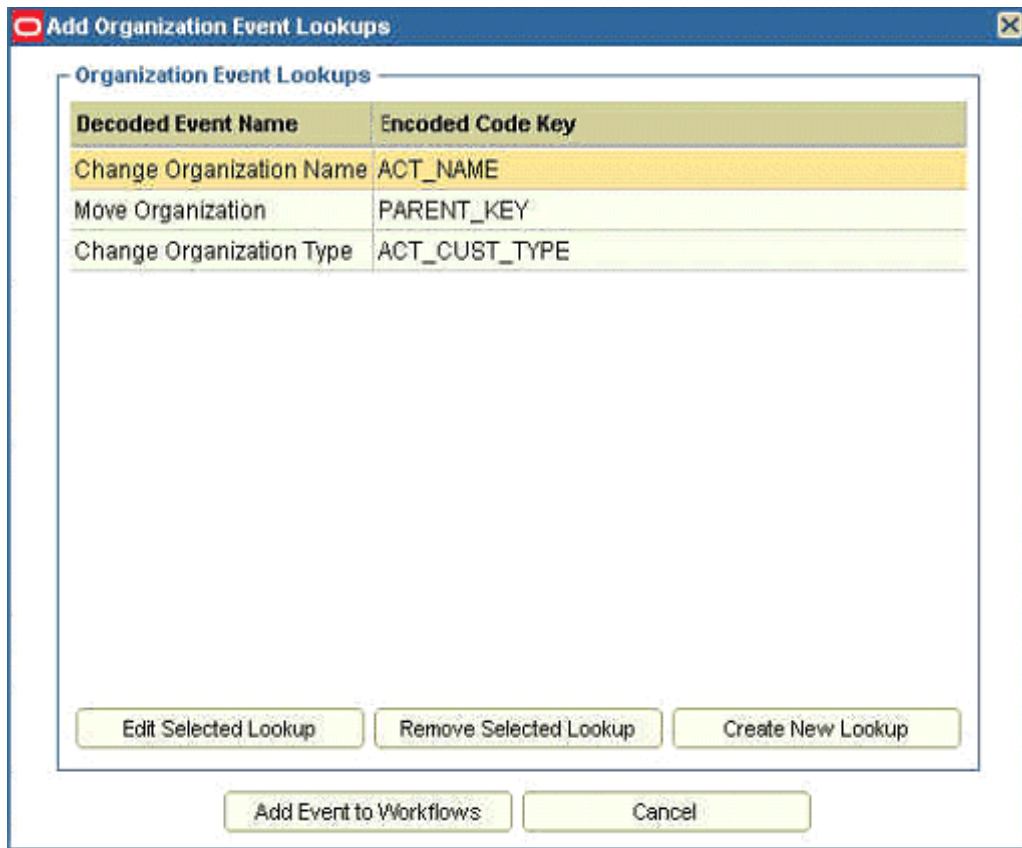
Figure 11-14 Edit Lookup Event Dialog Box

- Remove Selected Lookup: When you click this button, the Remove Lookup Event dialog box is displayed, as shown in [Figure 11-15](#).

Figure 11-15 Remove Lookup Event Dialog Box

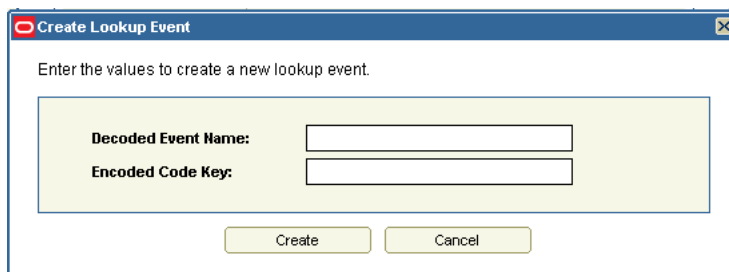
- **Org Event:** This provides a list of tasks that are run based on the events on organizations. They have the following default names:
 - Change Organization Type
 - Change Organization Name
 - Move Organization

An organization event can be inserted into the workflow by clicking the + icon next to the Org Event menu item. Clicking the + icon opens the Add Organization Event Lookups dialog box with a list of currently available event tasks, as shown in [Figure 11-16](#). You can select a task and click **Add Event to Workflows** to create a new menu item under the Org Event menu and open the page for that workflow.

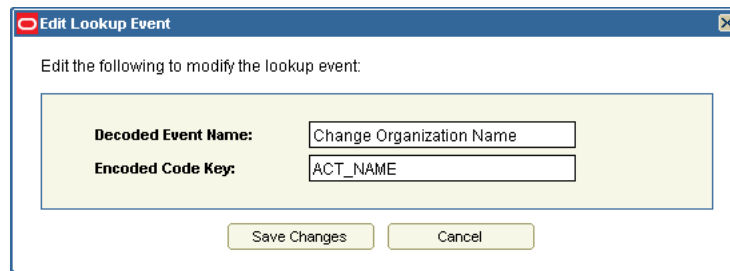
Figure 11–16 Add Organization Event Lookups Dialog Box

The Add Organization Event Lookups dialog box also provides the following options to create new lookup events and edit or remove existing lookup events:

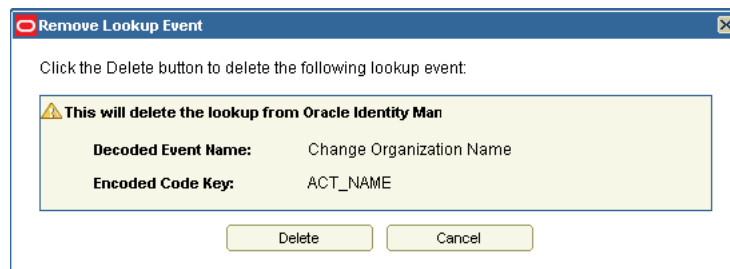
- Create New Lookup: When you click this button, the Create Lookup Event dialog box is displayed, as shown in [Figure 11–17](#).

Figure 11–17 Create Lookup Event Dialog Box

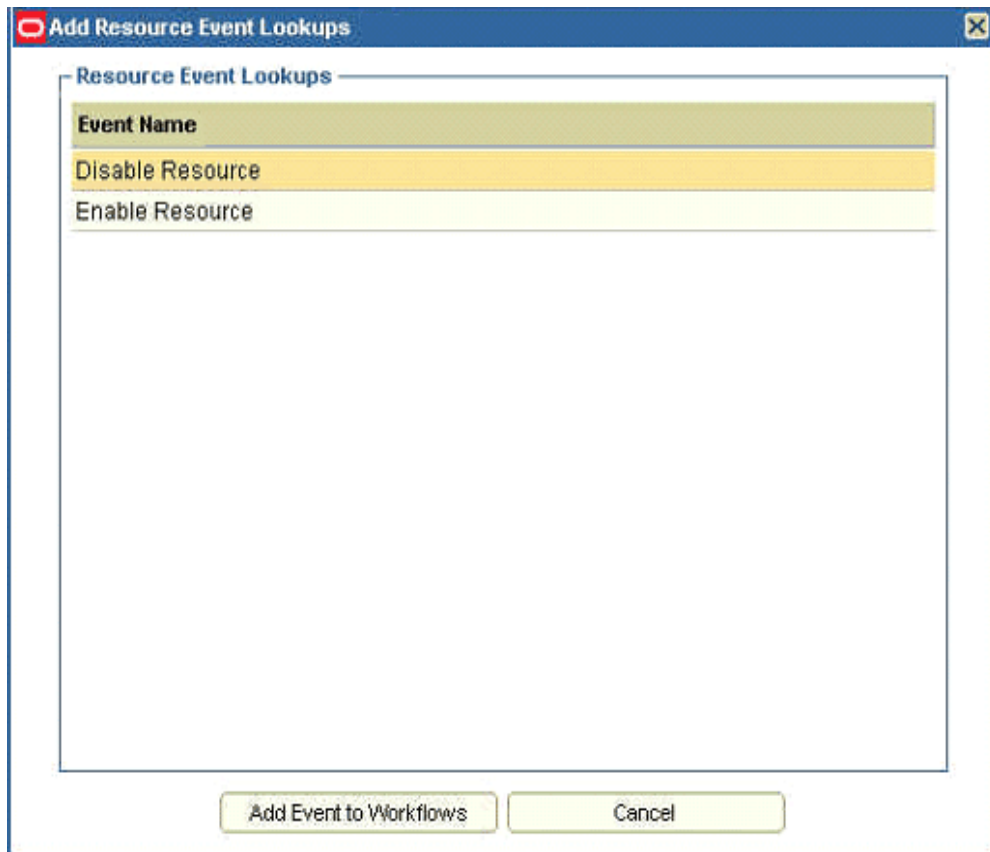
- Edit Selected Lookup: When you click this button, the Edit Lookup Events dialog box is displayed, as shown in [Figure 11–18](#).

Figure 11–18 Edit Lookup Event Dialog Box

- Remove Selected Lookup: When you click this button, the Remove Lookup Events dialog box is displayed, as shown in [Figure 11–19](#).

Figure 11–19 Remove Lookup Event Dialog Box

- **Resource Event:** This provides a list of tasks that are inserted into the workflow and run when an event occurs on the resource. These events are defined as disabled or enabled events on the resource. There are submenu items for Enable Resource and Disable Resource under the Resource Event menu item. A resource event can be inserted into the workflow by clicking the + icon next to the Resource Event menu item. Clicking the + icon opens the Add Resource Event Lookups dialog box with two options, Enable Resource and Disable Resource, as shown in [Figure 11–20](#). You can select an option and click **Add Event to Workflows** to create a new menu item under the Resource Event menu and open the page for that workflow.

Figure 11–20 Add Resource Event Lookups Dialog Box

- Form Event:** This provides a list of tasks that get inserted and run based on an event on a form field or child table. For events on parent process form fields, the name of the tasks have the following convention:

Field *field_name* Updated

The events on child tables are named based on the child table name and the type of event such as insert, update, and delete. A form event can be inserted into the workflow by clicking the + icon next to the menu item. Clicking the + icon opens the Add Form Event Lookups dialog box with the fields shown in [Figure 11–21](#).

Figure 11–21 Add Form Event Lookups Dialog Box

The screenshot shows a dialog box titled "Add Form Event Lookups". At the top, there is a "Form Type" dropdown menu currently set to "Parent Form". Below this, there are two sections: "Parent Form" and "Child Form". The "Parent Form" section is active and contains an "Operation" dropdown set to "Update" and a "Field" dropdown menu. The "Child Form" section is disabled and contains an "Operation" dropdown set to "Insert" and a "Form" dropdown menu. At the bottom of the dialog are two buttons: "Add Event to Workflows" and "Cancel".

In the Add Form Event Lookups dialog box, you can select either parent form or child form in the Form Type field. When you select **Parent Form**, the fields in the Child Form section are disabled. Similarly, when you select **Child Form**, the fields in the Parent Form section are disabled. In the Parent Form section, only the Update operation is available. In the Child Form section, the available operations are Insert, Update, and Delete. These operations trigger the event. Each section has fields for the form fields of the parent form, or the form names in case of child forms. The Task names for only the child table event tasks can be modified after creation.

Note: The parent form field event names are fixed, and the task name fields cannot be edited. Although the name is inherently in a fixed format, it can be customized and localized by updating the `global.workflow.startMarker.UpdatedField` property in the `xlRichClient.properties` file. See [Chapter 28, "Customizing Oracle Identity Manager Interfaces"](#) for information about customizing the UI.

- **Attestation:** This menu item is for the attestation events. There are two types of attestation events, User Attestation and Resource Attestation. No new events can be added to attestation although the existing workflows can be modified similar to other subworkflows.

11.6.4 Creating and Configuring Tasks and Responses

A workflow can consist of more than one task. This section discusses the following topics related to tasks:

- [General Menu Options](#)
- [Task Options](#)
- [Response Options](#)

- [Link Options](#)
- [Configuring Tasks](#)
- [Configuring Responses](#)

11.6.4.1 General Menu Options

You can right-click the designer page to display a menu with general options to create tasks and responses. The general menu options are:

- **Create New Task:** Creates a new task with a default name, which can be further modified and configured. The task is represented as an icon.
- **Insert Existing Task:** Displays the Existing Tasks dialog box with the list of all existing tasks across the subworkflows except the tasks present in the current subworkflow and the main user, organization, resource, and form event tasks for provisioning workflows. You can select a task and insert it in the current workflow.
- **Create Response:** Creates a new response with a default response code, which can be further modified and configured. The response is represented as an icon.

Various options are available when you right-click the task icons, response icons, and the links between the tasks and responses.

11.6.4.2 Task Options

You can right-click a task icon to display a menu that provides the following options related to tasks:

- **Link To Response:** This option is used to link a task to a response. To use this option, first create a response. When you select this menu item, a link is displayed starting from the task icon. This link extends with the mouse pointer. When you click the response, the arrowhead of the link positions itself on the response, and the response is created for the task.
- **Link To Undo Task:** This option is used to link two tasks with the undo relationship. It is used when you want to add a task as the undo task of the current selected task. To do this:
 1. Select the task to which the undo task is to be added.
 2. Right-click the task icon, and select the **Link To Undo Task** menu item.
 3. Select the target tasks icon to add it as the undo task.

Note: If the **Display Undo Tasks** option in the Display Options toolbar is selected with the value **No**, then the Undo task will not be visible after creating the undo relationship. To see the undo task, select **Yes** for the **Display Undo Tasks** option.

- **Link To Recovery Task:** This is used to link two tasks with the recovery relationship. It is used when you want to add a task as the recovery task of the currently selected task. To do this:
 1. Select the task to which the recovery task is to be added.
 2. Right-click the task icon, and select the **Link To Recovery Task** menu item.
 3. Select the target task to add it as the recovery task.

Note: If the **Display Recovery Tasks** option in the **Display Options** toolbar button is selected with the value **No**, then the recovery task will not be displayed after creating the recovery relationship. To display the recovery task, select **Yes** for the **Display Recovery Tasks** option.

- **Remove Task and Subflow:** This is used to remove a task and all the elements under the task. This includes all the links originating from the task and all their child elements and their child elements and so on. When the same task is present in multiple subworkflows and it is removed from one subworkflow, it gets removed from all the subworkflows where this task has the same parent task, which is the task whose response-generated tasks contain the current removed task.

Removing a task or the children will not delete the tasks from the system but only from the workflows. Deleting a task from the system permanently can be done from the Task Library. Removing tasks from the designer page still keeps the task definitions and removes them only from the workflows.

11.6.4.3 Response Options

You can right-click a response icon to display a menu that provides the following options related to responses:

- **Add Response Generated Task:** This is used to add a task as a response-generated task for the selected response. To do this:
 1. Create the response-generated task.
 2. Right-click the response, and select **Add Response Generated Task**. A link is created.
 3. Select the task. The link positions on the task and the relationship are created.
- **Remove:** This is used to remove a response. When you select this option, a confirmation page is displayed. Confirming the deletion removes the response and all its children. When a response is removed that contains generated tasks, then those tasks will be removed but not deleted. When a task is removed, it is removed only from the workflow and is not deleted permanently. You can permanently delete a task from the Task Library.

11.6.4.4 Link Options

You can remove the relationships between some elements by right-clicking the link and clicking the **Remove** option. This option is not available for all links. For example, for reconciliation workflows, you cannot delete the default tasks connected to the start marker. Therefore, you cannot remove the relationship between the start markers and the default tasks. The link for which you can remove the relationship is highlighted with a broken arrow when you roll your mouse on the relationship. When the arrow is highlighted, right-click the arrow and the **Remove** option is displayed. This helps in removing the link between a response and a task and to assign another response to the task, or to assign another task to the response, without the need to delete the link and create new ones.

11.6.4.5 Configuring Tasks

You can configure tasks in the Workflow Designer by using the Task Details dialog box. This dialog box is shown in [Figure 11–22](#). To open the Task Details dialog box, double-click the task icon on the designer page.

Figure 11–22 Task Details Dialog Box

The screenshot shows the 'Task Details' dialog box for a task named 'Change Username'. The dialog is divided into several sections:

- General:** Task Name is 'Change Username' and Task Description is 'Change Username'.
- Retry Configuration:** Includes fields for 'Retry Interval' and 'Retry Attempt Limit'.
- Properties:** Contains several checkboxes:
 - Allow Multiple Instances
 - Required For Workflow Completion
 - Disable Manual Insert
 - Complete On Recovery
 - Allow Cancellation While Pending

At the bottom of the dialog are 'Cancel' and 'Apply' buttons.

This section discusses the following tabs in the Task Details dialog box:

- [General Tab](#)
- [Automation Tab](#)
- [Notification Tab](#)
- [Task Assignment Tab](#)
- [Depends On Tab](#)
- [Resource Status Management Tab](#)

General Tab

[Figure 11–23](#) shows the General tab of the Task Details dialog box.

Figure 11–23 General Tab

The screenshot shows a window titled "Task Details" with a close button (X) in the top right corner. The main title is "Definition Of Task: Change Username". Below this is a tabbed interface with five tabs: "General" (selected), "Automation", "Notification", "Task Assignment", "Depends On", and "Resource Status Management".

Under the "General" tab, there are two input fields: "Task Name" with the value "Change Username" and "Task Description" with the value "Change Username".

Below these is a section titled "Retry Configuration" with two input fields: "Retry Interval" and "Retry Attempt Limit".

At the bottom is a section titled "Properties" with four checkboxes:

- Allow Multiple Instances
- Required For Workflow Completion
- Disable Manual Insert
- Complete On Recovery
- Allow Cancellation While Pending

At the very bottom of the dialog are two buttons: "Cancel" on the left and "Apply" on the right.

This tab lets you specify the general information about the task:

- **Task Name:** This is the name of the process task. This field can be edited, except when the task name cannot be changed. For example, on the Form Events page, the event task for parent field update.
- **Task Description:** This is descriptive information about the process task.
- **Retry Configuration:** This section is present only for provisioning workflows and consists of the following options:
 - **Retry Interval:** If a process task has the `Rejected` status, then this is the time interval in minutes before Oracle Identity Manager inserts a new instance of that task with a `Pending` status.
 - **Retry Attempt Limit:** This is the number of times Oracle Identity Manager retries a rejected task.
- **Properties:** This section has the following options:
 - **Allow Multiple Instances:** This check box determines whether or not the process task can be inserted into the current process more than once. If you select this check box, then multiple instances of the process task can be added to the process. If you deselect this check box, then the process task can be added to the current process only once.
 - **Required for Workflow Completion:** This check box determines whether or not the current process task must be completed for the process to be completed. If you select this check box, then the process cannot be completed if the process task does not have a `Completed` status. If you deselect this check box, then the status of the process task does not affect the completion status of the process.
 - **Complete On Recovery:** This check box determines whether or not the status of the task must be set to `Completed` on completion of the recovery tasks.
 - **Allow Cancellation While Pending:** This check box determines whether or not the process task can be canceled if its status is `Pending`. If you select this check box, then the process task can be canceled if it has a `Pending` status. If

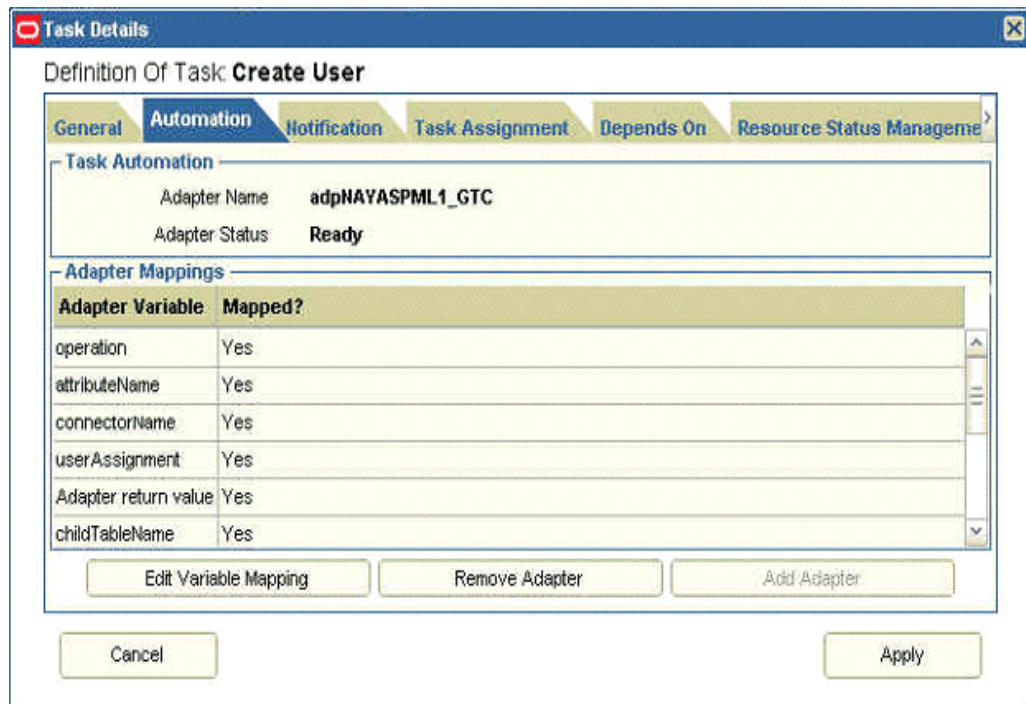
you deselect this check box, then the process task cannot be canceled if its status is Pending.

- **Disable Manual Insert:** This check box determines whether or not a user can manually add the current task to the workflow. If this check box is selected, then the task cannot be added to the workflow manually. If you deselect this check box, then a user can add the task to the process.

Automation Tab

Figure 11–24 shows the Automation tab of the Task Details dialog box.

Figure 11–24 Automation Tab



The Automation tab lets you attach an event handler or an adapter with the task that helps in the automation of the process task.

The options on this tab are divided into two parts. The task automation section shows the currently attached adapter with the status of the adapter. The Adapter Mappings section shows the adapter variable mappings. There are buttons on the tab that enable you to add an adapter or event handler, remove the adapter, and edit the variable mappings when an adapter is attached.

When you click **Add Adapter**, a dialog box is displayed. This dialog box consists of a section for the handler type with an option each for system event handlers and adapters. Selecting each option displays the corresponding descriptive text below the handler type section. You can select an item in the list and click **Add**.

The Adapter Mappings section shows the variables associated with the adapters along with the mappings. It displays the variable name and whether or not it has been mapped. When you select a variable, the Edit Variable Mapping button is enabled. You can click this button to open the Adapter Mappings dialog box with all the various options available to map this variable. This dialog box provides the following options:

- **Variable Name:** This text label displays the name of the adapter variable for which you are setting a mapping, such as UUID.
- **Data Type:** This text label displays the data type of the adapter variable. For example, String is the data type for the UUID variable.
- **Map To:** This list displays the types of mappings that you can set for the adapter variable, such as IT Resources.

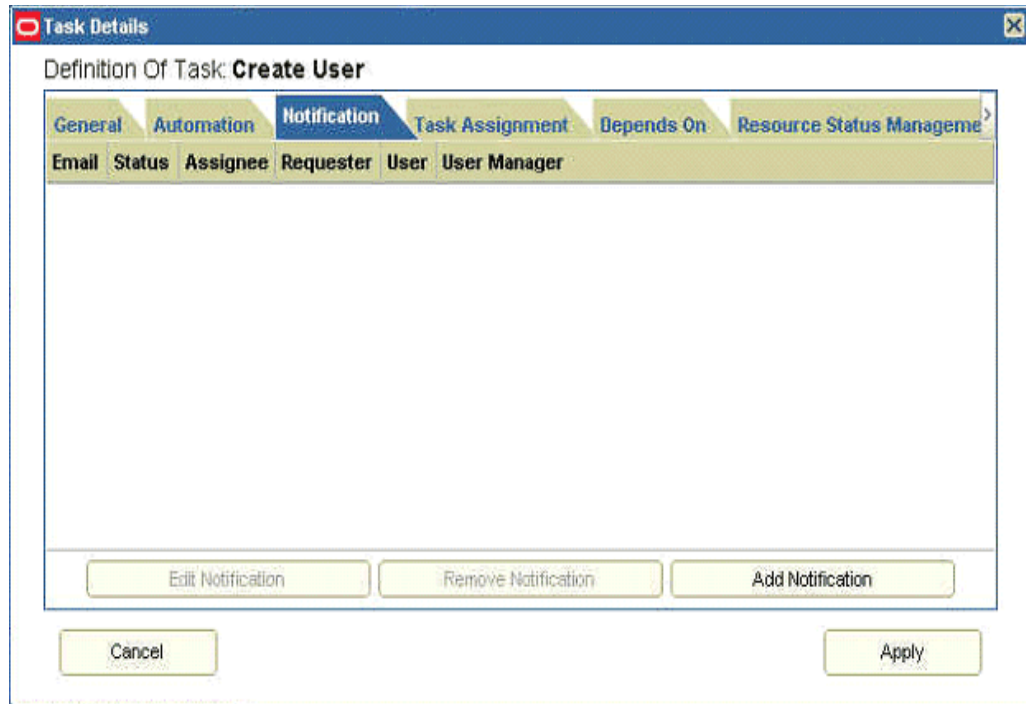
When you map the adapter variable to a location or contact, Oracle Identity Manager enables a list with values for a specific type of location or contact to which you are mapping the adapter variable. In addition, if you map the adapter variable to a custom process form and this form contains child tables, then Oracle Identity Manager enables the adjacent list. From this list, select the child table to which you are mapping the adapter variable. If you are not mapping the adapter variable to a location, contact, or child table of a custom process form, then this list is disabled.

- **Qualifier:** This list contains the qualifiers for the mapping that is selected in the Map To list, such as IT Asset.
- **Old Value:** This check box specifies whether or not you map the adapter variable to the value that was originally selected in the **Qualifier** check box before modification. Process task adapters associated with process tasks are conditionally triggered when some field on the process form is changed. If you select the **Old Value** option and the process task is marked `Conditional`, then the value that is passed to the adapter is the previous value of the field or variable for which the mapping is being selected. This is useful for fields that accept passwords. For example, if you want to disallow setting the password to the same value, then you can use the old value for comparison. If you are not mapping the adapter variable to a field that belongs to a child table of a custom process form, then this check box is disabled.

Note: Different fields may be displayed in the Adapter Mappings dialog box, based on what you select from the Qualifier and Map To lists.

Notification Tab

Figure 11-25 shows the Notification tab of the Task Details dialog box.

Figure 11–25 Notification Tab

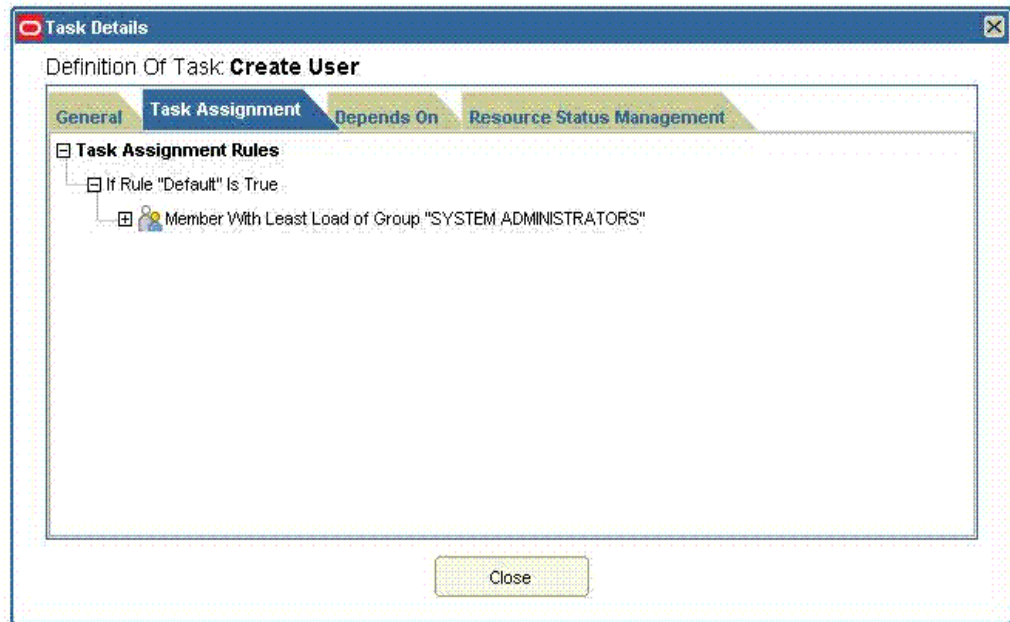
This tab lets you designate the e-mail notification to be generated when the current process task achieves a particular status. For each status that a task can achieve, a separate e-mail notification can be generated. If an e-mail notification is no longer valid, then you can remove it from the Notification tab.

Note: For Oracle Identity Manager to send an e-mail notification to a user, a template for the e-mail message must first be created by using the E-mail Definition form.

There are three buttons in the dialog box: Add Notification, Remove Notification, and Edit Notification. You can use these buttons to configure the notifications tab by adding, deleting, and editing notifications.

Task Assignment Tab

Figure 11–26 shows the Task Assignment tab of the Task Details dialog box.

Figure 11–26 Task Assignment Tab

This tab lets you add task assignment rules for the current task. It provides options to add the rules, assignment type, whom the task must be assigned to, adapter, e-mail template, and escalation time. The added rules are displayed in a tree based on the priority. The shortcut menu that is displayed when you right-click the rule provides options to change the priority of the rule, and to edit or delete the rule.

When you click **Add Task Assignment Rule**, the Task Assignment Rule dialog box opens with different input fields needed for assignments, as shown in [Figure 11–27](#).

Figure 11–27 Task Assignment Rule Dialog Box

Task Assignment Rule

Provide Task Assignment Values

Rule Name *

Assignment Type *

Assign To [Clear](#)

Adapter [Clear](#)

Email Template [Clear](#)

Send Email

Escalation Time (ms)

* Indicates Required Field

[Add](#) [Close](#)

The Task Assignment Rule dialog box provides the following options:

- **Rule Name:** A lookup field with a list of the rules.
- **Assignment Types:** A lookup field with the following options for assignment types:
 - Object Administrator User with Least Load
 - Group User with Least Load
 - Request Target Users Manager
 - Object Administrator
 - User
 - Object Authorizer User with Least Load
 - Requestor's Manager
 - Group
- **Assign To:** A lookup field. The values of this field vary with the selection in the Assignment Types field. Therefore, the value selected in the Assignment Types field is validated first.
- **Adapter:** A lookup field that brings up a list of the available task assignment adapters.
- **Email Template:** A lookup field that opens a dialog box with a list of e-mail templates from which to choose.
- **Send Email:** A check box. When this is selected, Oracle Identity Manager sends the e-mail notification to a user or role after the current process task is assigned.
- **Escalation Time (ms):** A text field to specify the amount of time (in milliseconds) in which the user or role has to complete the process task. The user or role is

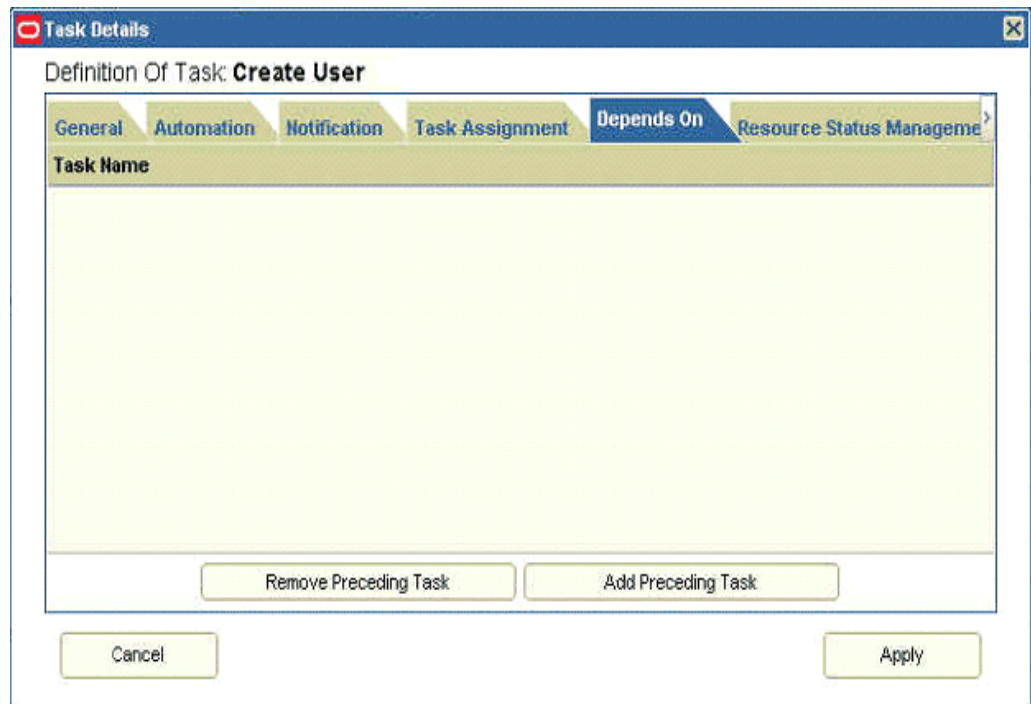
associated with the rule that Oracle Identity Manager triggers. If this process task is not completed within the allotted time, then Oracle Identity Manager reassigns it to another user or role. The escalation rule adheres to the order defined by the assignment type parameter.

When an assignment rule is created, it is displayed in the Task Assignment tab of the Task Details dialog box with a tree structure.

Depends On Tab

Figure 11–28 shows the Depends On tab of the Task Details dialog box.

Figure 11–28 Task Details Dialog Box



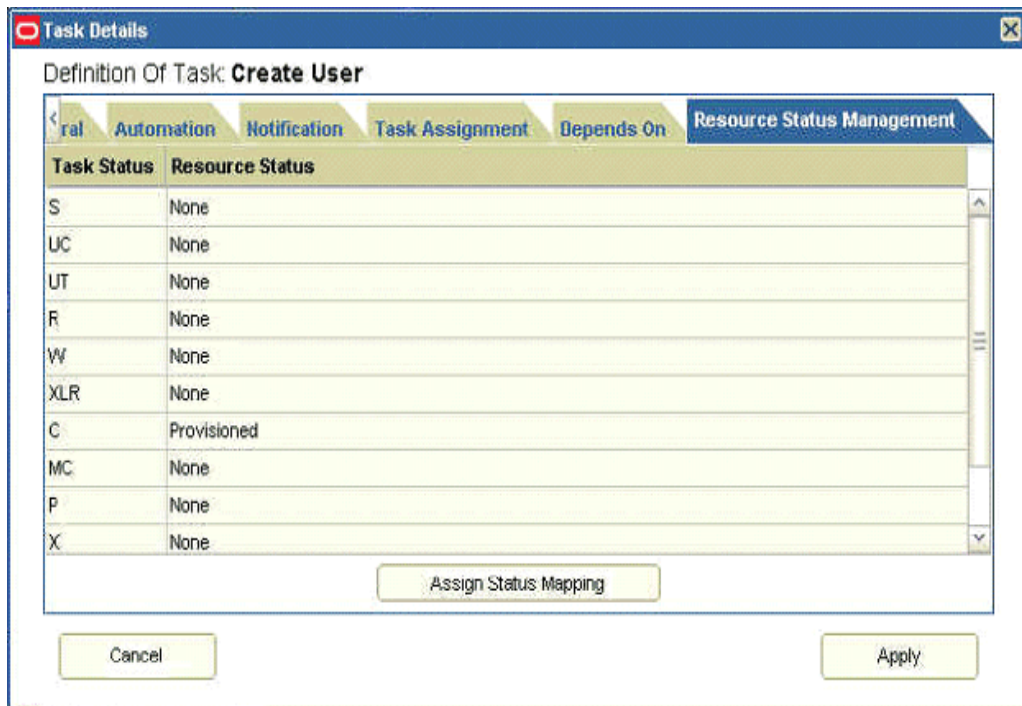
This tab lets you add tasks that the current task will depend on. This is useful in setting up dependencies between tasks. This dialog box consists of buttons to add and remove tasks from this list. Any task in this list must be run before the current task is run.

When you click **Add Preceding Task**, the Assign Preceding Task dialog box is displayed. This dialog box lists the tasks and the corresponding workflows in which they are used. You can select a task from this list and click **OK**.

When you select a task from the list and click **Remove Preceding Task**, the task is removed from the list.

Resource Status Management Tab

Figure 11–29 shows the Resource Status Management tab of the Task Details dialog box.

Figure 11–29 Resource Status Management Tab

This tab lets you establish a link between the status of a process task and the provisioning status of the resource object to which it is assigned.

A resource object contains data that is used to provision resources to users and applications. In addition, a resource object is provided with predefined provisioning statuses. Provisioning status changes through the life cycle of the resource object after the provisioning kicks off. The provisioning status represents the various statuses of the resource object throughout its lifecycle when it is provisioned to the target user or organization. The provisioning status of a resource object is determined by the status of its associated provisioning processes, as well as the tasks that comprise these processes. For this reason, a link between the status of a process task and the provisioning status of the resource object to which it is assigned must be provided.

This tab provides two columns that display the tasks status and the resource status. When no mappings are done, the list under the resource status column has a value of **None** for all task status. When you click **Assign Status Mapping**, the Object Status dialog box is displayed. This dialog box has the list of resource statuses from which to select and map to the task status.

After you make changes on all the tabs of the Task Details dialog box, click **Apply** to apply all changes to the task. Alternatively, click **Cancel** to cancel the operation.

11.6.4.6 Configuring Responses

You can double-click a response icon to open the Response Details dialog box that provides options to configure the response. [Figure 11–30](#) shows the Response Details dialog box.

Figure 11–30 Response Details Dialog Box

The Response Details dialog box has the following fields:

- **Response Code:** This field is used to specify the response code. This code for the response uniquely identifies a response for a task.
- **Response Status:** This lookup field is used to select the response status, such as Cancelled, Completed, or Rejected.
- **Response Description:** This field is used to provide a description of the response.

After you specify the response configuration information, click **Update Response** to apply the input for the response. In the designer page, the response code is displayed in the response icon.

11.6.5 Configuring Data Flows

Data flows are used for transferring data to the workflow form fields without the need for the user to enter information. This is used for both provisioning and reconciliation. For provisioning, form data flows are used. For reconciliation, reconciliation data flows are used.

In reconciliation data flows, the flow is from reconciliation fields to workflow fields instead of between resource fields and workflow fields. For a trusted resource, the user attributes are displayed instead of the workflow form fields.

The Configure Reconciliation Data Flows page is used to define the relationship between the data elements in the target resource or trusted source and the fields within Oracle Identity Manager with which they are to be linked.

Only the fields defined in the Reconciliation Fields section of the associated resource are available for mappings. These mappings are used to determine which fields in Oracle Identity Manager must be populated with the information provided by using reconciliation events from the target system. In addition, for target resources, the key fields are indicated on this tab. Key fields are fields for which the values on the process form and the reconciliation event must be the same for a match to be generated on the Processes Matched Tree tab of the Reconciliation Manager form.

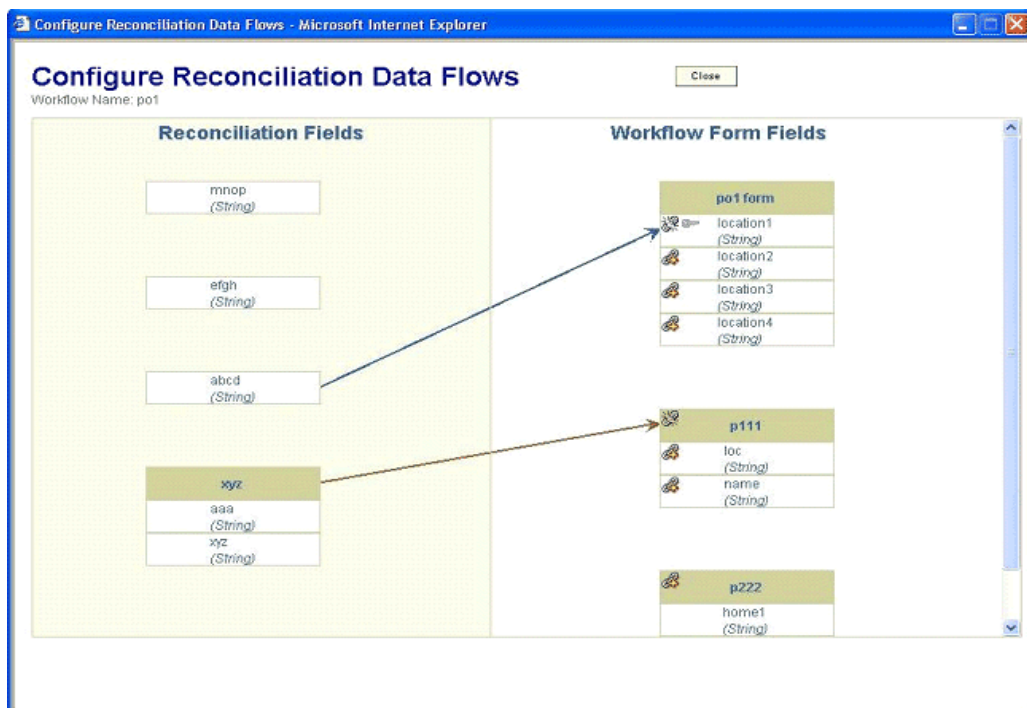
Note: The reconciliation fields created in the Reconciliation Fields tab of the associated resource can be of the types Multi-Valued, String, Number, Date, and IT resource.

You configure reconciliation data flow on the Configure Reconciliation Data Flows page. The reconciliation data flow rules are as follows:

- When a workflow form field or child table is mapped to a reconciliation field, it cannot be mapped to another field unless the first one is removed.
- Each reconciliation field can be mapped only once.

Figure 11–31 shows the Configure Reconciliation Data Flows page.

Figure 11–31 Configure Reconciliation Data Flows Page



An additional property for reconciliation data flows is called the Key Reconciliation field. Each workflow field that is mapped for data flow can be set as a key field for reconciliation. This means that the reconciliation rules corresponding to this field must be met. This is represented in the form of a disabled key icon next to an established data flow. By default, each field is not a key field. To set a field as a key field, click the key icon. Click the key icon again to remove the key field setting.

Clicking the key icon sets the field as a key field, and the icon changes to an enable key icon. Clicking the icon again removes the field as a key field.

See Also:

- [Chapter 28, "Customizing Oracle Identity Manager Interfaces"](#) for information about customizing the Workflow Designer
- [Chapter 31, "Using APIs"](#) for information about the API methods used by the Workflow Designer

11.7 Creating IT Resources

To create an IT resource:

1. In the Welcome page of the Advanced Administration, under Configuration, click **Create IT Resource**.

Alternatively, click the **Configuration** tab, click **Resource Management**, and then select **Create IT Resource**.

2. On the Step 1: Provide IT Resource Information page, enter the following information:

- **IT Resource Name:** Enter a name for the IT resource.
- **IT Resource Type:** Select an IT resource type for the IT resource.

If you want to create an IT resource of the Remote Manager type, then select **Remote Manager** from the **IT Resource Type** list.

- **Remote Manager:** If you want to associate the IT resource with a particular remote manager, then select the remote manager from this list. If you do not want to associate the IT resource with a remote manager, then leave this field blank.

Note: If you select **Remote Manager** from the **IT Resource Type** list, then you must not select a remote manager from the **Remote Manager** list.

3. Click **Continue**.
4. On the Step 2: Specify IT Resource Parameter Values page, specify values for the parameters of the IT resource, and then click **Continue**.
5. On the Step 3: Set Access Permission to IT Resource page, if you want to assign roles to the IT resource and set access permissions for the roles, then:
 - a. Click **Assign Role**.
 - b. For the roles that you want to assign to the IT resource, select **Assign** and the access permissions that you want to set. For example, if you want to assign the `ALL USERS` role and set the Read and Write permissions to this role, then you must select the respective check boxes in the row, as well as the Assign check box, for this role.
 - c. Click **Assign**.
6. On the Step 3: Set Access Permission to IT Resource page, if you want to modify the access permissions of roles assigned to the IT resource, then:

Note: You cannot modify the access permissions of the `SYSTEM ADMINISTRATORS` role. You can modify the access permissions of only other roles that you assign to the IT resource.

- a. Click **Update Permissions**.
- b. Depending on whether you want to set or remove specific access permissions for roles displayed on this page, select or deselect the corresponding check boxes.
- c. Click **Update**.

7. On the Step 3: Set Access Permission to IT Resource page, if you want to unassign a role from the IT resource, then:

Note: You cannot unassign the `SYSTEM ADMINISTRATORS` role. You can unassign only other roles that you assign to the IT resource.

- a. Select the **Unassign** check box for the role that you want to unassign.
- b. Click **Unassign**.
8. Click **Continue**.
9. On the Step 4: Verify IT Resource Details page, review the information that you provided on the first, second, and third pages. If you want to make changes in the data entered on any page, click **Back** to revisit the page and then make the required changes.
10. To proceed with the creation of the IT resource, click **Continue**.
11. The Step 5: IT Resource Connection Result page displays the results of a connectivity test that is run using the IT resource information. If the test is successful, then click **Create**. If the test fails, then you can perform one of the following steps:
 - Click **Back** to revisit the previous pages and then make corrections in the IT resource creation information.
 - Click **Cancel** to stop the procedure, and then begin from the first step onward.
 - Proceed with the creation process by clicking **Continue**. You can fix the problem later, and then rerun the connectivity test by using the Diagnostic Dashboard.

Note: If no errors are encountered, then the label of the button is **Create**, not **Continue**.

See "Test Basic Connectivity" on page 16-11 for more information.

12. Click **Finish**.

11.8 Managing IT Resources

To locate an IT resource:

1. In the Welcome page of the Advanced Administration, under Configuration, click **Manage IT Resource**.

Alternatively, click the **Configuration** tab, click **Resource Management**, and then select **Manage IT Resource**.

2. On the Manage IT Resource page, you can use one of the following search options to locate the IT resource that you want to view:
 - IT Resource Name: Enter the name of the IT resource, and then click **Search**.
 - IT Resource Type: Select the IT resource type of the IT resource, and then click **Search**.
 - Click **Search**.

On the Manage IT Resource page, the list of IT resources that meet the search criteria is displayed.

From this point onward, you can perform one of the following procedures on the IT resource:

- [Viewing IT Resources](#)
- [Modifying IT Resources](#)
- [Deleting IT Resources](#)

11.8.1 Viewing IT Resources

To view an IT resource:

1. From the list of IT resources displayed in the search results, click the IT resource name.

Note: If you want to edit the IT resource, then click the edit icon in the same row.

2. If you want to view the IT resource parameters and their values, then select **Details and Parameters** from the list at the top of the page. Similarly, if you want to view the administrative roles assigned to the IT resource, then select **Administrative Roles** from the list.

11.8.2 Modifying IT Resources

To modify an IT resource:

1. From the list of IT resources displayed in the search results, click the edit icon for the IT resource that you want to modify.
2. If you want to modify values of the IT resource parameters, then:
 - a. Select **Details and Parameters** from the list at the top of the page.
 - b. Make the required changes in the parameter values.
 - c. To save the changes, click **Update**.
3. If you want to modify the administrative roles assigned to the IT resource, first select **Administrative Roles** from the list at the top of the page and then perform the required modification.
4. If you want to unassign an administrative role, select the **Unassign** check box in the row in which the role name is displayed and then click **Unassign**.

Note:

- When you click **Unassign**, the administrative roles that you select are immediately unassigned from the IT resource. You are not prompted to confirm that you want to unassign the selected administrative roles.
 - You cannot unassign the `SYSTEM ADMINISTRATORS` role.
-
-

5. If you want to assign new administrative roles to the IT resource, then:

- a. Click **Assign Role**.
 - b. For the administrative roles that you want to assign to the IT resource, select the access permission check boxes and the **Assign** check box.
 - c. Click **Assign**.
6. If you want to modify the access permissions of the administrative roles that are currently assigned to the IT resource, then:
- a. Click **Update Permissions**.
 - b. Depending on the changes that you want to make, select or deselect the check boxes in the table.

Note: You cannot change the access permissions of the `SYSTEM ADMINISTRATORS` role.

- c. To save the changes, click **Update**.

11.8.3 Deleting IT Resources

To delete an IT resource:

1. From the list of IT resources displayed in the search results, click the Delete icon for the IT resource that you want to delete.
2. To confirm that you want to delete the IT resource, click **Confirm Delete**.

11.9 Managing Resources By Using the Design Console

This chapter describes resource management in the Design Console. It contains the following sections:

- [Overview of Resource Management](#)
- [IT Resources Type Definition Form](#)
- [Rule Designer Form](#)
- [Resource Objects Form](#)
- [Service Account Management](#)

11.9.1 Overview of Resource Management

The Resource Management folder provides you with tools to manage Oracle Identity Manager resources. This folder contains the following forms:

- **IT Resources Type Definition:** Use this form to create resource types that are displayed as lookup values on the IT Resources form.
- **Rule Designer:** Use this form to create rules that can be applied to password policy selection, automatic role membership, provisioning process selection, task assignment, and prepopulating adapters.
- **Resource Objects:** Use this form to create and manage resource objects. These objects represent resources that you want to make available to users and organizations.

See Also: See [Chapter 2, "Developing Adapters"](#) and [Chapter 3, "Using Adapters"](#) for more information about adapters and adapter tasks

11.9.2 IT Resources Type Definition Form

The IT Resources Type Definition form is in the Resource Management folder. You use the IT Resources Type Definition form to classify IT resource types, for example, AD, Microsoft Exchange, and Solaris. Oracle Identity Manager associates resource types with resource objects that it provisions to users and organizations.

After you define an IT resource type on this form, it is available for selection when you define an IT resource. The type is displayed in the Create IT Resource and Manage IT Resource pages of Advanced Administration.

IT resource types are templates for the IT resource definitions that reference them. If an IT resource definition references an IT resource type, the resource inherits all of the parameters and values in the IT resource type. The IT resource type is the general IT classification, for example, Solaris. The resource is an instance of the type, for example, Solaris for Statewide Investments.

You must associate every IT resource definition with an IT resource type.

[Figure 11–32](#) shows the IT Resources Type Definition form.

Figure 11–32 The IT Resources Type Definition Form

[Table 11–8](#) describes the fields of the IT Resources Type Definition form.

Table 11–8 Fields of the IT Resources Type Definition Form

Field Name	Description
Server Type	The name of the IT resource type
Insert Multiple	Specifies whether or not this IT resource type can be referenced by more than one IT resource

Note: If an IT resource must access an external resource but is not able to do so by using the network, you must associate it with a remote manager. For more information, see "Installing and Configuring a Remote Manager" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager*.

11.9.2.1 Defining a Template (a Resource Type) for IT Resources

To define an IT resource type:

1. Enter the name of the IT resource type in the **Server Type** field, for example, Solaris.
2. To make the IT resource type available for multiple IT resources, select **Insert Multiple**.
3. Click **Save**.

The IT resource type is defined. You can select it when defining IT resources in the Create IT Resource page of Advanced Administration.

11.9.2.2 Tabs on the IT Resource Type Definition Form

After you save the basic information for a new IT resource type, and when an IT resource type is returned on a query, the fields on the tabs of the IT Resources Type Definition form's lower region are enabled.

The IT Resources Type Definition form contains the following tabs:

- IT Resource Type Parameter tab
- IT Resource tab

11.9.2.2.1 IT Resource Type Parameter Tab You use the IT Resource Type Parameter tab to specify default values and encryption settings for all connection parameters for the IT resource type, as shown in [Figure 11-32](#). Oracle recommends that you do not specify default values for passwords and encrypted fields. Parameters and values on this tab are inherited by all IT resources that reference this IT resource type.

When you define a new parameter, the parameter and its values and encryption settings are added to the current IT resource type and to any new or existing IT resource definitions that reference this IT resource type. For an applicable resource definition, the new parameter is displayed in the **Details and Parameters** section of the Create IT Resource and Manage IT Resource pages of Advanced Administration.

Note: You can customize the values and encryption settings for these parameters within each IT resource.

Adding a Parameter to an IT Resource Type

To add a parameter to an IT Resource Type:

1. Click **Add**.
A new row is displayed in the **IT Resource Type Parameter** tab.
2. In the **Field Name** field, enter the name of the parameter.
3. In the **Default Field Value** field, enter a default value.

This value is inherited by all IT resources that reference this IT resource type

4. Select or clear the **Encrypted** option.

This check box determines if this parameter's value is masked, that is, represented with asterisk (*) in a form field.

If you want the parameter's value to be masked, select this check box.

5. Click **Save**.

Removing a Parameter from an IT Resource Type

To remove a parameter from an IT Resource Type:

1. Select the parameter you want to remove.
2. Click **Delete**.

The parameter and its associated value are removed from the IT resource type and from IT resource definitions that reference this type.

11.9.2.2 IT Resource Tab This tab displays IT resources that reference a selected IT resource type. All IT resources on this tab share the same parameters, but the values can be unique for each IT resource.

11.9.2.3 IT Resource Type Definition Table

The IT Resource Type Definition Table displays the following information:

Field Name	Description
Server Type	The name of the resource asset type, as defined in the IT Resource Type Definition form
Insert Multiple	Indicates whether or not multiple instances of this IT Resource Definition can be created

11.9.3 Rule Designer Form

Rules are criteria that enable Oracle Identity Manager to match conditions and take action based on them. A rule can be assigned to a specific resource object or process, or a rule can apply to all resource objects or processes.

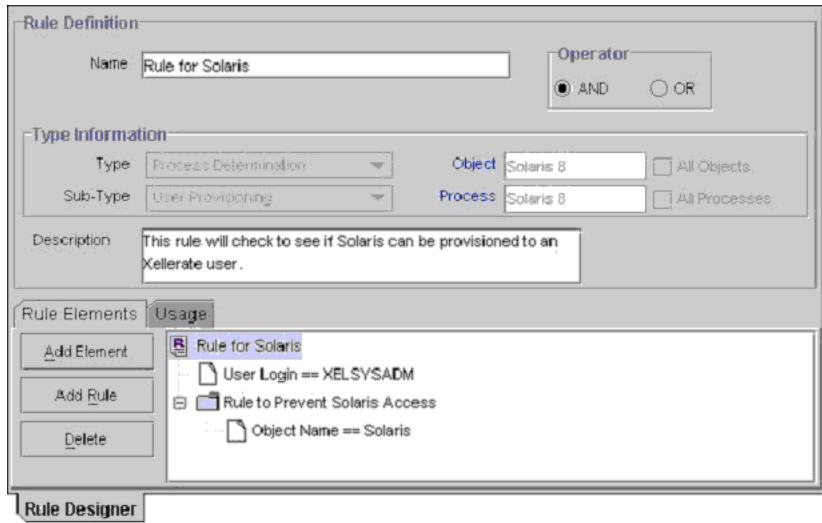
The following are examples of rule usage:

- Determining a password policy to apply to a resource object of type Application.
- Enabling users to be added to roles automatically.
- Specifying the provisioning process that apply to a resource object after that resource object is assigned to a request.
- Determining how a process task is assigned to a user.
- Specifying which prepopulate adapter is executed for a given form field.

See Also: *Oracle Identity Manager Tools Reference* for more information about prepopulate adapters

The Rule Designer form shown in [Figure 11–33](#) is in the Resource Management folder. You use this form to create and manage rules that are used with resources.

Figure 11–33 Rule Designer Form



There are four types of rules:

General: Enables Oracle Identity Manager to add a user to a role automatically and to determine the password policy that is assigned to a resource object.

Process Determination: Determines the provisioning processes for a for a resource object.

Task Assignment: Specifies the user or role that is assigned to a process task.

Prepopulate: Determines which prepopulate adapter is executed for a form field.

A rule contains the following items:

A rule element: Consists of an attribute, an operator, and a value. In [Figure 11–33](#), the attribute is `User Login`, the operator is `==`, and the value is `XELSYSADM`.

A nested rule: If one rule must be placed inside another rule for logic purposes, the internal rule is known as a nested rule. In [Figure 11–33](#), a **Rule to Prevent Solaris Access** is nested in a **Rule for Solaris**.

An operation: When a rule contains multiple rule elements or nested rules, an operation shows the relationship among the components. In [Figure 11–33](#), if the **AND** operation is selected, the `User Login==XELSYSADM` rule element and the **Rule to Prevent Solaris Access** nested rule must both be true for the rule to be successful.

[Table 11–9](#) describes the fields of the Rule Designer form.

Table 11–9 Fields of the Rule Designer Form

Field Name	Description
Name	The rule's name.

Table 11–9 (Cont.) Fields of the Rule Designer Form

Field Name	Description
AND/OR	<p>These options specify the operation for the rule.</p> <p>To stipulate that a rule is successful only when all the outer rule elements and nested rules are true, select AND. To indicate that a rule is successful if any of its outer rule elements or nested rules are TRUE, select OR.</p> <p>Important: These options do not reflect the operations for rule elements that are contained within nested rules. In Figure 11–33, the AND operation applies to the <code>User Login == XELSYSADM</code> rule element and the <code>Rule to Prevent Solaris Access</code> nested rule. However, this operation has no effect on the <code>Object Name != Solaris</code> rule element within the <code>Rule to Prevent Solaris Access</code> rule.</p>
Type	<p>The rule's classification status. A rule can belong to one of four types:</p> <ul style="list-style-type: none"> ■ General: Enables Oracle Identity Manager to add a user to a role automatically and determines the password policy that is assigned to a resource object. ■ Process Determination: Determines the provisioning processes for a resource object. ■ Task Assignment: Determines which user or role is assigned to a process task. ■ Prepopulate: Determines which prepopulate adapter is used for a form field.
Sub-Type	<p>A rule of type Process Determination, Task Assignment, or Prepopulate can be categorized into one of four subtypes:</p> <ul style="list-style-type: none"> ■ Organization Provisioning: Classifies the rule as a provisioning rule. Determines the organization for which a process is provisioned, a task is assigned, or the prepopulate adapter is applied. ■ User Provisioning: Classifies the rule as a provisioning rule. Determines the user for which a process is provisioned, a task is assigned, or a prepopulate adapter is applied. <p>For Task Assignment or Prepopulate rule types, the approval and standard approval items are not displayed in the Sub-Type box. The Sub-Type box is grayed out for the General rule type.</p>
Object	The resource object to which this rule is assigned.
All Objects	If selected, the rule can be assigned to all resource objects.
Process	The process to which this rule is assigned.
All Processes	If selected, the rule can be assigned to all processes.
Description	Explanatory information about the rule.

11.9.3.1 Creating a Rule

To create a rule:

Note: In the following procedure, note that the options do not apply to rule elements within nested rules. For example, in [Figure 11–33](#) the **AND** operation applies to the `User Login==XELSYSADM` rule element and the `Rule to Prevent Solaris Access` nested rule. But this operation has no effect on the `Object Name != Solaris` rule element in the `Rule to Prevent Solaris Access` rule.

1. Open the Rule Designer form.
2. In the **Name** field, enter the name of the rule.
3. To stipulate that a rule is successful only when all of its rule elements or nested rules are true, select the **AND** option.
To indicate that a rule is successful if any of its rule elements or nested rules are true, select the **OR** option.
4. Click the **Type** box, and in the custom menu select the classification status (**General**, **Process Determination**, **Task Assignment**, or **Prepopulate**) to associate with the rule.
For **Process Determination**, click **Sub-Type** and select the classification status (**Organizational Provisioning**, **User Provisioning**, **Approval**, or **Standard Approval**) to associate with the rule.
For **Task Assignment** or **Prepopulate**, click **Sub-Type** and select the classification status (**Organization Provisioning** or **User Provisioning**) to associate with the rule.
If you select **General** from the **Type** box, go to Step 7.
5. To associate the rule with a single resource object, double-click the **Object** lookup field, and in the Lookup dialog box select a resource object.
If you want the rule to be available to all resource objects, select the **All Objects** option.
6. To assign a rule to one process, double-click the **Process** lookup field, and from the Lookup dialog box, select the process to associate with the rule.

Note: The only processes that are displayed in this Lookup window are the ones that are associated with the resource object you selected in Step 5.

If you want the rule to be available to all processes, select the **All Processes** option.

Note: If you select a resource object in Step 5 by selecting the **All Processes** option, this rule is available to every process that is associated with the selected resource object.

7. In the **Description** field, enter explanatory information about the rule.
8. Click **Save**.

11.9.3.2 Tabs on the Rule Designer Form

The Rule Designer form contains the following tabs:

- Rule Elements tab
- Usage tab

Each of these tabs is discussed in the following sections.

11.9.3.2.1 Rule Elements Tab From this tab, you can create and manage elements and nested rules for a rule. For example, in [Figure 11-34](#), the Rule for Solaris contains the `User Login==XELSYSADM` rule element. It also has a nested Rule to

Prevent Solaris Access. [Figure 11–34](#) displays the Rule Elements tab of the Rule Designer form.

Figure 11–34 Rule Elements Tab of the Rule Designer Form

The rule in [Figure 11–34](#) can be applied to a provisioning process for the Solaris resource object. After this resource object is assigned to a request, the rule is triggered. If the target user's login is XELSYSADM, and the name of the resource object is Solaris, the Solaris resource object is provisioned to the user. Otherwise, the user cannot access Solaris.

When a rule element or nested rule is no longer valid, remove it from the rule.

The following procedures describe how to:

- Add a rule element to a rule
- Add a nested rule to a rule
- Remove a rule element or nested rule from a rule

Adding a Rule Element to a Rule

To add a rule element to a rule:

1. Click **Add Element**.

The Edit Rule Element dialog box is displayed.

The custom menus in the boxes on the Edit Rule Element dialog box reflect the items in the **Type** and **Sub-Type** boxes of the Rule Designer form.

[Table 11–10](#) describes the data fields in the Edit Rule Element dialog box.

Table 11–10 Fields of the Edit Rule Element Dialog Box

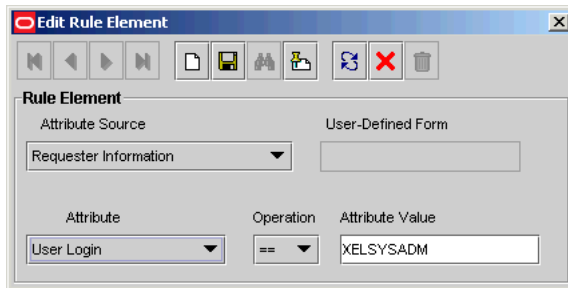
Name	Description
Attribute Source	From this box, select the source of the attribute. For example, if the attribute you wish to select is Object Name, the attribute source to select would be Object Information.

Table 11–10 (Cont.) Fields of the Edit Rule Element Dialog Box

Name	Description
User-Defined Form	This field displays the user-created form that is associated with the attribute source that is displayed in the adjacent box. Note: If Process Data are not displayed in the Attribute Source box, the User-Defined Form field will be empty.
Attribute	From this box, select the attribute for the rule.
Operation	From this box, select the relationship between the attribute and the attribute value (== or !=)
Attribute Value	In this field, enter the value for the attribute. Note: The attribute's value is case-sensitive.

- Set the parameters for the rule you are creating, as shown in [Figure 11–35](#).

Figure 11–35 Edit Rule Element Window



In this example, if the Login ID of the target user is XELSYSADM, the rule element is `true`. Otherwise, it is `false`.

See Also: For more information about the parameters, see "[Rule Elements Tab](#)" on page 11-54.

- From the Toolbar of the Edit Rule Element dialog box, click **Save**, and click **Close**.
The rule element is displayed in the **Rule Elements** tab of the Rule Designer form.
- From the main screen's toolbar, click **Save**.
The rule element is added to the rule.

Adding a Nested Rule to a Rule

To nest a rule within a rule:

Note: In the following procedure, only rules of the same type and subtype as the parent rule are displayed in the Select Rule window.

- Click **Add Rule**.
The Select Rule dialog box is displayed.
- Select a nested rule and click **Save**.
- Click **Close**.

The nested rule is displayed in the **Rule Elements** tab of the Rule Designer form.

4. From the main screen's Toolbar, click **Save**.

The nested rule is added to the rule.

Removing a Rule Element or Nested Rule from a Rule

To remove a rule element or a nested rule:

1. Select the rule element or nested rule that you want to remove.
2. Click **Delete**.

The rule element or nested rule is removed from the rule.

11.9.3.2.2 Usage Tab This tab is displayed on the Rule Designer form. The information in the Usage tab reflects the rule's classification type. For example, if a rule type is prepopulate, the user-created field that this rule is applied to is displayed in this tab.

Figure 11–36 shows the Usage tab.

Figure 11–36 Usage Tab of the Rule Designer Form

The screenshot shows the 'Usage' tab of the Rule Designer form. The 'Rule Definition' section includes the name 'Rule to Approve Solaris' and the operator 'OR'. The 'Type Information' section shows the type as 'Process Determination', sub-type as 'Approval', object as 'is Resource Object', and process as 'to Approve Solaris'. The description is 'This rule will determine whether the target user can approve the provisioning of the Solaris resource object.' Below this, the 'Usage' tab is active, displaying a table with one row of data.

	Object	Process	Type	Priority
1	The Solaris Resource Object	Process to Approve Solaris	A	1

This tab displays the following items:

- The password policy, resource object, process, process task, auto-role membership criteria, role, Oracle Identity Manager form field, and prepopulate adapter associated with a rule.
- A one-letter code, signifying the rule's classification type: P=Provisioning. This code is displayed for process determination rules only.
- The rule's priority number.

11.9.3.3 Rule Designer Table

The Rule Designer Table, as shown in Figure 11–37, displays all available rules defined in the Rule Designer form.

Figure 11–37 Rule Designer Table

Rule Name	Rule Type	Rule Sub-Type	Operator	Description	Last Updated
User Edit Task Assign	Process Determin...	General: Approval	AND		007-004-3107-04
User Edit Role	Process Determin...	General: Approval	AND		007-004-3107-04
Call Process Task	Pre-Populate	User Provisioning	AND	This rule is a user...	007-004-3107-04
User List Role	Process Determin...	User Provisioning	AND	This rule is a single...	007-004-3107-04
Confirmation Rule	Process Determin...	General: Approval	AND	Confirmation Rule to...	007-004-3107-04
User ID	Pre-Populate	User Provisioning	AND	This rule is a user...	007-004-3107-04
IS Role Process Task	Task Assignment	User Provisioning	AND	This rule is a user...	007-004-3107-04

Table 11–11 shows the information displayed in the Rule Designer Table.

Table 11–11 Information in the Rule Designer Table

Field Name	Description
Rule Name	The name of the rule.
Rule Type	<p>A rule can belong to one of four types:</p> <ul style="list-style-type: none"> ■ General: Enables Oracle Identity Manager to add a user to a role automatically and determines the password policy that is assigned to a resource object. ■ Process Determination: Determines the provisioning processes that are selected for a resource object. ■ Task Assignment: Determines which user, role, or both are assigned to a process task. ■ Pre-Populate: Determines which prepopulate adapter is executed for a given form field.
Rule Sub-Type	<p>A rule of type Process Determination, Task Assignment, or Pre-Populate can be categorized into one of four sub-types:</p> <ul style="list-style-type: none"> ■ Organization Provisioning: Classifies the rule as a provisioning rule. You use this subtype to determine the organization for which a process is provisioned, a task is assigned, or the prepopulate adapter is applied. ■ User Provisioning: Classifies the rule as a provisioning rule. You use this subtype to determine the user for which a process is provisioned, a task is assigned, or a pre-populate adapter is applied.
Rule Operator	The relationship between the attribute and the attribute value represented by the == or != operators.
Description	Explanatory information about the rule.
Last Updated	The date when the rule was last updated.

11.9.4 Resource Objects Form

The Resource Objects form is in the Resource Management folder. You use this form to create and manage the resource objects for the Oracle Identity Manager resources that you want to provision for organizations or users. Resource object definitions are templates for provisioning the resource. However, the provisioning of the resource depends on the design of the provisioning processes that you link to the resource object.

Table 11–12 describes the data fields of the Resource Objects form.

Table 11–12 Fields of the Resource Objects Form

Field Name	Description
Table Name	The name of the resource object form that is associated with this resource. (This is actually the name of the table that represents the form.)
Order For User/Order For Organization	Options that determine whether or not the resource object can be requested for users or organizations. To request the resource object for a user, select Order For User . To request the resource object for an organization, select Order For Organization .
Type	The resource object's classification status. A resource object can belong to one of three types: <ul style="list-style-type: none"> ■ Application: Classifies this resource object as an application. ■ Generic: Contains business-related processes. ■ System: Oracle Identity Manager uses this type of resource object internally. Do not modify system resource objects without first consulting Oracle.
Allow Multiple	Designates if the resource is provisioned more than once to a user or organization. If it is selected, the resource object can be provisioned more than once for each user or organization.
Self Request Allowed	By selecting this check box, users as well as the system administrator can request the resource object for themselves. Note: The resources allowed for self request can be further requested at the request template level.
Allow All	By selecting this check box, the resource object can be requested for all Oracle users. This setting takes precedence over whether or not the organization to which a user belongs has allowed the resource that can be requested for its users.
Provision by Object Admin Only	This check box determines who can provision this resource. If this check box is selected, only users who are members of the roles listed on the Object Administrators tab will be able to provision this resource object (either directly or by manually initiating the provisioning process from the request). If this check box is deselected, no restrictions are placed on who can directly provision this resource.

Table 11–12 (Cont.) Fields of the Resource Objects Form

Field Name	Description
Sequence Recon	<p>If you select this check box, reconciliation events are processed in the sequence in which they are created.</p> <p>The application of this feature can be illustrated by the following example:</p> <p>Suppose there are two reconciliation events for the OIM User resource object for user John Doe. The first reconciliation event (E1) data is as follows:</p> <ul style="list-style-type: none"> ■ Login: testuser1 ■ First Name: John ■ Last Name: Doe ■ Organization: Xellerate Users ■ Type: End-User ■ Role: Full-Time <p>The second reconciliation event (E2) data is as follows:</p> <ul style="list-style-type: none"> ■ Login: testuser1 ■ First Name: John1 ■ Last Name: Doe1 ■ Organization: Xellerate Users ■ Type: End-User ■ Role: Full-Time <p>Between the first and second events, the first name and last name of the user was changed.</p> <p>During trusted source reconciliation, if events are processed in the order in which they are created, this change in first and last names is correctly reconciled into Oracle Identity Manager. However, if the second event is processed before the first one, data in the target system does not match data in Oracle Identity Manager at the end of the reconciliation run. This inconsistency will be reflected in the auditing tables, and will remain until another event from the trusted source is created for this user.</p> <p>If you enable the Sequence Recon option, you can ensure that events for the same entity (for example, same user or same process form) are processed in the order in which they were created.</p>
Trusted Source	<p>You can select this check box if you want to use the resource object for trusted user reconciliation.</p> <p>By default, this check box is not selected. It is selected by default only for the Xellerate User resource object.</p>

11.9.4.1 Creating a Resource Object

To create a resource object:

1. Open the Resource Objects form.
2. In the **Name** field, enter the name of the resource object.
3. To request the resource object for a user, select **Order For User**.
To request the resource object for an organization, select **Order For Organization**.

Note: A resource object can be requested for either one user or one organization.

4. Double-click the **Type** lookup field.
From the Lookup dialog box that is displayed, select the classification status (**Application**, **Generic**, or **System**) to associate with the resource object.
5. If you want multiple instances of the resource object to be requested for a user or an organization, select the **Allow Multiple** option. Otherwise, go to Step 6.
6. If you want to be able to request the resource object for yourself, select the **Self Request Allowed** option. Otherwise, go to Step 7.
7. To provision the resource object for all users, regardless of whether the organization to which the user belongs has the resource object assigned to it, select the **Allow All** check box. Otherwise, go to Step 8.
8. If you want to use the resource object for trusted source user reconciliation, you must select the **Trusted Source** option. Otherwise, go to Step 9.

Note: You must deselect the Self Request Allowed and Allow All check boxes to ensure that the resource object is not available for provisioning requests and resource profiles.

9. To restrict the roles that can provision this resource object to roles that are displayed in the **Object Administrators** tab of the Resource Objects form, select the **Provision by Object Admin Only** option. This applies to resource objects that are provisioned directly or by assignment to a request. Otherwise, go to Step 10.
10. Click **Save**.

The resource object is created.

11.9.4.2 Tabs on the Resource Objects Form

When you start the Resource Objects form and create a resource object, the tabs of this form become functional.

The Resource Objects form contains the following tabs:

- [Depends On Tab](#)
- [Object Authorizers Tab](#)
- [Process Determination Rules Tab](#)
- [Event Handlers/Adapters Tab](#)
- [Resource Audit Objectives](#)
- [Status Definition Tab](#)
- [Administrators Tab](#)
- [Password Policies Rule Tab](#)
- [User-Defined Fields Tab](#)
- [Process Tab](#)
- [Object Reconciliation Tab](#)

11.9.4.2.1 Depends On Tab From this tab, you can select resource objects that Oracle Identity Manager must provision before provisioning the current resource object. If Oracle Identity Manager can provision the current resource object without first provisioning a resource object that is displayed on the **Depends On** tab, you must remove that resource object from the tab.

The following topics are related to the Depends On tab:

- Selecting a resource object on which the current resource object is dependent
- Removing the dependent resource object

Selecting a Dependent Resource Object

To select a dependent resource object:

1. Click **Assign**.

The Assignment dialog box is displayed.

2. Select the resource object.

3. Click **OK**.

The dependent resource object is selected.

Removing a Dependent Resource Object

To remove a dependent resource object:

1. Select the dependent resource object that you want to remove.
2. Click **Delete**.

The resource object is removed from the **Depends On** tab.

11.9.4.2.2 Object Authorizers Tab Use this tab to specify roles that are the object authorizers for this resource. You can select users who are members of the Object Authorizers roles as targets for task assignments.

Each role on the Object Authorizers tab has a priority number. The priority number can also be referenced when a task assigned to a role is escalated due to lack of action. You can increase or decrease the priority number for any role on this tab.

For example, suppose that you configure members of the SYSTEM ADMINISTRATORS roles to be object authorizers. Also suppose that a process task associated with this resource object has a task assignment rule attached to it. The first user authorized to complete this process task is the user with the priority number **1**. If the user does not complete the process task in a user-specified time, Oracle Identity Manager reassigns the task to the user with the next priority in the SYSTEM ADMINISTRATORS role.

See Also: ["Rule Designer Form"](#) on page 11-51 and ["Assignment Tab of the Editing Task Window"](#) on page 12-29 for more information about task assignment rules and process tasks

Assigning a Role to a Resource Object

To assign a role to a resource object:

1. Click **Assign**.

The Assignment dialog box is displayed.

2. Select a role.

3. Click **OK**.

The role is selected.

Removing a Role from a Resource Object

To remove a role from a resource object:

1. Select the desired role.
2. Click **Delete**.

The role is removed from the **Object Authorizers** tab.

11.9.4.2.3 Process Determination Rules Tab A resource object is a template for the resource that is provisioned to users or organizations. This template can be linked to multiple provisioning processes. Oracle Identity Manager uses process determination rules to select a provisioning process when a resource is requested or directly provisioned.

Process determination rules provide the following criteria:

- Which provisioning process to select when a resource is requested
- Which provisioning process to select when a resource is provisioned directly

Each provisioning process has a process determination rule. Each rule and process combination has a priority number that indicates the order in which Oracle Identity Manager will evaluate it.

If the condition of a rule is false, Oracle Identity Manager evaluates the rule with the next highest priority. If a rule is true, Oracle Identity Manager executes the process associated with it.

Adding a Process Determination Rule to a Resource Object

To add a process determination rule to a resource object:

1. Click **Add** in the **Provisioning Processes** region, depending on the rule or process combination you intend to create.
2. From the row that is displayed, double-click the **Rules** lookup field.
3. From the Lookup dialog box, select a rule, and assign it to the resource object (only rules of *Process Determination* type are available for selection).
4. Click **OK**.
5. In the adjacent column, double-click the **Processes** lookup field.
6. From the Lookup dialog box, select a process, and assign it to the rule.
7. Click **OK**.
8. Enter a numeric value in the **Priority** field.

This determines the order in which Oracle Identity Manager evaluates the rule and process combination.

9. Click **Save**.

The rule and process combination is added to the resource object.

Remove a Process Determination Rule From a Resource Object

To remove a process determination rule from a resource object:

1. Select a rule and process combination.

2. Click **Delete**.

The rule and process combination is removed from the resource object.

11.9.4.2.4 Event Handlers/Adapters Tab A resource object's provisioning process contains tasks that must be completed automatically. When this occurs, you must assign an event handler or an adapter to the resource object. An event handler is a software routine that provides the processing of this specialized information. An adapter is a specialized type of event handler that generates Java code, which enables Oracle Identity Manager to communicate and interact with external resources.

When an event handler or adapter that is assigned to a resource object that is no longer valid, you must remove it from the resource object.

For this example, the **adpAUTOMATEPROVISIONINGPROCESS** adapter was assigned to the **Solaris** resource object. Once this resource object is assigned to a request, Oracle Identity Manager triggers the adapter, and the associated provisioning process is executed automatically.

Assigning an Event Handler or Adapter to a Resource Object

To assign an event handler to an adapter or a resource object:

1. Click **Assign**.

The Assignment dialog box is displayed.

2. Select an event handler, and assign it to the resource object.
3. Click **OK**.

The event handler is assigned to the resource object.

Remove an Event Handler or Adapter from a Resource Object

To remove an event handler or adapter from a resource object, perform the following steps:

1. Select an event handler.
2. Click **Delete**.

The event handler is removed from the resource object.

11.9.4.2.5 Resource Audit Objectives The Resource Objects form in the Design Console includes a resource attribute named **Resource Audit Objectives**. This resource attribute helps you link resources to regulatory mandates.

Figure 11–38 The Resource Objects Form

A lookup is defined for the values of the Resource Audit Objectives resource attribute. The predefined values in the Resource Audit Objectives list are:

- SOX (Hosts Financially Significant Information)
- HIPAA (Hosts Private Healthcare Information)
- GLB (Hosts Non-Public Information)
- Requires Quarterly Review
- Requires Annual Review

You can extend this list by editing the Lookups.Resource Audit Objective.Type lookup by using the Lookup Definition Form in the Design Console.

11.9.4.2.6 Status Definition Tab You use this tab to set provisioning status for a resource object. A provisioning status indicates the status of a resource object throughout its lifecycle, until it is provisioned to the target user or organization.

Every provisioning status of a resource object is associated with a task status for the relevant provisioning process. Oracle Identity Manager selects the provisioning process when the resource object is assigned to a request. For example, if the **Provision for Developers** process is selected, and a task in this process achieves **Completed** status, the corresponding status of the resource object can be set to **Provisioned**. This way, you can see how the resource object relates to the provisioning process, quickly and easily.

A resource object has the following predefined statuses:

- **Waiting:** This resource object depends on other resource objects that have not yet been provisioned.
- **Revoked:** The resources represented by the resource object are provisioned to target users or organizations that have been permanently deprovisioned from using the resources.
- **Ready:** This resource object either does not depend on any other resource objects, or all resource objects upon which this resource object depends are provisioned.
After a resource is assigned to a request and the resource object's status is **Ready**, Oracle Identity Manager evaluates the process determination rules to determine the provisioning process. When this happens, the status of the resource object changes to **Provisioning**.
- **Provisioning:** The resource object is assigned to a request and a provisioning process has been selected.
- **Provisioned:** The resources represented by the resource object are provisioned to the target users or organizations.
- **Provide Information:** Additional information is required before the resources represented by the resource object can be provisioned to the target users or organizations.
- **None:** This status does not represent the provisioning status of the resource object. Rather, it signifies that a task that belongs to the provisioning process that Oracle Identity Manager selects has no effect on the status of the resource object.
- **Enabled:** The resources represented by the resource object are provisioned to the target users or organizations, and these users or organizations have access to the resources.
- **Disabled:** The resources represented by the resource object are provisioned to the target users or organizations, but these users or organizations have temporarily lost access to the resources.

Each provisioning status has a corresponding Launch Dependent check box. If the check box is selected and if the parent resource object achieves that provisioning status, then Oracle Identity Manager will continue the provisioning of the dependent resource object.

For example, suppose that the Exchange resource object depends on Active Directory and has the Launch Dependent check box selected for the Provisioned and Enabled provisioning statuses. When the provisioning status of Active Directory changes to Provisioned or Enabled, and if Exchange provisioning is waiting on it, then Oracle Identity Manager will continue the provisioning process of Exchange.

You might want to add additional provisioning statuses to a resource object to reflect the various task statuses of a provisioning process. For example, when the status of a task that belongs to a provisioning process is **Rejected**, you might want to set the corresponding provisioning status of the resource object to **Revoked**.

Similarly, when an existing provisioning status is no longer valid, you must remove it from the resource object.

The following sections discuss how to add a provisioning status to a resource object and remove a provisioning status from a resource object.

Adding a Provisioning Status to a Resource Object

To add a provisioning status to a resource object:

1. Click **Add**.
2. Add a provisioning status in the **Status** field.
3. When you want other, dependent resource objects to launch their own provisioning process once the resource object achieves the provisioning status you are adding, select the **Launch Dependent** check box. Otherwise, go to Step 4.
4. Click **Save**.

The provisioning status is added to the resource object.

Removing a Provisioning Status from a Resource Object

The following procedure describes removing a provisioning status from a resource object:

1. Select a provisioning status.

2. Click **Delete**.

The provisioning status is removed from the resource object.

11.9.4.2.7 Administrators Tab This tab is used to select roles that can view, modify, and delete the current resource object.

When the **Write** check box is selected, the corresponding role can modify the current resource object. When the **Delete** check box is selected, the associated role can delete the current resource object.

The following sections describe how to assign a role to a resource object, and remove a role from a resource object.

Assigning a Role to a Resource Object

To assign a role to a resource object:

1. Click **Assign**.

The Assignment dialog box is displayed.

2. Select the role, and assign it to the resource object.

3. Click **OK**.

The role is displayed in the **Administrators** tab. By default, all members of this role can view the active record.

4. If you want this role to be able to modify the current resource object, select the corresponding **Write** check box.

Otherwise, go to Step 5.

5. If you want this role to be able to delete the current resource object, select the associated **Delete** check box.

Otherwise, go to Step 6.

6. Click **Save**.

The role is assigned to the resource object.

Tip: If you want to assign a permission for provisioning resource objects to users other than members of the SYSTEM ADMINISTRATORS role, then perform the following steps:

1. Using the Administrative and User Console, add the resource object to Resources tab for the Xellerate Users organization.
2. Assign a role as the administrator for the resource object.
3. Make the same role as the Administrative Role of Xellerate Users organization.
4. Assign the Manage Users menu item to the role so that the role members are able to perform provisioning.
5. In the Design Console, make the role as administrators of various Oracle Identity Manager entities, such as process definition or form, associated with the resource object.

For example, if Role A is the administrator of the AD resource object, then add Role A to the administrator/authorizer tab of each Oracle Identity Manager entity associated with the AD resource object.

6. In the Resource Objects form, select the following options:
 - Provision by Object Admin Only: Selecting this is mandatory. See [Table 11-12, "Fields of the Resource Objects Form"](#) for information about this option.
 - Any other option as required.

Removing a Role from a Resource Object

To remove a role from a resource object:

1. Highlight the role that you want to remove.
2. Click **Delete**.

The role is removed from the resource object.

11.9.4.2.8 Password Policies Rule Tab If a resource object is of type Application, and you want to provision the resource object to a user or organization, you might want that user or organization to meet password criteria before accessing the resource object. This password criteria is created and managed in the form of password policies. These policies are created by using the Password Policies form.

Because the resource object definition is only a template for governing how a resource is to be provisioned, Oracle Identity Manager must be able to make determinations about how to provision the resource based on actual conditions and rules. These conditions might not be known until the resource is actually requested. Therefore, rules must be linked to the various processes and password policies associated with a resource. This enables Oracle Identity Manager to decide which ones to invoke in any given context.

Oracle Identity Manager determines which password policy to apply to the resource when creating or updating a particular user's account. This is done by evaluating the password policy rules of the resource and applying the criteria of the policy associated with the first rule that is satisfied. Each rule has a priority number, which indicates the order in which Oracle Identity Manager will evaluate it.

The following sections discuss how to add and remove a password policy rule from a resource object.

Adding a Password Policy Rule to a Resource Object

To add a password policy rule to a resource object:

1. Click **Add**.
2. From the row that is displayed, double-click the **Rule** lookup field.
3. From the Lookup dialog box, select a rule, and assign it to the resource object.
4. Click **OK**.
5. In the adjacent column, double-click the **Policy** lookup field.
6. From the Lookup dialog box, select an associated password policy, and assign it to the resource object.
7. Click **OK**.
8. Add a numeric value in the **Priority** field.
This field contains the rule's priority number.
9. Click **Save**.

The password policy rule is added to the resource object.

Note:

- If the resource type is Order for Organization, you cannot attach a password policy to the resource object. The exception to this rule is the Xellerate User resource object. Although this resource object is of Order for Organization type, password policies can be attached to it.
 - If two or more rules evaluate to True, the password policy attached to the rule with the highest priority is applied.
 - A Default rule is predefined in Oracle Identity Manager. This rule always evaluates to True. If no rules have been created through the Rule Designer, a password policy can be attached to the Default rule.
-
-

Removing a Password Policy Rule from a Resource Object

To remove a password policy from a resource object:

1. Select a password policy rule.
2. Click **Delete**.

The password policy rule is removed from the resource object.

11.9.4.2.9 User-Defined Fields Tab You use this tab to view and access user-defined fields that were created for the Resource Objects form. After a user-defined field is created, it is displayed on this tab and can accept and supply data.

See Also: See "[User Defined Field Definition Form](#)" on page 15-5 for instructions about how to create user-defined fields on existing Oracle Identity Manager forms

11.9.4.2.10 Process Tab The **Process** tab displays all provisioning processes that are associated with the current resource object. The **Default** check boxes on this tab indicate what provisioning processes are the defaults for the resource.

Note: You create provisioning processes and associate them with a resource by using the Process Definition form. Each process can be linked to a process determination rule by using the **Process Determination Rules** tab of the Resource Object form.

For example, suppose that the Solaris resource object has one provisioning processes (Provision Solaris for Devel.) associated with it. The Provision Solaris for Devel. has been designated as the default provisioning process for this resource object.

11.9.4.2.11 Object Reconciliation Tab The Object Initial Reconciliation Date field on the Object Reconciliation Tab displays the date when initial reconciliation was performed for the resource.

Note: The purpose of initial reconciliation is to bring all the user accounts from the target system into Oracle Identity Manager.

The date value stored in the Object Initial Reconciliation Date field is used to distinguish between initial reconciliation and subsequent reconciliations events. This date value is used by the two exception reports introduced in release 9.1.0. These exception reports display differences in the entitlements a user must have as compared to what the user actually has in the target system. The differences in entitlements are determined by using reconciliation data, along with other data items. The exception reports return data associated with only those reconciliation events that are created after the date stored in the Object Initial Reconciliation Date field. In addition, exception data is generated only if the Initial Object Reconciliation Date field displays a date value that is in the past. If required, you can enter a date value in this field so that the exception reports are generated.

The Object Reconciliation tab contains two subtabs, Reconciliation Fields and Reconciliation Action Rules.

- The **Reconciliation Fields** tab is used to define the fields on the target resources or trusted sources that are to be reconciled with (for example, mapped to) information in Oracle Identity Manager
- The **Reconciliation Action Rules** tab is used to specify the actions Oracle Identity Manager is to take when particular matching conditions are met.

Click the **Create Reconciliation Profile** button in the Object Reconciliation tab to generate reconciliation profile whenever any changes are made to the resource object or associated process forms.

Reconciliation Fields Tab

This tab is used to define the fields on the target resources or trusted sources that are to be reconciled with (for example, mapped to) information in Oracle Identity Manager. For each field on the target system or trusted source, the following information will be listed:

- Name of the field on the target resource or trusted source that is to be reconciled with data in Oracle Identity Manager (for example, targetfield1)
- Data type associated with the field (for example, String). Possible values are multi-valued, string, number, date, IT resource

- Indicator that designates whether or not this field is required in a reconciliation event

Note: Oracle Identity Manager will not begin to match provisioning processes, users or organizations to the reconciliation event until all fields are processed on the Reconciliation section of the Event Management tab in the Advanced Administration.

The following is an example of a reconciliation field definition:

```
TargetField1 [String], Required
```

In the Reconciliation Fields tab, you can perform the following:

- Add a reconciliation field

The following procedure adds fields from the target system or trusted source to the list of fields that are to be reconciled with information in Oracle Identity Manager.

Note: Before Oracle Identity Manager can successfully perform reconciliation with an external target resource or trusted source, the fields you have defined on this tab must be mapped to the appropriate Oracle Identity Manager fields by using the **Field Mappings** tab of the resource's default provisioning process.

To add a reconciliation field:

1. Click **Add Field**.

The Add Reconciliation Field dialog box is displayed.

2. Enter the name of the field on the target resource or trusted source in the **Field Name** field.

This is the name that will reference the target resource or trusted source field in Oracle Identity Manager.

3. Select one of the following values from the menu in the **Field Type** field:

- Multi-Valued

This is meant for use with fields that contain one or more component fields.

- String

- String

- Date

- IT resource

During reconciliation event creation, the value this field receives must be the same as the name of an IT resource defined in Oracle Identity Manager.

4. Select the **Required** check box.

If selected, the reconciliation field must be processed on the Reconciliation section of the Event Management tab in the Advanced Administration before

Oracle Identity Manager will begin matching a provisioning process, user, or organization to the reconciliation event. If this check box is not selected, the inability to process this field in a reconciliation event will not prevent matching from occurring.

5. Click Save.

The field will be available for mapping in the resource's default provisioning process.

- **Delete a reconciliation field**

Use the following procedure to remove a target system field from the list of fields that are to be reconciled with information in Oracle Identity Manager. For a trusted source, this must be the user resource definition.

To delete a reconciliation field:

1. Select the field you wish to remove.
2. Click **Delete Field**.

The selected field will be removed from the list of fields with which Oracle Identity Manager reconciles data on the target system (this will have no effect on the data in the target system itself).

Reconciliation Action Rules Tab

By using this tab, you can specify the actions that Oracle Identity Manager will perform when some matches within reconciliation event records are encountered. Each record in this tab is a combination of:

- The matching condition criteria
- The action to be performed

The conditions and actions from which you can select are predefined. Depending on the matching conditions, certain actions might not be applicable. A complete list of the available options is provided in [Table 11-13](#).

Table 11-13 Rule Conditions and Possible Rule Actions

Rule Condition	Possible Rule Actions
No matches found	None Create User (only available with the trusted source)
One Process Match Found	None Establish Link
Multiple Process Matches Found	None
One Entity Match Found	None Establish Link
Multiple Entity Matches Found	None

See Also: ["Assignment Tab of the Editing Task Window"](#) on page 12-29 for a description of the classification types for the users and roles listed in the preceding table

Adding a Reconciliation Action Rule

To add a reconciliation action rule:

1. Click **Add Field**.

The **Add a new Action Rule** dialog box is displayed.

2. Select the desired value from the **Rule Condition** menu.

This is the matching condition that will cause the associated action to be executed. Each match condition can only be assigned to a single rule action.

3. Select a value from the **Rule Action** menu.

This is the action that will be executed if the matching condition is met.

4. Click **Save**, and close the Add a new Action Rule dialog box.

Deleting a Reconciliation Action Rule

To delete a reconciliation action rule:

1. Select the matching action combination to delete.

2. Click **Delete**.

The reconciliation action rule will be removed and the action associated with its condition will not be executed automatically.

11.9.4.3 Multiple Trusted Source Reconciliation

In earlier releases, you could set up only the Xellerate User resource object as a trusted source to reconcile identities. Now, you can do this by creating the reconciliation fields, reconciliation action rules, field mappings, and matching roles for the Xellerate User resource object and the process definition.

If there are two trusted sources from which you want to reconcile identities to create OIM Users, you are not able to configure a single resource object (Xellerate User) for both the trusted sources. Even if you create reconciliation fields for both the trusted sources in the Xellerate User resource object, you cannot create the corresponding reconciliation field mappings in the Xellerate User process definition.

From release 9.1.0 onward, you can configure resource objects other than Xellerate User as trusted sources for identity reconciliation. You can do this by selecting the **Trusted Source** check box in the Resource Objects form while creating a resource object.

For a resource object to which the Trusted Source flag is attached, you can create multiple reconciliation fields to denote the target system fields. You can also configure the reconciliation action rule in which if there are no process matches found, either a user is created or the data is sent to the administrator or authorizer for identity creation. If a process match is found, the link is established.

When defining provisioning process for trusted source resources, do not attach user-defined process forms. For these provisioning processes, reconciliation field mappings can be created between reconciliation fields defined on the resource and OIM User attributes.

Note: If the resource object is for target resource reconciliation, then the mapping is between the reconciliation fields and process data fields.

Do not use any resource objects that are defined as a trusted source for provisioning activities. These resources are meant to be used only for OIM Users' reconciliation.

Another addition in this release is the attribute authoritative sources feature. This means sources are trusted for only attributes of the identities and not the identities themselves. You can configure attribute authoritative source reconciliation by creating appropriate reconciliation action rules. If no process match is found, it is assigned to the administrator. This ensures that a user is not created by mistake even if there are no matches found. If a process match is found, the reconciliation action rule will establish a link.

The following sections discuss two use cases in which you can implement multiple trusted source reconciliation:

Note: At some places in this document:

- Multiple trusted source reconciliation has been referred to as MTS.
 - The terms fields and attributes have been used interchangeably.
-
-

- [Multiple Trusted Source Reconciliation Using MTS-Compatible Connectors](#)
- [Multiple Trusted Source Reconciliation Using Connectors That Are Not MTS-Compatible](#)

Note: For both use cases, create reconciliation profiles by referring to "Creating New Reconciliation Profiles" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager*.

11.9.4.3.1 Multiple Trusted Source Reconciliation Using MTS-Compatible Connectors

Note: To determine whether or not your connector is MTS-compatible, see connector-specific documentation.

The following sections discuss scenarios in which you can implement multiple trusted source reconciliation by using MTS-compatible connectors:

- [Configuring MTS-Compatible Connectors for Trusted Source Reconciliation by User Type](#)
- [Configuring MTS-Compatible Connectors for Trusted Source Reconciliation of Specific OIM User Attributes](#)

Configuring MTS-Compatible Connectors for Trusted Source Reconciliation by User Type

In this context, user type refers to the type of users whose records you want to reconcile. Examples of user types are `Employee` and `Customer`.

To implement trusted source reconciliation by user type, perform the procedure to implement trusted source reconciliation while deploying the connectors of each target system that you want to configure as a trusted source.

During reconciliation, all the target system records of the specified user types are reconciled. If the target systems contain multiple user types, you can use the Limited Reconciliation feature to specify the user type for which records must be reconciled from each target system.

Configuring MTS-Compatible Connectors for Trusted Source Reconciliation of Specific OIM User Attributes

You might want to configure trusted source reconciliation for specific OIM User attributes from multiple target systems. The procedure to implement this is described with the help of the following sample scenario:

You want to reconcile identities from one target system, for example TS1, and specific attributes of these identities (for example `attr1`, `attr2`, and `attr3`) from another target system, for example TS2. This means that TS1 is the trusted source for the identities, and TS2 is the trusted source for specific attributes of those identities and not the identities themselves. TS1 must provide all the mandatory OIM User attributes for the successful creation of an OIM User. TS2 will provide only those OIM User attributes (either a mandatory OIM User attribute or a non-mandatory one) for which TS2 is the trusted source. If you reconcile a mandatory OIM User attribute from TS2, the value of this attribute overwrites the value contained in this attribute after the OIM User is created from TS1. If you want to reconcile only non-mandatory OIM User attributes from TS2, you can choose not to reconcile these attributes from TS1 during OIM User creation.

Note: When there are multiple trusted sources, the logic to reconcile the entity attributes from the trusted sources is provided by the connector.

For the TS1 connector:

1. Perform all the steps required to deploy the TS1 connector and configure it for trusted source reconciliation.

See Also: The documentation for the connector you are deploying for information about the procedure to configure trusted source reconciliation

2. In the Reconciliation Fields tab on the Object Reconciliation page, delete all the TS1 attributes that you want to reconcile from TS2 (in this case `attr1`, `attr2`, and `attr3`).
3. In the Reconciliation Field Mappings tab on the Process Definition page, delete all the mappings other than the ones you want to retain.

Instead of deleting reconciliation fields, you can remove the reconciliation field mappings of those fields for which you do not want to reconcile the values into the OIM User created through reconciliation.

4. In the Reconciliation Action Rules tab on the Object Reconciliation page, ensure that the following rule condition and action mappings exist:

Rule Condition: No Matches Found

Action: Create User

For the TS2 connector:

1. Perform all the steps required to deploy the TS2 connector and configure it for trusted source reconciliation.

See Also: The documentation for the connector you are deploying for information about the procedure to configure trusted source reconciliation

2. In the Reconciliation Field Mappings tab on the Process Definition page, delete all the mappings other than the ones you want to retain.

Instead of deleting reconciliation fields, you can also choose to just remove the reconciliation field mappings of those fields for which you do not want to reconcile the values into the OIM User created through reconciliation.

3. In the Reconciliation Fields tab on the Object Reconciliation page, delete all the TS2 attributes other than `attr1`, `attr2`, and `attr3`. In addition, retain the attributes that you want to use to match OIM Users with existing TS2 accounts. This means that you retain only those attributes that will be used for reconciliation rule evaluation. For example, you might want to use the `username` attribute in Oracle Identity Manager to match the value of the `first name` attribute in TS1.
4. In the Reconciliation Action Rules tab on the Object Reconciliation page, create rule conditions and action mappings. One of these rule condition-action mappings must be the following:

Rule Condition: `No Matches Found`

Action: Anything other than `Create User`

11.9.4.3.2 Multiple Trusted Source Reconciliation Using Connectors That Are Not MTS-Compatible

Note: To determine whether or not your connector is MTS-compatible, see connector-specific documentation.

For a connector that is not MTS-compatible, the following prerequisites must be addressed before you can use the connector in a multiple trusted source reconciliation setup:

- i. Only one of the trusted source resource objects can be `Xellerate User`. In your operating environment, if the `Xellerate User` resource object is already in use by a connector for trusted source reconciliation, for the trusted source connector that you want to configure, you must create a new resource object and process definition.
- ii. The scheduled task of the connector must have an attribute that accepts the name of the resource object used for trusted source user reconciliation as its value.

The following sections discuss scenarios in which you can implement multiple trusted source reconciliation by using non-MTS-compatible connectors:

- [Configuring Non-MTS-Compatible Connectors for Trusted Source Reconciliation by User Type](#)
- [Configuring Non-MTS-Connectors for Trusted Source Reconciliation of Specific OIM User Attributes](#)

Configuring Non-MTS-Compatible Connectors for Trusted Source Reconciliation by User Type

In this context, user type refers to the type of users whose records you want to reconcile. Examples of user types are Contractor, Employee, and Customer.

You use Microsoft Active Directory and Oracle e-Business Suite as trusted sources in your operating environment. Active Directory is used to store information about identities that belong to the Contractor user type. Oracle e-Business Suite is used to store information about identities that belong to the Customer and Employee user type. You want to reconcile Contractor records from Active Directory and Employee records from Oracle e-Business Suite. To do this, perform the following:

For Active Directory:

1. Perform all the steps required to deploy the Active Directory connector and configure it for trusted source reconciliation.

See Also: The documentation for the connector you are deploying for information about the procedure to configure trusted source reconciliation

When you import the connector XML file for trusted source reconciliation, information specific to Active Directory is added in the `Xellerate User` resource object and process definition.

2. On the Resource Object tab, create the `ActDir` resource object for trusted source reconciliation with Active Directory.

Note: You can assign any name to the resource object. This procedure is based on the use of `ActDir` as the name assigned to the resource object.

For detailed information about the procedure to create a resource object, see "[Resource Objects Form](#)" on page 11-58.

While creating the resource object:

- a. Select the **Trusted Source** check box on the Resource Object tab.
- b. On the Object Reconciliation>>Reconciliation Fields tab, see `Xellerate User` resource object and add the Active Directory-specific fields that you want to reconcile in `ActDir`. All the mandatory OIM User fields must be covered by the fields that you add on this tab.
3. On the Object Reconciliation>>Reconciliation Action Rules tab, create rule conditions and action mappings. One of these rule condition-action mappings must be the following:

Rule Condition: No Matches Found

Action: Create User
4. Delete the fields specific to Active Directory and the corresponding rules from the `Xellerate User` resource object.
5. Create the `ActDir` process definition in the Process Definition form.

For detailed information about the procedure to create a process definition, see "[Process Definition Form](#)" on page 12-5. Based on the reconciliation field mappings

in the `Xellerate User` process definition, on the Reconciliation Field Mappings tab, add the reconciliation field mappings for the `ActDir` process definition.

6. Delete the Active Directory-specific field mappings in the `Xellerate User` resource object.
7. In the Reconciliation Rule Builder form on the Reconciliation Rules page, query and open the reconciliation rule for this connector and change the value of the Object field to map to the resource object that you have created. By default, the value of this field is mapped to that of the `Xellerate User` resource object.

For Oracle e-Business Suite, repeat all the steps you performed for Active Directory. Perform the following steps of that procedure differently for the Oracle e-Business Employee Reconciliation connector:

1. On the Resource Object tab, create the `EmpRecon` resource object for trusted source reconciliation with Oracle e-Business Suite.

Note: You can assign a name to the resource object. This procedure is based on the use of `EmpRecon` as the name assigned to the resource object.

2. On the Object Reconciliation>>Reconciliation Action Rules tab, create rule conditions and action mappings. One of these rule condition-action mappings must be the following:

Rule Condition: No Matches Found

Action: Create User

Use the Limited Reconciliation feature to specify that only identities that belongs to the Employee user type must be reconciled.

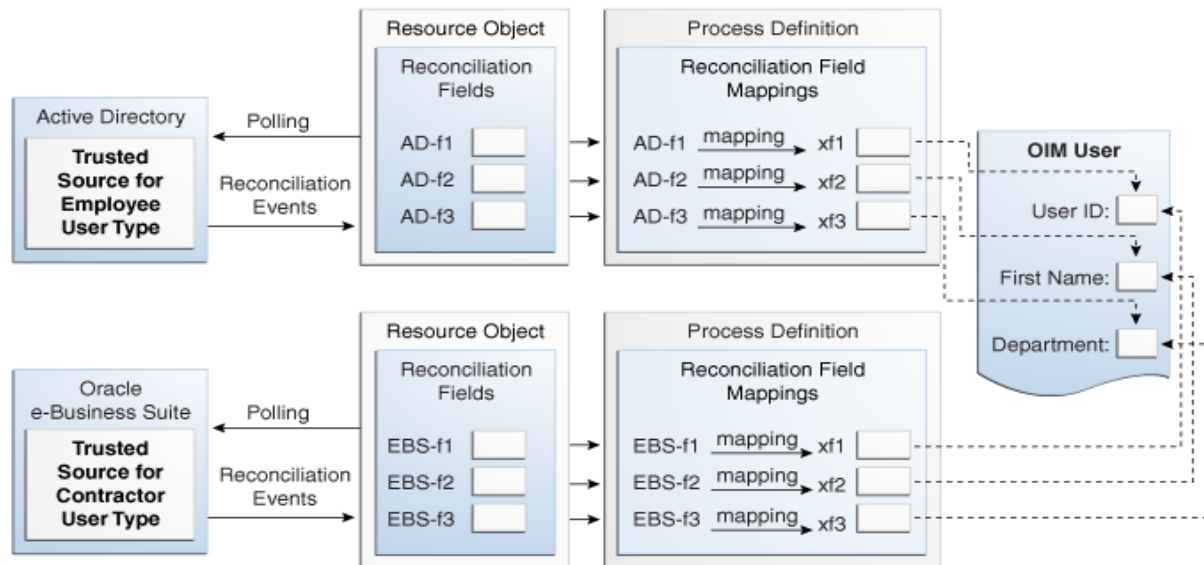
3. After you add the fields and the reconciliation rules, delete the Oracle e-Business Suite-specific fields and the corresponding rules created in the `Xellerate User` resource object.
4. Create the `EmpRecon` process definition in the Process Definition form. For detailed information about the procedure to create a process definition, see "[Process Definition Form](#)" on page 12-5. Based on the `Xellerate User` reconciliation field mappings, on the Reconciliation Field Mappings tab, add the field mappings for the `EmpRecon` process definition.
5. Delete the Oracle e-Business Suite-specific field mappings in the `Xellerate User` resource object.
6. On the Reconciliation Rules>>Reconciliation Rule Builder form, query and open the reconciliation rule for this connector and change the value of the Object field to map to the resource object that you have created. By default, the value of this field is mapped to that of the `Xellerate User` resource object.

For both Active Directory and Oracle e-Business Suite, perform the rest of the steps required to configure trusted source reconciliation. For example, while configuring the reconciliation scheduled task for each connector, specify the name of the trusted source resource object that must be used during trusted source user reconciliation.

The current value of the scheduled task attribute would be `Xellerate User` and it must be updated with the name of the new resource object configured for trusted source user reconciliation for this connector.

Figure 11–39 shows the design time implementation of trusted source reconciliation based on the user type.

Figure 11–39 Trusted Source Reconciliation by User Type



Configuring Non-MTS-Connectors for Trusted Source Reconciliation of Specific OIM User Attributes

You might want to configure trusted source reconciliation for specific OIM User attributes from multiple target systems. The procedure to implement this is described with the help of the following sample scenario:

You use Microsoft Active Directory and IBM Lotus Notes as your target systems. You want to reconcile identities from Active Directory and only the value of the `e-mail` attribute of each identity (reconciled into Oracle Identity Manager from Active Directory) from Lotus Notes. To achieve this:

For the Active Directory connector:

1. Perform all the steps required to deploy the Active Directory connector and configure it for trusted source reconciliation.

See Also: The documentation for the connector you are deploying for information about the procedure to configure trusted source reconciliation

When you import the connector XML file for trusted source reconciliation, Active Directory-specific information is added in the `Xe11erate User` resource object and process definition.

2. On the Resource Object tab, create the `ActDir` resource object for trusted source reconciliation with Active Directory.

Note:

You can assign any name to the resource object. This procedure is based on the use of `ActDir` as the name assigned to the resource object.

For detailed information about the procedure to create a resource object, see "[Resource Objects Form](#)" on page 11-58.

While creating the resource object:

- i. Select the **Trusted Source** check box on the Resource Object tab.
- ii. On the Object Reconciliation>>Reconciliation Fields tab, see `Xellerate User` resource object and add the Active Directory-specific fields that you want to reconcile in `ActDir`. All the mandatory OIM User fields must be covered by the fields that you add on this tab.
3. On the Object Reconciliation>>Reconciliation Action Rules tab, create rule conditions and action mappings. One of these rule condition-action mapping must be the following:
 - Rule Condition: No Matches Found
 - Action: Create User
4. Delete the Active Directory-specific fields and the corresponding rules from the `Xellerate User` resource object.
5. Create the `ActDir` process definition in the Process Definition form. For detailed information about the procedure to create a process definition, see "[Process Definition Form](#)" on page 12-5. Based on the reconciliation field mappings in the `Xellerate User` process definition, on the Reconciliation Field Mappings tab, create the field mappings for the `ActDir` process definition.
6. Delete the Active Directory-specific field mappings in the `Xellerate User` resource object.
7. On the Reconciliation Rules>>Reconciliation Rule Builder form, query and open the reconciliation rule for this connector and change the value of the Object field to map to the resource object that you have created. By default, the value of this field is mapped to that of the `Xellerate User` resource object.

For IBM Lotus Notes, repeat all the steps you performed for Active Directory. Perform the following steps of that procedure differently for the Lotus Notes connector:

1. On the Resource Object tab, create the `LotNotes` resource object for trusted source reconciliation with Lotus Notes.

Note: You can assign a name to the resource object. This procedure is based on the use of `LotNotes` as the name assigned to the resource object.

2. When you create the resource object, add only the `e-mail address` attribute.
3. On the Object Reconciliation>>Reconciliation Action Rules tab, create rule conditions and action mappings. Create any rule condition other than user creation if no matches are found. If a match is found, the link is established.

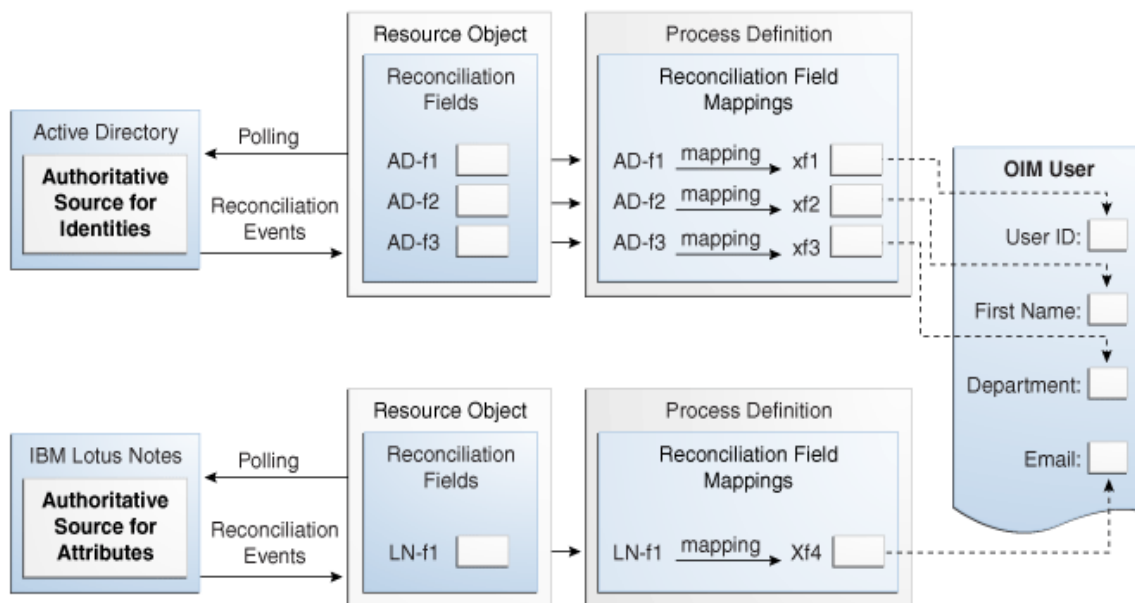
4. After you have added the fields and the reconciliation rules, delete the Lotus Notes-specific fields and the corresponding rules created in the `Xellerate User` resource object.
5. Create the `LotNotes` process definition in the Process Definition form. For detailed information about the procedure to create a process definition, see ["Process Definition Form"](#) on page 12-5. Based on the `Xellerate User` reconciliation field mappings, on the Reconciliation Field Mappings tab, add the field mappings for the `LotNotes` process definition.
6. Delete the Lotus Notes-specific field mappings in the `Xellerate User` resource object.

For both Active Directory and Lotus Notes, perform the rest of the steps required to configure trusted source reconciliation. For example, while configuring the reconciliation scheduled task for each connector, specify the name of the trusted source resource object that must be used during reconciliation.

The current value of the scheduled task attribute would be `Xellerate User` and it must be updated with the name of the new resource object configured for trusted source user reconciliation for this connector.

[Figure 11–40](#) shows the design time implementation of trusted source reconciliation of specific OIM User attributes.

Figure 11–40 Trusted Source Reconciliation for Specific OIM User Attributes



11.9.5 Service Account Management

Oracle Identity Manager supports service accounts. Service accounts are general administrator accounts (for example, `admin1`, `admin2`, `admin3`, and so on) that are used for maintenance purposes, and are typically shared by a set of users. The model for managing and provisioning service accounts is slightly different from normal provisioning.

Service accounts are requested, provisioned, and managed in the same manner as regular accounts. They use the same resource objects, provisioning processes, and

process forms as regular accounts. A service account is distinguished from a regular account by an internal flag.

When a user is provisioned with a service account, Oracle Identity Manager manages a mapping from the user's identity to the service account. When the resource is revoked, or the user gets deleted, the provisioning process for the service account does not get canceled (which would cause the undo tasks to start). Instead, a task is inserted into the provisioning process (the same way Oracle Identity Manager handles Disable and Enable actions). This task removes the mapping from the user to the service account, and returns the service account to the pool of available accounts.

This management capability is available through APIs.

Developing Provisioning Processes

This chapter describes process management with the Design Console. It contains the following topics:

- [Overview of Process Management](#)
- [Email Definition Form](#)
- [Process Definition Form](#)

12.1 Overview of Process Management

The Process Management folder provides you with tools for creating and managing Oracle Identity Manager processes and e-mail templates.

This folder contains the following forms:

- **Email Definition:** This form enables you to create templates for e-mail notifications.
- **Process Definition:** This form lets you create and manage provisioning processes. It also lets you start the Workflow Definition Renderer that displays your workflow definition graphically.

12.2 Email Definition Form

The Email Definition form, as shown in [Figure 12-1](#), is located in the Process Management folder. You use this form to create templates for e-mail notifications. These notifications can be set for sending to the user when:

- A task is assigned to the user.
- The task achieves a particular status.

Figure 12–1 Email Definition Form

You apply e-mail definitions through the **Assignment** tab of the Process Definition form.

12.2.1 Specifying the E-Mail Server

Before using the Email Definition form, you must specify the address of the e-mail server that Oracle Identity Manager will use to send e-mail notifications to users.

In Oracle Identity Manager 11g Release 1 (11.1.1), the e-mail server is specified by using the Administrative and User Console. To specify the e-mail server:

1. Login to the Administrative and User Console, and go to Advanced Administration.
2. Click the **System Management** tab, and then click **System Configuration**.
3. Search for the Email Server system property, and click the property to open the details of the property.
4. Ensure that the property name is set to the name of the resource asset instance that represents your e-mail server, and click **Save**.

Note: The value of the Email Server system property must be the e-mail server IT resource and not the hostname of the e-mail server.

5. In the Administrative and User Console, click Advanced, and then click System Management. Search for the Email Server system property. The value of this property is the Email Server IT resource that is associated with your e-mail server.

6. Once this IT resource is displayed, specify the IP address of the e-mail server and the name and password of the user who validates the usage of this server.

12.2.2 Email Definition Form

Table 12–1 describes the fields of the Email Definition form.

Table 12–1 Fields of the Email Definition Form

Field Name	Description
Name	The name of the e-mail definition.
Type	<p>This region contains three options for the following:</p> <ul style="list-style-type: none"> ■ Whether or not to categorize the e-mail definition as related to a request or a provisioning process ■ Whether or not to associate a variable for the e-mail definition with a request or a provisioning process ■ Whether or not to associate a variable for the e-mail definition with a general process <p>To classify the e-mail definition as a provisioning definition or to associate the e-mail variable with a provisioning process, select the Provisioning Related option.</p> <p>To categorize the e-mail definition as a general announcement, select the General option.</p>
Object Name	<p>From this lookup field, select the resource object that is associated with the provisioning process to which the e-mail definition is related.</p> <p>Note: Leave this lookup field empty to make the e-mail definition available for use with all resource objects.</p>
Process Name	<p>From this lookup field, select a provisioning process that was assigned to the selected resource object. This is the provisioning process to which the e-mail definition is to be related.</p> <p>Note: If the Provisioning Related option is not selected, both the Object Name and Process Name lookup fields are grayed out.</p>
Language	From this lookup field, select the language that is associated with the e-mail definition.
Region	From this lookup field, select the region that is associated with the language in the e-mail definition.
Targets	<p>Select the source of the variable for the e-mail definition. For example, if the variable you want to select is User Login, then the source to select is the User Profile Information.</p> <p>Note: The items that are displayed in this box reflect the options you selected from the Type region.</p>
Variables	From this box, select the variable for the e-mail definition, for example, User Login. The variables, which are displayed in this box, reflect the items you selected from the Targets box.
From	<p>Currently, two types of users can be selected from this box:</p> <ul style="list-style-type: none"> ■ Requester: The user who created the request. ■ User: Any Oracle User with an e-mail address, which is displayed in the Contact Information tab of their Users form.
User Login	<p>The ID of the user in the From region of the e-mail notification.</p> <p>Note: If the User item is not displayed in the From box, the User Login field is grayed out.</p>

Table 12–1 (Cont.) Fields of the Email Definition Form

Field Name	Description
Subject	The title of the e-mail definition.
Body	The content of the e-mail definition.

12.2.3 Creating an E-Mail Definition

To create an e-mail definition:

1. Open the Email Definition form.
2. In the **Name** field, enter the name of the e-mail definition.
3. If the e-mail definition is to be used with a provisioning process, select the **Provisioning Related** option.
4. Double-click the **Language** lookup field, and select a language to associate with this e-mail definition.
5. Double-click the **Region** lookup field, and select a region to associate with the e-mail definition language.

Note: E-mail notification is based on the locale that was specified when you first installed Oracle Identity Manager.

6. Click **Save**.

The remaining data fields of the Email Definition form are now operational.

7. To associate this e-mail definition with a particular resource object, double-click the **Object Name** lookup field in the Lookup dialog box. Then, select the resource object that is associated with the provisioning process to which this e-mail definition is related.

Leave this lookup field empty to make the e-mail definition available for use with all resource objects.

8. Double-click the **Process Name** lookup field.

From the Lookup dialog box, select a provisioning process that is assigned to the resource object you selected in Step 7. This is the provisioning process to which this e-mail definition is to be related.

Note: If the Provisioning Related option is not selected, both the Object Name and Process Name lookup fields are grayed out.

9. Click the **From** box.

From the custom menu that is displayed, select the type of the user (**Requester, User, or Manager of Provisioned User**) that is displayed in the From region of the e-mail notification.

Note: If the **Provisioning Related** option is not selected in Step 3, the **Manager of Provisioned User** item will not be displayed in the **From** box.

10. Optional. If you have selected the User option in the **From** box, double-click the **User Login** lookup field.

From the Lookup dialog box, select the user ID that is displayed in the From region of the e-mail notification.

If you did not select the User item in the From box, the User Login field is grayed out.

11. Add information in the **Subject** field.

This field contains the title of the e-mail definition.

12. Add information in the Body text area.

This text area contains the contents of the e-mail definition.

13. When necessary, populate the Subject field and Body text area with e-mail variables.

The following table describes the e-mail variables that you can customize for the e-mail definition.

Name	Description
Type	<p>These options specify if a variable for the e-mail definition will be related to a provisioning process.</p> <p>To associate the e-mail variable with a provisioning process, select the Provisioning Related option.</p>
Targets	<p>From this box, select the source of the variable for the e-mail definition. For example, if you want to use the User Login variable, the source to select will be User Profile Information.</p>
Variables	<p>From this box, select the variable for the e-mail definition, for example, User Login.</p>

Note: The items that are displayed in the custom menu of the **Targets** box reflect the selection of either the **Provisioning Related** or the **General** radio button. Similarly, the items that are displayed in the custom menu of the **Variables** box correspond to the items that are displayed in the **Targets**, **Location Types**, and **Contact Types** boxes.

14. Create an e-mail variable for the Subject field or Body text area.

15. Click **Save**.

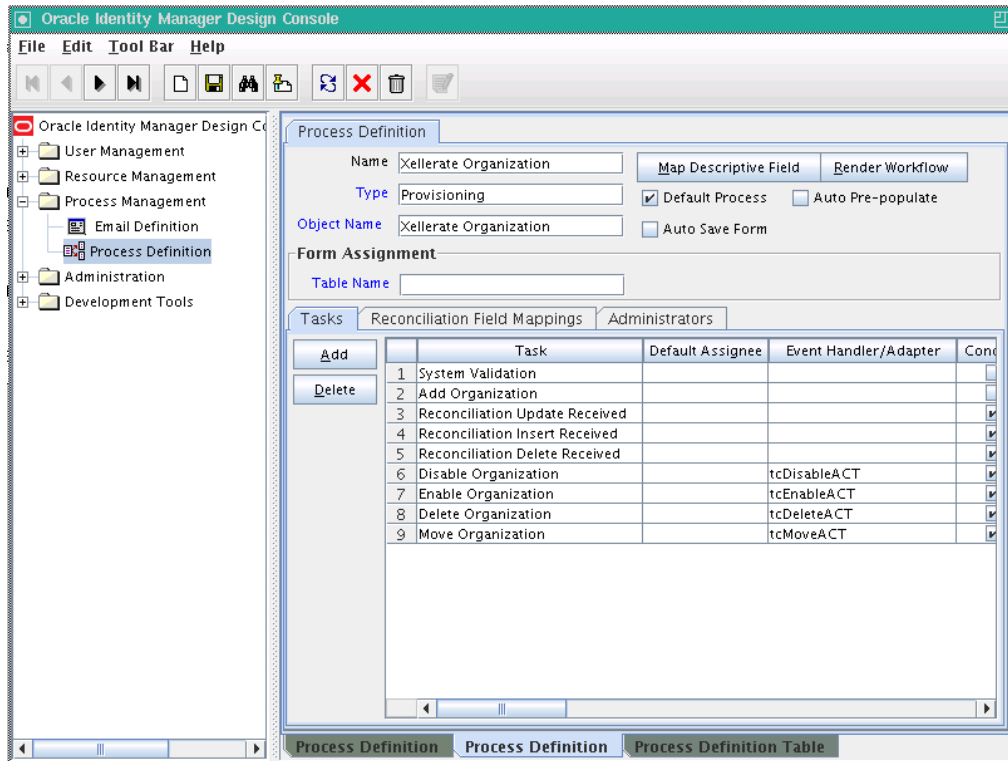
The e-mail definition is created.

12.3 Process Definition Form

A process is the mechanism for representing a logical workflow for provisioning in Oracle Identity Manager. Process definitions consist of tasks. Process tasks represent the steps that you must complete to fulfill the purpose of a process. For example, in a provisioning process, tasks are used to enable a user or organization to access the target resource.

The Process Definition form shown in [Figure 12-2](#) is in the Process Management folder. You use this form to create and manage the provisioning processes that you associate with your resource objects.

Figure 12–2 Process Definition Form



In Figure 12–2, the Xellerate Organization provisioning process is created and assigned to the resource object of the same name.

Note: Not all the form columns are captured in Figure 12–2; additional field columns extend on the right of the Tasks table.

Table 12–2 describes the fields of the Process Definition form.

Table 12–2 Fields of the Process Definition Form

Field Name	Description
Name	The name of the process.
Type	The classification type of the process definition.
Object Name	The name of the resource object to which the process will be assigned.
Map Descriptive Field	Click this button to select a field that will be used as an identifier of the process definition after an instance is assigned to a resource object.
Render Workflow	Click this button to start a Web browser and display the current workflow definition by using the Workflow Renderer tool.

Table 12–2 (Cont.) Fields of the Process Definition Form

Field Name	Description
Default Process	<p>This check box determines if the current process is the default provisioning process for the resource object with which it is associated.</p> <p>Select the check box to set the process as the default provisioning process for the resource object to which it is assigned. If you deselect the check box, the process will not be the default. It will only be invoked if a process selection rule causes it to be chosen.</p>
Auto Save Form	<p>This check box designates whether Oracle Identity Manager suppresses the display of the custom form associated with this provisioning process or display it and allow a user to supply it with data each time the process is instantiated.</p> <p>Select this check box to automatically save the data in the custom process form without displaying the form. If you select this check box, you must supply either system-defined data or ensure that an adapter is configured to populate the form with the required data because the user will not be able to access the form. Deselect this check box to display the custom process form and allow users to enter data into its fields.</p>
Auto Pre-Populate	<p>This check box designates whether the fields of a custom form are populated by Oracle Identity Manager or a user. Two types of forms are affected:</p> <ul style="list-style-type: none"> ■ Forms that are associated with the process ■ Forms that contain fields with prepopulated adapters attached to them <p>If the Auto Pre-Populate check box is selected, when the associated custom form is displayed, the fields that have prepopulate adapters attached to them will be populated by Oracle Identity Manager.</p> <p>When this check box is deselected, a user must populate these fields by clicking the Pre-Populate button on the toolbar or by manually entering the data.</p> <p>Note: This setting does not control the triggering of the prepopulate adapter. It only determines if the contents resulting from the execution of the adapter are displayed in the associated form field(s) because of Oracle Identity Manager or a user.</p> <p>For more information about prepopulate adapters, see "Working with Prepopulate Adapters" on page 3-9.</p> <p>Note: This check box is only relevant if you have created a process form that is to be associated with the process and prepopulate adapters are used with that form.</p>
Table Name	The name of the table that represents the form that is associated with the process definition.

12.3.1 Creating a Process Definition

To create a process definition:

1. Open the Process Definition form.
2. In the **Name** field, type the name of the process definition.
3. Double-click the **Type** lookup field.

From the Lookup dialog box that is displayed, select the classification type (Approval) of the process definition.

4. Double-click the **Object Name** lookup field.

From the Lookup dialog box that is displayed, select the resource object that will be associated with the process definition.

5. Optional. Select the **Default Process** check box to make this the default provisioning process for the resource object to which it is assigned.

If you do not want the current process definition to be the default, go to Step 6.

6. Optional. Select the **Auto Save Form** check box to suppress the display of the provisioning process' custom form and automatically save the data in it.

This setting is only applicable to provisioning processes.

To display provisioning process' custom form and solicit users for information, deselect this check box.

Note: If you select the **Auto Save Form** check box, ensure that all fields of the associated "custom" process form have adapters associated with them. However, a process form can have default data or object to the process data flow mapping or organization defaults.

For more information about adapters and their relationship with fields of custom forms, see [Chapter 3, "Using Adapters"](#).

7. If a custom form is to be associated with the process definition, this form contains fields that have prepopulate adapters attached to them, and you want these fields to be populated automatically by Oracle Identity Manager, select the **Auto Pre-Populate** check box.

If the fields of this form are to be populated manually (by a user clicking the **Pre-Populate** button on the Toolbar), deselect the **Auto Pre-Populate** check box.

Note: If the process definition has no custom form associated with it, or this form's fields have no pre-populate adapters attached to them, deselect the **Auto Pre-Populate** check box. For more information about prepopulate adapters, see "[Working with Prepopulate Adapters](#)" on page 3-9.

8. Double-click the **Table Name** lookup field.

From the Lookup window that is displayed, select the table that represents the form associated with the process definition.

9. Click **Save**.

The process definition is created and the **Map Descriptive Field** button is enabled. If you click this button, the Map Descriptive Field dialog box is displayed.

From this window, you can select the field (for example, the Organization Name field) that will be used as an identifier of the process definition when an instance of the process is assigned to a resource object. This field and its value will be displayed in the reconciliation Manger form.

See Also: If a process has a custom process form attached to it, the fields on that form will also be displayed in this window and be available for selection.

- Click the **Render Workflow** button to view your workflow definition in a graphical presentation.

The Workflow Renderer is a powerful tool in helping you develop your process definition.

Note: See *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager* for details about how to use the Workflow Definition Renderer

12.3.2 Tabs on the Process Definition Form

After you start the Process Definition form and create a process definition, the tabs of this form become functional.

The Process Definition form contains the following tabs:

- Tasks Tab
- Reconciliation Field Mappings Tab
- Administrators Tab

Each of these tabs is described in the following sections.

12.3.2.1 Tasks Tab

You use this tab to:

- Create and modify the process tasks that comprise the current process definition
- Remove a process task from the process definition (when it is no longer valid)

Figure 12–3 displays the Tasks tab of the Process Definition form.

Figure 12–3 Tasks Tab of the Process Definition Form

Add	Delete	Task	Default Assignee	Event Handler/Adapter	Cor
		1 System Validation			
		2 Reconciliation Update Received			
		3 Reconciliation Insert Received			
		4 Reconciliation Delete Received			
		5 Add User			
		6 Disable User		tcDisableUser	
		7 Enable User		tcEnableUser	
		8 Delete User		tcCompleteTask	
		9 Archive User Data			
		10 Move To New Organization			

See Also: See ["Modifying Process Tasks"](#) on page 12-15 for information about editing process tasks

12.3.2.1.1 Adding a Process Task

Process tasks represent the steps that you must complete in a process.

To add a process task:

1. Click **Add**.
The Creating New Task dialog box is displayed.
2. In the **Task Name** field, enter the name of the process task.
3. From the Toolbar of the Creating New Task window, click **Save**. Then, click **Close**.

The process task is added to the process definition.

12.3.2.1.2 Editing a Process Task

For instructions about how to edit and set process tasks, see ["Modifying Process Tasks"](#) on page 12-15.

12.3.2.1.3 Deleting a Process Task

To delete a process task:

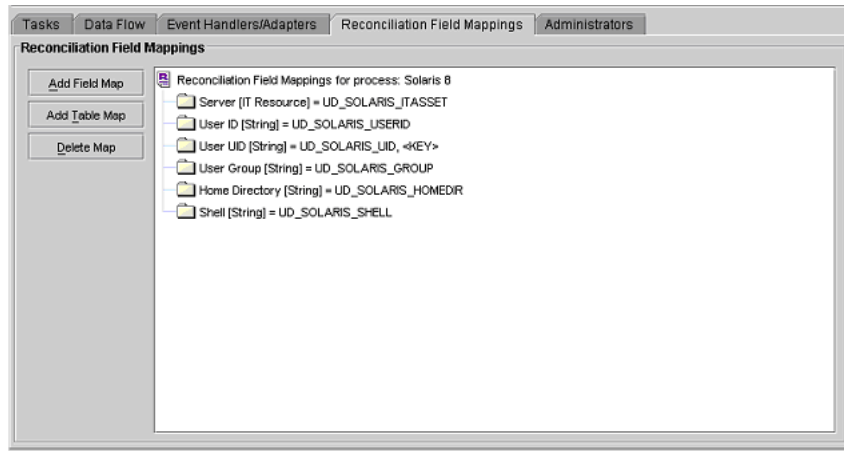
1. Select the process task that you want to delete.
2. Click **Delete**.

The process task is removed from the process definition.

12.3.2.2 Reconciliation Field Mappings Tab

You use the Reconciliation Field Mappings tab shown in [Figure 12-4](#) to define a relationship between data elements in a target system or trusted source and fields in Oracle Identity Manager.

Figure 12-4 Reconciliation Field Mappings Tab of the Process Definition Form



Only fields that you define in the **Reconciliation Fields** tab of the associated resource are available for mapping. Using a reconciliation event, these mappings determine which fields in Oracle Identity Manager to populate with information from the target

system. For target resources (not trusted sources), you can use this tab to indicate which fields are key fields. Key fields determine the values that must be same on the process form and the reconciliation event to generate a match on the **Processes Matched Tree** tab of the Reconciliation Manager form.

For each mapping, the following information is displayed:

- Name of the field, as defined on the **Reconciliation Fields** tab of the associated resource, on the target system or trusted source that is to be reconciled with data in Oracle Identity Manager.
- Data type associated with the field, as defined on the **Reconciliation Fields** tab of the associated resource.

Possible values are **Multi-Valued**, **String**, **Number**, **Date**, and **IT resource**.

- **For trusted sources:** For user discovery, mapping of the data in the trusted source field to the name of a field on the users form, or for organization discovery, mapping of the data in the trusted source field to the name of a field on the Oracle Identity Manager Organizations form.

If you are performing user and organization discovery with a trusted source, organization discovery must be conducted first.

See Also: "[Multiple Trusted Source Reconciliation](#)" on page 11-73 for information about how fields are mapped for multiple trusted source reconciliation

- **For target resources:** The name of the field on the resource's custom (provisioning) process form to which the data in the target resources field is to be mapped.
- **For target resources:** Indicator designating if the field is a key field in the reconciliation for this target resource.

For provisioning processes to match a reconciliation event data, the key field values in their process forms must be the same as those in the reconciliation event.

12.3.2.2.1 User Account Status Reconciliation

To configure user account status reconciliation, you must do the following:

- **For trusted sources:** You must create a reconciliation field, for example, *Status*, in the corresponding trusted resource object, which denotes the status of the user in the target. The value of this field must be either *Active* or *Disabled*. This reconciliation field must be mapped to the user attribute *status* in the corresponding process definition.
- **For target resources:** You must create a reconciliation field, for example, *Status*, in the corresponding resource object, which denotes the status of the resource in the target. This reconciliation field must be mapped to the process attribute *OIM_OBJECT_STATUS* in the corresponding process definition. The following statuses are supported for target resource reconciliation:
 - Revoked
 - Provisioned
 - Ready
 - Provide Information
 - Enabled

- None
- Waiting
- Provisioning
- Disabled

12.3.2.2.2 Mapping a Target Resource Field to Oracle Identity Manager

You can map the fields on a target resource or trusted source, as defined on the **Reconciliation Fields** tab of the associated resource definition, to applicable fields in Oracle Identity Manager. These mappings determine the fields that must be updated in Oracle Identity Manager in a reconciliation event. These mappings occur when you click one of the following on the Reconciliation Manager form:

- The **Create User** or **Create Organization** button
- The **Link** button on the **Matched Users** or **Matched Organizations** tab
- The **Establish Link** button on the **Processes Matched Tree** tab

For user discovery on a trusted source, you define the fields to be mapped from the **User** resource to fields in the User provisioning process. The fields (that is, the user attributes) to which you will map your trusted source fields are derived from the **Users** form.

For organization discovery on a trusted source, you define fields to be mapped from the Oracle Identity Manager Organization resource to fields in the Oracle Identity Manager Organization provisioning process. The fields (that is, the organization attributes) to which you will map your trusted source fields are derived from the **Organizations** form.

After you have accessed the provisioning process definition for the associated resource and selected the **Reconciliation Field Mappings** tab, use one of the two procedures described in the following sections.

Mapping a Single Value Field

To map a single value field:

1. Click **Add Field Map**.

The Add Reconciliation Field Mappings dialog box is displayed.

2. Select the field on the target system that you want to map from the menu in the **Field Name** field.

Oracle Identity Manager will automatically supply the field type based on what was entered for this field on the associated **Resource Object** form.

3. For trusted sources:

Select a value from the **User Attribute** menu and click **OK**. Go to Step 4.

For target resources:

Double-click **Process Data Field**. Select the correct mapping from the **Lookup** dialog box and click **OK**.

4. If you are defining mapping for a trusted source, go to step 5.

Set the **Key Field for Reconciliation Matching** check box for target resources only. If this check box is selected, Oracle Identity Manager evaluates if the value of this field on the provisioning process form matches the value of the field in the reconciliation event. All matched processes are displayed on the **Processes**

Matched Tree tab of the Reconciliation Manager form. If this check box is deselected, Oracle Identity Manager does not require the value of this field to match the process form and reconciliation event for process matching.

Note: To set a field as a key field, it must be set as required on the **Object Reconciliation** tab of the applicable resource.

5. Click **Save**.

The mapping for the selected fields is applied the next time a reconciliation event is received from the target resource or trusted source.

Mapping a Multi-Value Field (For Target Resources Only)

To map a multi-value field:

1. Click **Add Table Map**.

The Add Reconciliation Table Mappings dialog box is displayed.

2. Select the multi-value field on the target system that you want to map from the menu in the **Field Name** field.

Oracle Identity Manager will automatically supply the field type based on what was entered for this field on the associated Resource Object form.

3. Select the child table you defined on the target resource's process form from the **Table Name** menu.

4. Double-click **Process Data Field**, and select the correct mapping from the Lookup dialog box, and click **OK**.

5. Save and close the Add Reconciliation Table Mappings dialog box.

6. Right-click the multi-value field you just mapped, and select Define a property field map from the menu that is displayed.

7. Select the component (child) field you want to map.

Oracle Identity Manager will automatically supply the field type based on what was entered for this field on the associated Resource Object form.

8. Double-click the **Process Data Field** field.

Select the correct mapping from the Lookup dialog box and click **OK**.

9. Set the **Key Field for Reconciliation Matching** check box.

If this check box is selected, Oracle Identity Manager compares the field value on the provisioning process child form with the field value in the reconciliation event. All matching processes are displayed on the **Processes Matched Tree** tab of the Reconciliation Manager form. If you deselect this check box, the value of this field does not have to match on the process form and reconciliation event for process matching. Ensure that at least one component (child) field of each multi-valued field is set as a key field. This improves the quality of the matches generated on the **Process Matched Tree** tab.

Note: Key fields must be set as required on the **Object Reconciliation** tab of the applicable resource.

10. Repeat Steps 6 through 9 for each component (child) field defined on the multi-value field.
11. Click **Save**.

The mapping for the selected fields will be applied the next time a reconciliation event is received from the target resource.

12.3.2.2.3 Deleting a Mapping

This procedure is used to delete a mapping that has been established between a field in Oracle Identity Manager and a field on the target system or trusted source as defined on the **Reconciliation Fields** tab of the associated resource definition.

To delete a mapping:

1. Go to the provisioning process definition for the associated resource.
2. Select the **Reconciliation Field Mappings** tab.
3. Select the field mapping you want to delete.
4. Click **Delete Map**.

The mapping for the selected field is deleted.

12.3.2.3 Administrators Tab

You use this tab to select the roles that can view, modify, and delete the current process definition.

On this tab, when the **Write** check box is selected, the corresponding role can read and modify the current process definition. When the **Delete** check box is selected, the associated role can delete the current process definition.

12.3.2.3.1 Assigning a Role to a Process Definition

To assign a role:

1. Click **Assign**.
The Groups window is displayed.
2. Select the unassigned role, and assign it to the process definition.
3. Click **OK**.
The role is displayed in the **Administrators** tab.
4. To enable this role to view or modify, or view and modify the current process definition, double-click the corresponding **Write** check box. Otherwise, go to Step 5.
5. To enable this role to delete the current process definition, double-click the associated **Delete** check box. Otherwise, go to Step 6.
6. Click **Save**.
The role is assigned to the process definition.

12.3.2.3.2 Removing a Role From a Process Definition

To remove a role:

1. Highlight the role that you want to remove.
2. Click **Delete**.

The role is removed from the process definition.

12.3.3 Modifying Process Tasks

To modify a process task for a process definition, double-click its row heading. The Editing Task window is displayed, containing additional information about the process task.

The Editing Task window contains the following tabs:

- [General Tab](#)
- [Integration Tab](#)
- [Task Dependency Tab](#)
- [Responses Tab](#)
- [Undo/Recovery Tab](#)
- [Notification Tab](#)
- [Task to Object Status Mapping Tab](#)
- [Assignment Tab of the Editing Task Window](#)

12.3.3.1 General Tab

You use this tab to set high-level information for the task that you want to modify. For this example, the **Create User** task is used to create a user in the Solaris environment.

[Table 12-3](#) describes the fields of the General tab.

Table 12-3 *Fields of the General Tab of the Editing Task Dialog Box*

Field Name	Description
Task Name	The name of the process task.
Task Description	Explanatory information about the process task.
Duration	The expected completion time of the current process task in days, hours, and minutes.
Conditional	<p>This check box determines if a condition is met to add the current process task to the process.</p> <p>Select this check box to prevent the process task from being added to the process unless a condition has been met.</p> <p>Clear this check box to not require the condition to be met for the process task to be added to the process.</p>
Required for Completion	<p>This check box determines if the current process task must be completed for the process to be completed.</p> <p>Select this check box to require the process task to have a status of Completed before the process can be completed.</p> <p>Deselect this check box to ensure that the status of the process task does not affect the completion status of the process.</p>
Constant Duration	Not applicable
Task Effect	<p>From this box, select the process action you want to associate with the task, for example, disable or enable. A process can enable or disable a user's access to a resource. When the disable action is chosen, all tasks associated with the disable action are inserted.</p> <p>Note: If you do not want the process task to be associated with a particular process action, select NONE from the box.</p>

Table 12–3 (Cont.) Fields of the General Tab of the Editing Task Dialog Box

Field Name	Description
Disable Manual Insert	<p>This check box determines if a user can manually add the current process task to the process.</p> <p>Select this check box to prevent the process task from being added to the process manually.</p> <p>Deselect this check box to enable a user to add the process task to the process.</p>
Allow Cancellation while Pending	<p>This check box determines if the process task can be canceled if its status is Pending.</p> <p>Select this check box to allow the process task to be canceled if it has a Pending status.</p> <p>Deselecting this check box to prevent the process task from being canceled if its status is Pending.</p>
Allow Multiple Instances	<p>This check box determines if the process task can be inserted into the current process more than once.</p> <p>Select this check box to enable multiple instances of the process task to be added to the process.</p> <p>Deselect this check box to enable the process task to be added to the current process only once.</p>
Retry Period in Minutes	<p>If a process task is rejected, this field determines the interval before Oracle Identity Manager inserts a new instance of that task with the status of Pending.</p> <p>When the value of the Retry Period in Minutes field is 30, it means that if the Create User process task is rejected, then in 30 minutes Oracle Identity Manager adds a new instance of this task and assigns it a status of Pending.</p>
Retry Count	<p>Determines how many times Oracle Identity Manager retries a rejected task. When the value of the Retry Count field is 5, it means that if the Create User process task is rejected, then Oracle Identity Manager adds a new instance of this task, and assigns it a status of Pending. When this process task is rejected for the fifth time, Oracle Identity Manager no longer inserts a new instance of it.</p>
Child Table/ Trigger Type	<p>These boxes specify the action that Oracle Identity Manager performs in the child table of a custom form that is associated with the current process, as indicated by the Table Name field of the Process Definition form.</p> <p>From the Child Table box, select the child table of the custom form where Oracle Identity Manager will perform an action.</p> <p>From the Trigger Type box, specify the action that Oracle Identity Manager is to perform in the child table. These actions include:</p> <ul style="list-style-type: none"> ■ Insert. Adds a new value to the designated column of the child table ■ Update. Modifies an existing value from the corresponding column of the child table ■ Delete. Removes a value from the designated column of the child table <p>Note: If the custom process form does not have any child tables associated with it, the Child Table box will be empty. In addition, the Trigger Type box will be grayed out.</p>
Off-line	<p>This flag is applicable only for user attribute propagation tasks. If the flag is set for a user attribute propagation task, the task insertion is asynchronous.</p>

12.3.3.1.1 Modifying a Process Task's General Information

To modify the general information for a process task:

1. Double-click the row heading of the task you want to modify.
The Editing Task dialog box is displayed.
2. Click the **General** tab.
3. In the **Description** field, enter explanatory information about the process task.
4. Optional. In the **Duration** area, enter the expected completion time of the process task (in days, hours, and minutes).
5. If you want a condition to be met for the process task to be added to the Process Instance, select the **Conditional** check box. Otherwise, go to Step 6.

Note: If you select the **Conditional** check box, you must specify the condition to be met for the task to be added to the process.

6. When you want the completion status of the process to depend on the completion status of the process task, select the **Required for Completion** check box.
By doing so, the process cannot be completed if the process task does not have a status of Completed.
If you do not want the status of the process task to affect the completion status of the process, go to Step 7.
7. To prevent a user from manually adding the process task into a currently running instance of the process, select the **Disable Manual Insert** check box. Otherwise, go to Step 8.
8. To enable a user to cancel the process task if its status is Pending, select the **Allow Cancellation while Pending** check box. Otherwise, go to Step 9.
9. To allow this task to be inserted multiple times in a single process instance, select the **Allow Multiple Instances** check box. Otherwise, go to Step 10.
10. Click the **Task Effect** box.

From the custom menu that is displayed, select one of the following:

- **Enable Process or Access to Application.** If a resource is reactivated by using the enable function, all tasks with this effect are inserted into the process. If you select this option, you must also select the **Allow Multiple Instances** check box.
 - **Disable Process or Access to Application.** If a resource is deactivated by using the disable function, all tasks with this effect are inserted into the process. If you select this option, you must also select the **Allow Multiple Instances** check box.
 - **No Effect.** This is the default process action associated with all tasks. If this option is selected, the task is only inserted during normal provisioning unless it is conditional.
11. Optional. If the process task is **Rejected**, you might want Oracle Identity Manager to insert a new instance of this process task (with a status of **Pending**).

For this to occur, enter a value in the **Retry Period in Minutes** field. This designates the time in minutes that Oracle Identity Manager waits before adding this process task instance.

In the **Retry Count** field, enter the number of times Oracle Identity Manager will retry a rejected task. For example, suppose **3** is displayed in the **Retry Count** field. If the task is rejected, Oracle Identity Manager adds a new instance of this task, and assigns it a status of Pending. After this process task is rejected for the fourth time, Oracle Identity Manager no longer inserts a new instance of the process task.

Note: If either **Retry Period** or **Retry Count** is selected, you must specify parameters for the other option because they are both related.

12. From the **Child Table** box, select the child table of the custom form where Oracle Identity Manager will perform an action.

From the **Trigger Type** box, specify the action that Oracle Identity Manager will perform in the child table. These actions include the following:

- **Insert:** Adds a new value to the designated column of the child table
- **Update:** Modifies an existing value from the corresponding column of the child table
- **Delete:** Removes a value from the designated column of the child table

Note: If the custom process form does not have any child tables associated with it, the **Child Table** box will be empty. In addition, the **Trigger Type** box will be grayed out.

13. Click **Save**.

The modifications to the process task's top-level information reflects the changes you made in the **General** tab.

12.3.3.1.2 Triggering Process Tasks for Events Defined in Lookup.USR_PROCESS_TRIGGERS Fields

When a user attribute is defined in Lookup.USR_PROCESS_TRIGGERS, for each modification of the attribute, the corresponding process task is triggered for each provisioned resource. This is same for the First Name, Last Name, Display Name (USR_DISPLAY_NAME) user attributes and custom user attributes. However, for the Lookup.USR_PROCESS_TRIGGERS fields USR_STATUS, USR_LOCKED, USR_LOCKED_ON, and USR_MANUALLY_LOCKED, the attached process task is not triggered.

The following sections describe how to trigger the process tasks for the Lookup.USR_PROCESS_TRIGGERS fields:

For the USR_STATUS Attribute

It is not possible to run a task via Lookup.USR_PROCESS_TRIGGERS for the USR_STATUS attribute because this attribute is processed separately by Oracle Identity Manager. This attribute is changed by enabling, disabling, or deleting a user. These operations have a special effect on the provisioned resources because the corresponding process tasks are started via the Task Effect setting, as described in [Table 12–3, "Fields of the General Tab of the Editing Task Dialog Box"](#). For these three

operations, the Lookup.USR_PROCESS_TRIGGERS is not used. Therefore, when the status changes, perform the following to run the process task:

For transition from Disabled to Enabled status:

1. In the Process Definition form, create a process task named `Enable User`.
2. Open the Editing Task window, and click the **General** tab.
3. From the Task Effect list, select **Enables Process or Access to Application**.
4. Select **Conditional** and specify the condition to be met for the task to be added to the process.

For transition from Enabled to Disabled status:

1. In the Process Definition form, create a process task named `Disable User`.
2. Open the Editing Task window, and click the **General** tab.
3. From the Task Effect list, select **Enables Process or Access to Application**.
4. Select **Conditional** and specify the condition to be met for the task to be added to the process.

For transition from Enabled/Disabled/Provisioned to Revoked status:

1. In the Process Definition form, create a process task named `Delete User`.
2. Then set this task as an Undo task for the Create User task, which is the task that creates the user and is typically unconditional.
3. Select **Conditional** and specify the condition to be met for the task to be added to the process.

Note: when the OIM user is deleted, for each completed task in each resource, Oracle Identity Manager tries to run the Undo tasks.

For the USR_LOCKED, USR_LOCKED_ON, USR_MANUALLY_LOCKED Attributes

The lock and unlock operations, are handled in Oracle Identity Manager as separate orchestrations. The orchestration is on:

```
entity-type="User" operation="LOCK"
```

Or:

```
entity-type="User" operation="UNLOCK"
```

The event handler that does the evaluation for Lookup.USR_PROCESS_TRIGGERS is:

```
oracle.iam.transUI.impl.handlers.TriggerUserProcesses
```

This is triggered only in the following user orchestrations:

- **MODIFY:** For generic fields
- **CHANGE_PASSWORD, RESET_PASSWORD:** For USR_PASSWORD propagation
- **ENABLE, DISABLE, DELETE:** For handling the execution of process tasks

For lock/unlock operations, the TriggerUserProcesses event handler is not triggered. Therefore, for the attributes modified through lock/unlock operations, the Lookup.USR_PROCESS_TRIGGERS is not checked.

If you want to run custom code for these operations when these fields are changed, then you can create event handlers and register them on the orchestrations mentioned in this section.

12.3.3.2 Integration Tab

By using the **Integration** tab, you can:

- Automate a process task by attaching an event handler or task adapter to it.
- Map the variables of the task adapter, so Oracle Identity Manager can pass the appropriate information when the adapter is triggered. This occurs when the process task's status is Pending.
- Break the link between the adapter handler and the process task, once the adapter or event handler is no longer applicable with the process task.

For example, suppose that the adpSOLARISCREATEUSER adapter is attached to the Create User process task. This adapter has nine adapter variables, all of which are mapped correctly as indicated by the Y that precedes each variable name.

Note: Event handlers are preceded with tc (Thor class), such as tcCheckAppInstalled. These are event handlers that Oracle provides. Customer-created event handlers cannot have a tc prefix in their name. Adapters are preceded with adp, for example, adpSOLARISCREATEUSER.

See Also: [Chapter 2, "Developing Adapters"](#) and ["Event Handler Manager Form"](#) on page 5-1 for more information about adapters and event handlers

12.3.3.2.1 Assigning an Adapter or Event Handler to a Process Task

The following procedure describes how to assign an adapter or event handler to a process task.

Important: If you assign an adapter to the process task, the adapter will not work until you map the adapter variables correctly. See ["Mapping Adapter Variables"](#) on page 12-21 for details.

To assign an adapter or event handler to a process task:

1. Double-click the row heading of the process task to which you want to assign an event handler or adapter.

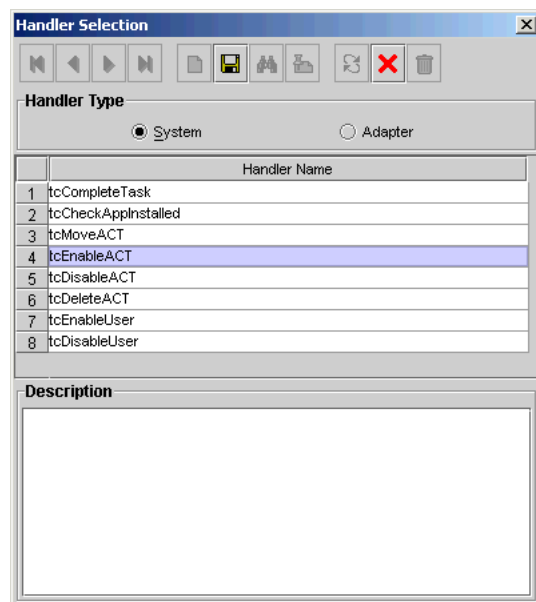
The Editing Task window is displayed.

2. Click the **Integration** tab.
3. Click **Add**.

The **Handler Selection** dialog box is displayed, as shown in [Figure 12-5](#).

4. To assign an event handler to the process task, select the **System** option.

To add an adapter to the process task, select the **Adapter** option. A list of event handlers or adapters, which you can assign to the process task, is displayed in the **Handler Name** region.

Figure 12–5 Handler Selection Dialog Box

5. Select the event handler or adapter that you want to assign to the process task.
6. From the Handler Selection window's Toolbar, click **Save**.
A confirmation dialog box is displayed.
7. Click **OK**.
The event handler or adapter is assigned to the process task.

12.3.3.2.2 Mapping Adapter Variables

See Also: "[Adapter Mapping Information](#)" on page 3-18 for more information about the items to select in this procedure

Note: To trigger a task associated with a change to a parent form field, the name of the task must be *fieldUpdated*, where *field* is the name of the parent form field. If the task is not named according to this convention, it is not triggered during a field update.

To map an adapter variable:

1. Select the adapter variable that you want to map.
2. Click **Map**.
The Data Mapping for Variable window is displayed.
3. Complete the **Map To**, **Qualifier**, **IT Asset Type**, **IT Asset Property**, **Literal Value**, and **Old Value** fields.
4. From the Data Mapping for Variable window's Toolbar, click **Save**.
5. Click **Close**.

The mapping status for the adapter variable changes from N to Y. This indicates that the adapter variable has been mapped.

12.3.3.2.3 Removing an Adapter or Event Handler from a Process Task

To remove an adapter or event handler from a process task:

1. Click **Remove**.

A confirmation dialog box is displayed.

2. Click **OK**.

The event handler or adapter is removed from the process task.

12.3.3.3 Task Dependency Tab

You use the **Task Dependency** tab to determine the logical flow of process tasks in a process. Through this tab, you can:

- Assign **preceding** tasks to a process task.
These tasks must have a status of **Completed** before Oracle Identity Manager or a user can trigger the current process task.
- Assign **dependent** tasks to a process task.
Oracle Identity Manager or a user can trigger these tasks only after the current process task has a status of **Completed**.
- Break the link between a preceding task and the current task so that the preceding task's completion status no longer has any effect on the current task being triggered.
- Break the link between the current task and a dependent task so that the current task's completion status no longer has any bearing on triggering the dependent tasks.

For example, the **Create User** process task does not have any preceding tasks. Oracle Identity Manager triggers this task whenever the task is inserted into a process (for example, when an associated resource is requested). The **Create User** process task has seven dependent tasks. Before completion of this process task, each dependent task will have a status of **Waiting**. Once this task achieves a status of **Completed**, each of these process tasks are assigned a status of **Pending**, and Oracle Identity Manager can trigger them.

12.3.3.3.1 Assigning a Preceding Task to a Process Task

To assign a preceding task to a process task:

1. Double-click the row heading of the process task to which you want to assign a preceding task.

The **Editing Task** window is displayed.

2. Click the **Task Dependency** tab.

3. From the Preceding Tasks region, click **Assign**.

The **Assignment** window is displayed.

4. From this window, select the preceding task, and assign it to the process task.

5. Click **OK**.

The preceding task is assigned to the process task.

12.3.3.3.2 Removing a Preceding Task from a Process Task

To remove a preceding task from a process task:

1. Select the preceding task that you want to delete.
2. From the Preceding Tasks region, click **Delete**.

The preceding task is removed from the process task.

12.3.3.3.3 Assigning a Dependent Task to a Process Task

To assign a dependent task to a process task:

1. Double-click the row heading of the process task to which you want to assign a dependent task.

The Editing Task window is displayed.

2. Click the **Task Dependency** tab.
3. From the **Dependent Tasks** region, click **Assign**.

The Assignment window is displayed.

4. From this window, select the dependent task, and assign it to the process task.
5. Click **OK**.

The dependent task is assigned to the process task.

12.3.3.3.4 Removing a Dependent Task from a Process Task

To remove a dependent task from a process task:

1. Select the dependent task that you want to delete.
2. From the **Dependent Tasks** region, click **Delete**.

The dependent task is removed from the process task.

12.3.3.4 Responses Tab

You use the Responses tab to do the following:

- Define the response codes that can be received in conjunction with the execution of a particular process tasks. You can use response codes to represent specific conditions on the target system.
- Define the conditional tasks that are started if a response code is received during execution of this process task. These tasks are called generated tasks.
- Remove a response from a process task.
- Remove a generated task from a process task.

For example, when a Create User process task is completed, the `SUCCESS` response is activated. This response displays a dialog box with the message "The user was created successfully." In addition, Oracle Identity Manager triggers the Enable User process task.

Note: By default, the `UNKNOWN` response is defined for each process task that is rejected. This way, even when the system administrator does not add any responses to a process task, if this task is rejected, the user will be notified in the form of an error message in a dialog box.

12.3.3.4.1 Adding a Response to a Process Task

To add a response to a process task:

1. Double-click the row heading of the process task to which you want to add a response.

The Editing Task window is displayed.

2. Click the **Responses** tab.
3. In the **Responses** region, click **Add**.

A blank row is displayed in the Responses region.

4. Enter information in the **Response** field.

This field contains the response code value. This field is case-sensitive.

5. Enter information in the **Description** field. This field contains explanatory information about the response.

If the process task triggers the response, this information is displayed in the task information dialog box.

6. Double-click the **Status** lookup field.

From the Lookup window that is displayed, select a task status level. If the response code is received, it will cause the task to be set to this status.

7. Click **Save**.

The response you added would now reflect the settings you have entered.

12.3.3.4.2 Removing a Response from a Process Task

To remove a response from a process task:

1. Select the response that you want to delete.
2. From the **Responses** region, click **Delete**.

The response is removed from the process task.

Note: You will not be able to delete a response from a process task that is invoked for any provisioning instance, even if the response is existing or is newly added. However, if the process task is not invoked for any provisioning instance, you will be able to delete the response.

12.3.3.4.3 Assigning a Generated Task to a Process Task

To assign a generated task to a process task:

1. Double-click the row heading of the process task to which you want to assign a generated task.

The Editing Task window is displayed.

2. Click the **Responses** tab.
3. Select the response code for which you want to assign generated tasks.
4. From the **Tasks to Generate** region, click **Assign**.

The Assignment window is displayed.

5. From this window, select the generated task, and assign it to the process task response.
6. Click **OK**.

The generated task is assigned to the process task.

12.3.3.4.4 Removing a Generated Task From a Process Task

To remove a generated task from a process task:

1. Select a response code.
2. Select the generated task that you want to delete.
3. From the Tasks to Generate region, click **Delete**.

The generated task is removed from the process task.

12.3.3.5 Undo/Recovery Tab

You use the Undo/Recovery tab for the following:

- To define process tasks that are triggered when the current process task is canceled. These process tasks are known as undo tasks.
- To remove an undo task from a process task, when it is no longer valid.
- To define process tasks that are triggered when the current process task is rejected. These tasks are called recovery tasks.
- To remove a recovery task from a process task.

For example, if the Create User process task has a `Canceled` status, the Delete User undo task is triggered. Similarly, if the Create User task is `Rejected`, Oracle Identity Manager triggers the Enable User recovery task.

Note: When the current process task is rejected, Oracle Identity Manager triggers recovery tasks that are assigned to the process task. If you select the Complete on Recovery check box, Oracle Identity Manager changes the status of the current process task from `Rejected` to `Unsuccessfully Completed` upon completion of all recovery tasks that are generated. This enables Oracle Identity Manager to trigger other dependent process tasks.

The following sections describe how to assign an undo and recovery task to the current process task, and how to remove an undo and recovery task from the current process task.

12.3.3.5.1 Assigning an Undo Task to a Process Task

To assign an undo task to a process task:

1. Double-click the row heading of the process task to which you want to assign an undo task.

The Editing Task window is displayed.

2. Click the **Undo/Recovery** tab.
3. In the **Undo Tasks** region, click **Assign**.

The Assignment window is displayed.

4. From this window, select the undo task, and assign it to the process task.
5. Click **OK**.

The undo task is assigned to the process task.

12.3.3.5.2 Removing an Undo Task From a Process Task

To remove an undo task from a process task:

1. Select the undo task that you want to delete.
2. From the **Undo Tasks** region, click **Delete**.

The undo task is removed from the process task.

12.3.3.5.3 Assigning a Recovery Task to a Process Task

To assign a recovery task to a process task:

1. Double-click the row heading of the process task to which you want to assign a recovery task.

The Editing Task window is displayed.

2. Click the **Undo/Recovery** tab.
3. From the **Recovery Tasks** region, click **Assign**.

The Assignment window is displayed.

4. From this window, select the recovery task, and assign it to the process task.
5. Click **OK**.

The recovery task is assigned to the process task.

6. Optional. If you want the status of the current process task to change from Rejected to Unsuccessfully Completed upon completion of all recovery tasks that are generated (so Oracle Identity Manager can trigger other, dependent process tasks) select the Complete on Recovery check box. Otherwise, do not select this check box.

12.3.3.5.4 Removing a Recovery Task from a Process Task

To remove an recovery task from a process task:

1. Select the recovery task that you want to delete.
2. From the **Recovery Tasks** region, click **Delete**.

The recovery task is removed from the process task.

12.3.3.6 Notification Tab

You use this tab to designate the e-mail notification to be generated when the current process task achieves a particular status. A separate e-mail notification can be generated for each status a task can achieve. If an e-mail notification is no longer valid, you can remove it from the Notification tab.

For example, when the Create User process task achieves a status of **Completed**, Oracle Identity Manager sends the Process Task Completed e-mail notification to the user who is to be provisioned with the resource. If the Create User process task is rejected, the Process Task Completed e-mail notification is sent to the user and the user's manager.

Note: Oracle Identity Manager can only send an e-mail notification to a user if you first create a template for the e-mail message by using the Email Definition form.

See "[Email Definition Form](#)" on page 12-1 for details.

The following sections describe how to assign e-mail notifications to a process task, and remove e-mail notifications from a process task.

12.3.3.6.1 Assigning an E-Mail Notification to a Process Task

To assign an e-mail notification to a process task:

1. Double-click the row heading of the process task to which you want to assign an e-mail notification.

The Editing Task dialog box is displayed.

2. Click the **Notification** tab.

3. Click **Assign**.

The Assignment dialog box is displayed.

4. From this window, select the e-mail template definition to use, and assign it to the process task.

5. Click **OK**.

The name of the e-mail notification is displayed in the Notification tab.

6. Double-click the **Status** lookup field.

From the Lookup window that is displayed, select a completion status level. When the process task achieves this status level, Oracle Identity Manager will send the associated e-mail notification.

7. Select the check boxes that represent the users who will receive the e-mail notification.

Currently, an e-mail notification can be sent to the following users:

- **Assignee.** This user is responsible for completing the associated process task.
- **Requester.** This user requested the process that contains the corresponding process task.
- **User.** This user will be provisioned with the resource once the associated process task is Completed.
- **User's Manager.** This user is the supervisor of the user, who will be provisioned with the resource once the corresponding process task is Completed.

8. Click **Save**.

The e-mail notification is assigned to the process task.

12.3.3.6.2 Removing an E-mail Notification from a Process Task

The following procedure describes how to remove an e-mail notification from a process task.

To remove an e-mail notification from a process task:

1. Select the e-mail notification that you want to delete.
2. Click **Delete**.

The e-mail notification is removed from the process task.

12.3.3.7 Task to Object Status Mapping Tab

A resource object contains data that is used to provision resources to users and applications.

In addition, a resource object is provided with predefined provisioning statuses, which represent the various statuses of the resource object throughout its life cycle as it is being provisioned to the target user or organization.

Note: Provisioning statuses are defined in the **Status Definition** tab of the **Resource Objects** form.

The provisioning status of a resource object is determined by the status of its associated provisioning processes, and the tasks that comprise these processes. For this reason, you must provide a link between the status of a process task and the provisioning status of the resource object to which it is assigned.

The **Task to Object Status Mapping** tab is used to create this link. Also, when this connection is no longer required, or you want to associate a process task status with a different provisioning status for the resource object, you must break the link that currently exists.

For this example, there are five mappings among process task statuses and provisioning statuses of a resource object. When the Create User process task achieves a status of **Completed**, the associated resource object will be assigned a provisioning status of **Provisioned**. However, if this task is canceled, the provisioning status for the resource object will be **Revoked**. **None** indicates that this status has no effect on the provisioning status of the resource object.

The following sections describe how to map a process task status to a provisioning status and unmap a process task status from a provisioning status.

12.3.3.7.1 Mapping a Process Task Status to a Provisioning Status

To map an process task status to a provisioning status:

1. Double-click the row heading of the process task, which has a status that you want to map to the provisioning status of a resource object.

The Editing Task window is displayed.

2. Click the **Task to Object Status Mapping** tab.
3. Select the desired process task status.
4. Double-click the **Object Status** lookup field.

From the Lookup window that is displayed, select the provisioning status of the resource object to which you want to map the process task status.

5. Click **OK**.

The provisioning status you selected is displayed in the Task to Object Status Mapping tab.

6. Click **Save**.

The process task status is mapped to the provisioning status.

12.3.3.7.2 Unmapping a Process Task Status From a Provisioning Status

To unmap an process task status from a provisioning status:

1. Select the desired process task status.
2. Double-click the **Object Status** lookup field.

From the Lookup window that is displayed, select None. None indicates that this status has no effect on the provisioning status of the resource object.

3. Click **OK**.

The provisioning status of None is displayed in the **Task to Object Status Mapping** tab.

4. Click **Save**.

The process task status is no longer mapped to the provisioning status of the resource object.

12.3.3.8 Assignment Tab of the Editing Task Window

This tab is used to specify assignment rules for the current process task. These rules will determine how the process task will be assigned.

Note: Task assignment rules are useful when associated with tasks that are to be completed manually. Most provisioning process tasks are automated, and as a result, they might not require task assignment rules.

If the criteria of the Solaris Process Tasks - User rule are not satisfied, Oracle Identity Manager evaluates the criteria of the Solaris Process Tasks - Group rule. If that rule's criteria are met, the task is assigned to the SYSTEM ADMINISTRATORS role, and the task is marked to escalate in 10 minutes.

Note: Only rules with a classification type of Task Assignment can be assigned to a process task. For more information about specifying the classification type of a rule, see "[Rule Designer Form](#)" on page 11-51. In addition, a Default rule is predefined in Oracle Identity Manager. This rule always evaluates to True. Therefore, it can be used as a safeguard mechanism to ensure that at least one predefined task assignment occurs if all the other rules fail.

[Table 12-4](#) describes the fields of the Assignment tab.

Table 12-4 *Fields of the Assignment Tab of the Editing Task Window*

Field Name	Description
Rule	The name of the Task Assignment rule to evaluate.

Table 12–4 (Cont.) Fields of the Assignment Tab of the Editing Task Window

Field Name	Description
Target Type	<p>The classification type of the user or role that is responsible for completing the current process task. Currently, the process task can be assigned to:</p> <ul style="list-style-type: none"> ■ User. An Oracle Identity Manager user. ■ Role. A role. ■ Group User with Least Load. The member of the specified role with the fewest process tasks assigned. ■ Request Target User's Manager. The supervisor of the user who is being provisioned with the resource. ■ Object Authorizer User with Least Load. The member of the role (designated as an Object Authorizer role for the resource) with the fewest process tasks assigned. ■ Object Administrator. A role that is defined as an administrator of the associated resource object. ■ Object Administrator User with Least Load. The member of the role (designated as an Object Administrator role) with the fewest process tasks assigned. <p>Note: Object Authorizer and Object Administrator roles are defined in the Object Authorizers and Administrators tabs, respectively, of the Resource Objects form.</p>
Adapter	This is the name of the adapter. Double-click this field to get a lookup form for all existing adapters.
Adapter Status	This is the status of the adapter.
Group	The role to which the current process task is assigned.
User	The user to which the current process task is assigned.
Email Name & Send Email	By selecting an e-mail notification from the Email Name lookup field, and selecting the Send Email check box, Oracle Identity Manager will send the e-mail notification to a user or role once the current process task is assigned.
Escalation Time	The amount of time (in milliseconds) that the user or role, which is associated with the rule that Oracle Identity Manager triggers, has to complete the process task. If this process task is not completed in the allotted time, Oracle Identity Manager will re-assign it to another user or role. The escalation rule adheres to the order defined by the target type parameter.
Priority	The priority number of the rule that is associated with the current process task. This number indicates the order in which Oracle Identity Manager will evaluate the rule.

The following sections describe adding a task assignment rule to a process task and how to remove it from the process task.

12.3.3.8.1 Adding a Rule to a Process Task

To add a rule to a process task:

1. Double-click the row heading of the task to which you want to add a rule.
The Editing Task window is displayed.
2. Click the **Assignment** tab.
3. Click **Add**.

A blank row is displayed in the Assignment tab.

4. Double-click the **Rule** lookup field.

From the Lookup window that is displayed, select the rule that you want to add to the process task. Then, click **OK**.

5. Double-click the **Target Type** lookup field.

From the Lookup window that is displayed, select the classification type of the user or role (User, Role, Group User with Least Load, Request Target User's Manager, Object Authorizer User with Least Load, Object Administrator, Object Administrator User with Least Load) that is responsible for completing the process task. Then, click **OK**.

6. Double-click the **Group** lookup field.

From the Lookup window that is displayed, select the role that is responsible for completing the process task. This setting is only necessary if you selected **Group** or **Group User with Least Load** in the **Target Type** field. Then, click **OK**.

OR

Double-click the **User** lookup field. From the Lookup window that is displayed, select the user who is responsible for completing the process task. This setting is only necessary if you selected User in the **Target Type** field. Then, click **OK**.

7. Double-click the **Email Name** field.

From the Lookup window that is displayed, select the e-mail notification that will be sent to the corresponding user or role once the task is assigned. Click **OK**. Then, select the **Send Email** check box.

If you do not want Oracle Identity Manager to send an e-mail notification when the task is assigned, go to Step 8.

8. In the **Escalation Time** field, enter the time (in milliseconds) that the selected user or role has to complete the process task.

When you do not want to associate a time limit with the rule you are adding to the process task, leave the **Escalation Time** field empty, and proceed to Step 10.

9. In the **Priority** field, enter the priority number of the rule that you are adding to the process task.

10. Click **Save**.

The rule is added to the process task.

12.3.3.8.2 Removing a Rule from a Process Task

To remove a rule from a process task:

1. Select the rule that you want to delete.
2. Click **Delete**.

The rule is removed from the process task.

Developing Process Forms

This chapter describes how to develop provisioning process forms by using the Design Console. It contains the following sections:

- [Form Designer Form](#)
- [Error Message Definition Form](#)

13.1 Form Designer Form

The information required to provision resources to a target user or organization cannot always be retrieved from an existing Oracle Identity Manager form. You can use the Form Designer form in the Development Tools folder to create a form with fields that contain the relevant information. After creating the form, you assign it to the process or resource object that is associated with provisioning resources to the user or organization. [Figure 13-1](#) shows the Form Designer Form.

Oracle Identity Manager displays a resource object or process form that a user creates by using the Form Designer form for the following reason:

When the process form is attached to the appropriate provisioning process, and the Launch Form menu command is selected by right-clicking the process from the **Object Process Console** tab of the Organizations or Users forms.

For example, when Oracle Identity Manager or one of its users attempts to complete the resource object or process, the assigned form is triggered. When this occurs, either Oracle Identity Manager or a user populates the fields of this form. After the data is saved, the corresponding process or resource object can achieve a status of Completed, and Oracle Identity Manager can provision the appropriate resources to the target organizations or users.

Figure 13–1 Form Designer Form

For example, the **Solaris** form (represented by the **UD_SOLARIS** name in the Table Name field) has been created and assigned to both the Solaris resource object and provisioning process.

Note: The table name contains a **UD_** prefix, followed by the form name. For this example, because the name of the form is **SOLARIS**, its table name is **UD_SOLARIS**.

Table 13–1 describes the data fields of the Form Designer form.

Table 13–1 Fields of the Form Designer Form

Field Name	Description
Table Name	The name of the database table that is associated with the form. Note: The table name contains the UD_ prefix, followed by the form name. If the name of the form is SOLARIS , its table name is UD_SOLARIS .
Description	Explanatory information about the form. Important: The text that is displayed in the Description field is the name of the form.
Preview Form	When you click this button, the form is displayed. This way, you can see how it looks and functions before you make it active.

Table 13–1 (Cont.) Fields of the Form Designer Form

Field Name	Description
Form Type	<p>These options are used to designate if the form is to be assigned to a process or a resource object.</p> <p>If you select the Process option, the form is associated with an approval or provisioning process.</p>
Object Name	<p>This is the name of the resource that can be provisioned (for example, a database, server, software application, file, or directory access). Also, referred to as a <i>resource object name</i>.</p> <p>Double-click this field to see the available resource object names.</p>
Latest Version	The most recent version of the form.
Active Version	<p>The version of the form that is used with the designated process or resource object.</p> <p>Note: After a version of the form is displayed in the Active Version field, it cannot be modified.</p>
Current Version	This version of the form is being viewed and contains information, which is displayed throughout the various tabs of the Form Designer form.
Create New Version	<p>If you click this button, you can assign an additional name to the existing version of a form. As a result, you can modify this version, without effecting the original version of the form.</p> <p>Note: If you create a new version of the form and click Refresh, the name that you provided for this version is displayed in the Current Version box.</p>
Make Version Active	<p>By clicking this button, you can specify that the current version of the form is the one that is to be assigned to the process or resource object. In other words, this version is now active.</p> <p>Note: After a version of the form is active, it cannot be modified. Instead, you must create another additional version of the form (by clicking the Create New Version button).</p>

The following sections describes how to work with forms:

- [Creating a Form](#)
- [Tabs of the Form Designer Form](#)
- [Creating an Additional Version of a Form](#)

13.1.1 Creating a Form

To create a form:

1. Open the Form Designer form.
2. In the **Table Name** field, enter the name of the database table that is associated with the form.

Note: The table name contains the **UD_** prefix followed by the form name. If the name of the form is **SOLARIS**, its table name is **UD_SOLARIS**.

3. In the **Description** field, enter explanatory information about the form.

4. Select the **Process** option. This is because the form is assigned to a provisioning process.
5. Click **Save**.

The form is created. The words **Initial Version** are displayed in the **Latest Version** field. This signifies that you can populate the tabs of the Form Designer form with information, so the form is functional with its assigned process or resource.

13.1.2 Tabs of the Form Designer Form

After you open the Form Designer form, and create a form, the tabs of this form become functional. The Form Designer form contains the following tabs:

- [Additional Columns Tab](#)
- [Child Table\(s\) Tab](#)
- [Object Permissions Tab](#)
- [Properties Tab](#)
- [Administrators Tab](#)
- [Usage Tab](#)
- [Pre-Populate Tab](#)
- [Default Columns Tab](#)
- [User Defined Fields Tab](#)

13.1.2.1 Additional Columns Tab

You use the **Additional Columns** tab to create and manage data fields. These data fields are displayed on the associated form that is created by using the Form Designer form.

[Table 13–2](#) describes the data fields of the Additional Columns tab.

Table 13–2 Fields of the Additional Columns Tab

Name	Description
Name	<p>The name of the data field that is displayed in the database and is recognized by Oracle Identity Manager.</p> <p>Note: This name consists of the <TABLENAME_> prefix followed by the name of the data field.</p> <p>For example, if the name in the Table Name field of the Form Designer form is UD_PASSWORD and the name for the data field is USERNAME, the data field name that is displayed in the database and that Oracle Identity Manager recognizes, would be UD_PASSWORD_USERNAME.</p>
Variant Type	<p>From this lookup field, select the variant type for the data field. The variant type denotes the type of data that the field accepts.</p> <p>This data field must be one of nine variant types: Byte, Double, Date, Byte Array, Boolean, Long, String, Short, and Integer.</p>
Length	The length in characters of the data field.
Field Label	The label that is associated with the data field. This label is displayed next to the data field on the form that is generated by Oracle Identity Manager.

Table 13-2 (Cont.) Fields of the Additional Columns Tab

Name	Description
Field Type	<p>From this lookup field, select the data type of the data field. The data type represents how the data must be displayed in the field.</p> <p>You can select one of the following data types:</p> <ul style="list-style-type: none"> ■ TextField: This data field is displayed on the generated form as a text field. If the text field is display-only (the text in the field is displayed in red font), a user can use the field only to run a query. Otherwise, the user can also populate the field with information, and save it to the database. ■ LookupField: This data field is displayed on the generated form as a lookup field. If this lookup field is display-only, a user can use the field only to run a query. Otherwise, the user can also populate the field with a value from the associated Lookup window, and save this value to the database. ■ TextArea: This data field is displayed on the generated form as a text area. If this text area is display-only, a user can only read the information that is displayed in it. Otherwise, the user can also populate the text area with data, and save this information to the database. ■ ITResourceLookupField: This data field is displayed on the generated form as a lookup field. From this lookup field, a user can select a lookup value that represents an IT resource, and save this value to the database. Note: If you select this data field, you must specify the type of server for the IT resource from the Property Value field. For more information about adding a property value to a data field, see "Adding a Property and Property Value to a Data Field" on page 13-12. ■ DateFieldWithDialog: This data field is displayed on the generated form as a text field. If this text field is display-only, a user can use the field only to run a query. Otherwise, the user can also populate the field with a date and time (by double-clicking the field and selecting a date and time from the Date & Time window). Then, this date and time can be saved to the database. ■ CheckBox: This data field is displayed on the generated form as a check box. If this check box is display-only, a user can only see whether the check box is selected or deselected. Otherwise, the user can also select or deselect the check box, and save this setting to the database. ■ PasswordField: The text entered in this field is displayed as a series of asterisk (*) characters. If the name of the column with field type as Password Field is PASSWORD and a password policy is attached to the associated resource object, the password entered in this field is verified against the password policy. ■ RadioButton: This data field is displayed on the generated form as an option. A user can select or deselect the radio button, and save this setting to the database. ■ DataCombobox: This data field is displayed on the generated form as a list. A user can select an item from the list and save this selection to the database. ■ DisplayOnlyField: This data field is not enabled for the user to enter a value. This type of fields can only display data based on values in other fields.

Table 13–2 (Cont.) Fields of the Additional Columns Tab

Name	Description
Default Value	<p>This value is displayed in the associated data field after the form is generated and if no other default value was specified from the following scenarios:</p> <ul style="list-style-type: none"> ▪ A pre-populate adapter, which is attached to the form field, is run. ▪ A data flow exists between a field of a custom form assigned to one process and a field of a custom form associated with another process. ▪ A resource object, which has been requested for an organization, has a custom form attached to it. In addition, one of the fields of this custom form has a default value associated with it. It is strongly recommended that you do not specify default values for passwords and encrypted fields.
Order	<p>The sequence number that represents where the data field is positioned on the generated form.</p> <p>For example, a data field with an order number of 2 is displayed below a data field with an order number of 1.</p>
Application Profile	<p>This check box designates if the most recent value of this field should be displayed on the Object Profile tab of the Users form after the resource associated with this form has been provisioned to the user and achieved the Enabled status.</p> <p>If this check box is selected, the label and value of this field is displayed on the Object Profile tab of the Users form for users provisioned with the resource.</p>
Encrypted	<p>This check box determines if the information, which is displayed in the associated data field, is to be encrypted when it is transmitted between the server and the client.</p> <p>If this check box is selected, the information that is displayed in the data field is encrypted when it is transmitted between the client and the server.</p>

13.1.2.1.1 Adding a Data Field to a Form

To add a data field to a form:

Note: Password fields are encrypted by default. When a data field of password field type is created, the value is displayed as asterisk (*) characters in the Administrative and User Console, and the data is encrypted in the database.

1. In the Additional Columns tab, click **Add**.
A blank row is displayed in the Additional Columns tab.
2. In the **Name** field, enter the name of the data field, which is displayed in the database, and is recognized by Oracle Identity Manager.

Note: This name consists of the <TABLENAME_> prefix, followed by the name of the data field.

For example, if the name that is displayed in the **Table Name** field is **UD_PASSWORD**, and the name for the data field is **USERNAME**, the data field name that is displayed in the database and Oracle Identity Manager recognizes, would be **UD_PASSWORD_USERNAME**.

3. Double-click the **Variant Type** lookup field.

From the Lookup window that is displayed, select the variant type for the data field.

Currently, a data field can have one of nine variant types: Byte, Double, Date, Byte Array, Boolean, Long, String, Short, and Integer.

4. In the **Length** field, enter the length (in characters) of the data field.
5. In the **Field Label** field, enter the label that will be associated with the data field.

This label is displayed next to the data field on the form that is generated by Oracle Identity Manager.

6. Double-click the **Field Type** lookup field.

From the Lookup dialog box that is displayed, select the data type for the data field. Presently, a data field can have one of nine data types: Text Field, Lookup Field, Text Area, IT Resource Lookup Field, Date Field, Check Box, Password Field, Radio Button, and box.

See Also: [Table 13-2](#) for more information about data types

7. In the **Default Value** field, enter the value that is displayed in the associated data field once the form is generated, and if no other default value has been specified.

See Also: [Table 13-2](#) for more information about the scenarios where a default value could be set

8. In the **Order** field, enter the sequence number, which will represent where the data field will be positioned on the generated form.

For example, a data field with an order number of 2 is displayed below a data field with an order number of 1.

9. If you want a specific organization or user's values to supersede the value that is displayed in the **Default Value** field, select the **Application Profile** check box. Otherwise, go to Step 10.
10. If you want the information that is displayed in the data field to be encrypted when it is transmitted between the client and the server, select the **Encrypted** check box. Otherwise, go to Step 11.
11. Click **Save**.

13.1.2.1.2 Removing a Data Field From a Form

To remove a data field from a form:

Note: While adding a new field, if you assign it the same name as a field that was removed, the variant type (data type) of the new field remains the same as that of the field that was removed. For example, suppose you remove the Addr1 field to which the String variant type was applied. You create a field with the same name and apply the Boolean variant type to it. Now, when you view or use the form on which the new Addr1 field is added, the variant type of the field is String and not Boolean.

1. Delete all properties that are associated with the data field you want to remove by following the instructions in [Section 13.1.2.4.3, "Removing a Property and Property Value From a Data Field"](#).
2. Select the data field that you want to remove.
3. Click **Delete**.

The data field is removed from the form.

While adding a new field, if you assign it the same name as a field that was removed, the variant type (data type) of the new field remains the same as that of the field that was removed. For example, suppose you remove the Addr1 field to which the String variant type was applied. You create a field with the same name and apply the Boolean variant type to it. Now, when you view or use the form on which the new Addr1 field is added, the variant type of the field is String and not Boolean.

13.1.2.2 Child Table(s) Tab

Sometime you might have to add the same data fields to multiple forms that are created by using the Form Designer form. There are two ways to do this:

- You can add the data fields to each form manually, through the form's **Additional Columns** tab.
- You can group the data fields together and save them under one form name. Then, you can assign this form to each form that requires these data fields.

If this form contains the data fields that are required by another form, it is known as a child table.

Assigning child tables to a form increases your efficiency as a user. Without child tables, for every form that needs data fields, you would have to set the parameters for each field. For example, if five forms require the identical data field, you would have to set the parameters for this field five, separate times (one for each form).

If you use a child table for one form, and decide that you want to apply it to another form, the Design Console enables you to do so. Remove the child table from the first form, and assign it to the target form. This way, the child table that you assign to one form can be reused for all forms created with the Form Designer form.

You can configure Oracle Identity Manager to perform one of the following actions in a column of a child table:

- **Insert:** Adds a new value to the designated column of the child table
- **Update:** Modifies an existing value from the corresponding column of the child table
- **Delete:** Removes a value from the designated column of the child table

See Also: See [Section 12.3, "Process Definition Form"](#) for more information about setting up Oracle Identity Manager to insert, edit, or delete a value from in a column of a child table

For example, suppose that the UD_SOUTH child table is assigned to the **Results of 1Q 2004 Sales** form (represented by the UD_SALES2 table name). After this form is started, the data fields in the UD_SOUTH child table are displayed in the form.

The following sections describe how to assign a child table to a form and how to remove a child table from a form.

Note: If the form, which is represented by the child table, has not been made active, you cannot assign it to the parent form.

13.1.2.2.1 Assigning a Child Table to a Form

To assign a child table to a form:

Note: If the form that is represented by the child table is active, it will not be displayed in the Assignment window, and you will not be able to assign it to the parent form.

1. Click **Assign**.

The Assignment window is displayed.

2. From this window, select the child table, and assign it to the form.

3. Click **OK**.

The selected child table is assigned to the form.

13.1.2.2.2 Removing a Child Table from a Form

To remove a child table from a form:

1. Select the child table that you want to remove.

2. Click **Delete**.

The child table is removed from the form.

13.1.2.3 Object Permissions Tab

You use this tab to select the user groups that can add, modify, and remove information from a custom form when it is instantiated.

When the **Allow Insert** check box is selected, the corresponding user group can add information into the fields of the user-created form. If this check box is not selected, the user group cannot populate the fields of this form.

When the **Allow Update** check box is selected, the associated user group can modify existing information in the fields of the user-created form. If this check box is not selected, the user group cannot edit the fields of this form.

When the **Allow Delete** check box is selected, the corresponding user group can delete data from instantiations of the user-created form. If this check box is not selected, the user group cannot delete data from fields of this form (when it is instantiated).

[Figure 13-2](#) shows the Object Permissions tab of the Form Designer Form.

Figure 13–2 Object Permissions Tab of the Form Designer Form

The screenshot shows the 'Object Permissions' tab in the Form Designer. The top section contains 'Table Information' with fields for Table Name (UD_SOLARIS), Description (Access to Solaris for Engineering), Form Type (Process), and Object Name (Solaris). Below this is 'Version Information' showing Latest and Active Versions as 'Initial Version', and 'Operations' with a Current Version dropdown and buttons for 'Create New Version' and 'Make Version Active'. The bottom section is a table with columns for 'Group Name', 'Allow Insert', 'Allow Update', and 'Allow Delete'. The table lists five user groups: SYSTEM ADMINISTRATORS, Web Client Group, Sales Engineer Group, Project L7 Admin Group, and ALL USERS. All groups have 'Allow Insert', 'Allow Update', and 'Allow Delete' checked.

Group Name	Allow Insert	Allow Update	Allow Delete
1 SYSTEM ADMINISTRATORS	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
2 Web Client Group	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
3 Sales Engineer Group	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
4 Project L7 Admin Group	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
5 ALL USERS	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Suppose the SYSTEM ADMINISTRATORS user group can create, modify, and delete information that is displayed in the Results of 1Q 2004 Sales form (represented by the UD_SALES2 name in the Table Name field). The IT DEPARTMENT user group can only delete records of this form (its **Allow Insert** and **Allow Update** check boxes are not selected). The HR DEPARTMENT user group can create and modify information from within the Results of 1Q 2004 Sales form. However, because the **Allow Delete** check box is not selected, this user group is not able to delete this information.

The following section describes how to assign a user group to a user-created form, and remove a user group from a user-created form.

13.1.2.3.1 Assigning a User Group to a User-Created Form

To assign a user group to a user-created form:

1. Click **Assign**.
The Assignment dialog box is displayed.
2. Select the user group, and assign it to the form that was created by a user.
3. Click **OK**.
The user group is displayed in the **Object Permissions** tab.
4. If you do not want this user group to be able to add information into a record of the user-created form, double-click the corresponding **Allow Insert** check box. Otherwise, go to Step 5.
5. If you do not want this user group to be able to modify information from within a record of the user-created form, double-click the associated **Allow Update** check box. Otherwise, go to Step 6.
6. If you do not want this user group to be able to delete a record of the user-created form, double-click the corresponding **Allow Delete** check box. Otherwise, go to Step 7.
7. Click **Save**.
The user group is assigned to the user-created form.

13.1.2.3.2 Removing a User Group From a User-Created Form

To remove a user group from a user-created form:

1. Select the user group that you want to remove.
2. Click **Delete**.

The user group is removed from the user-created form.

13.1.2.4 Properties Tab

Figure 13–3 shows the Properties Tab of the Form Designer Form. You use this tab to assign properties and property values to the data fields that are displayed on the form that is created through the Form Designer form.

Figure 13–3 Properties Tab of the Form Designer Form

For example, suppose that the Results of 1Q 2004 Sales form has two data fields: **User Name** and **Password**. Each data field contains the following properties:

- **Required**, which determines whether or not the data field must be populated for the generated form to be saved. The default value for the **Required** property is `false`.
- **Visible Field**, which establishes whether the data field is displayed on the form, once Oracle Identity Manager generates the form. The default value for the **Visible Field** property is `true`.

Because the property values for the **Required** and **Visible Field** properties are `true` for both data fields, once the Results of 1Q 2004 Sales form is generated, both of these data fields are displayed. In addition, each field must be populated for the form to be saved.

The following sections describe how to add a property and property value to a data field, and how to remove them from the data field.

Note: The Properties tab is grayed out until you create a data field for the form by using the **Additional Columns** tab.

For more information about the properties and property values you can select, see "[Rule Elements, Variables, Data Types, and System Properties](#)" on page 30-1.

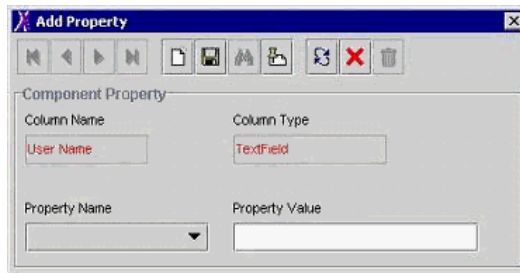
13.1.2.4.1 Adding a Property and Property Value to a Data Field

To add a property and property value to a data field:

1. Select the data field to which you want to add a property and property value.
2. Click **Add Property**.

The Add Property dialog box is displayed, as shown in [Figure 13-4](#).

Figure 13-4 Add Property Dialog Box



Note: The text that is displayed in the Column Name and Column Type fields are the names and types of data fields you selected.

In this example, the User Name data field was selected (as indicated by User Name displayed in the **Column Name** field). In addition, the data type of this field is a text field.

[Table 13-3](#) lists the fields of the Add Property dialog box.

Table 13-3 Fields of the Add Property Dialog Box

Name	Description
Column Name	The name of the data field.
Column Type	The data type of the data field.
Property Name	From this box, select the property for the data field.
Property Value	In this field, enter the property value, which is associated with the property that is displayed in the Property Name box.

Note: The menu items displayed in the Property Name box reflect the data type of the selected data field.

3. Set the parameters for the property and property value that you are adding to the data field. [Figure 13-5](#) shows the Add Property dialog box with values.

Figure 13–5 Add Property Dialog Box - Filled

For this example, because the value of the Required property for the User Name data field was set to true, once the associated form is generated, this field must be populated. Otherwise, the form cannot be saved.

See Also: See ["Rule Elements, Variables, Data Types, and System Properties"](#) on page 30-1 for more information about the parameters and property values to select

4. From the Add Property window's Toolbar, click **Save**.
5. Click **Close**.

The property and property value are added to the data field.

13.1.2.4.2 Adding a Property and Property Value for Customized Look up Query

To add a property and a property value for a customized lookup query:

1. Select the data field to which you want to add a property and a property value.
2. Click **Add Property**.

The Add Property dialog box is displayed, as shown in [Figure 13–6](#).

Figure 13–6 Add Property Dialog Box

Note: The text that is displayed in the Column Name and Column Type fields shows the name and type of the data field you selected (from the Properties tab of the Form Designer).

In this example, the **Name** data field was selected (as indicated by **Name** displayed in the **Column Name** field). In addition, the data type of this field is a lookup field.

The boxes of the Add Property dialog box are used to help build the WHERE clause in the custom lookup query. As you select the values for each box (from the menu), the WHERE clause is appended to the custom lookup query.

Table 13–4 describes the regions of the Add Property dialog box. Initially, all the fields are grayed out. After you have defined the lookup query and clicked **Save**, the fields become active.

Table 13–4 Fields of the Add Property Dialog Box

Name	Description
Column Name	The name of the data field.
Column Type	The data type of the data field.
Property Name	From this list, select the property for the data field.
Property Value	<p>In this field, enter the property value, which is associated with the property that is displayed in the Property Name box.</p> <p>In the case of a lookup query, you must specify both the Oracle Identity Manager form and field, which will be referenced for the query and will be recognized by the database.</p> <p>For example, if Oracle Identity Manager is referring to the user's login, you enter select usr_key fromusr in the Property Value field. After clicking Save, the Filter Column is active with all the columns of tables.</p>
Filter Column	<p>This is the Oracle Identity Manager form field that is referenced for the lookup query, and which is recognized by the database. This field is populated with all columns of table specified in the Property Value field. If multiple tables are used in the query, all tables are shown.</p> <p>For example, usr .USR_LOGIN signifies that Oracle Identity Manager will see the User Login field from the Users form for the lookup query.</p>
Source	<p>After the Filter Column variable is selected, the Source field is populated with all possible sources of value. The list of values in this field is dependent upon the type of form, for which the lookup field is being defined. For instance, the list displayed is different if the lookup query is for an Object form or a Process form. The Source field is a user-friendly name for the value that is displayed in the Filter Column box.</p> <p>For example, Requester Information refers to the usr.USR portion of the Filter Column value.</p>
Field	<p>This field is populated based on what value is selected in the Source field. Use this field to create the SELECT statement, which is needed for the column name.</p> <p>For example, the User Login corresponds to the _LOGIN part in the Filter Column value.</p>

Note: The menu items displayed in the **Property Name** box show the data type of the selected data field.

The **Source** and **Field** boxes of the Add Property dialog box are applicable only when **Lookup Query** is displayed in **Property Name**.

3. Set the parameters for the property and the property value that you are adding to the data field. [Figure 13–7](#) shows the Edit Property dialog box.

Figure 13–7 Edit Property Dialog Box

13.1.2.4.3 Removing a Property and Property Value From a Data Field

To remove a property and property value from a data field:

1. Select the property and the property value that you want to remove.
2. Click **Delete Property**.

The property and its associated value are removed from the data field.

13.1.2.5 Administrators Tab

This tab is used to select the user groups that can view, modify, and delete the current record of the form that was created by a user by using the Form Designer form.

When the **Write** check box is selected, the corresponding user group can view and modify information for the current record of the form. If this check box is not selected, the user group cannot view or edit information for this record.

When the **Delete** check box is selected, the associated user group can remove information from the current record of the form. If this check box is not selected, the user group cannot delete information from this record.

[Figure 13–8](#) shows the Administrators tab of the Form Designer Form.

Figure 13–8 Administrators Tab of the Form Designer Form

The following sections describe how to assign administrative privileges to a user group for a record of a user-created form and remove administrative privileges from a user group for a record of a user-created form.

13.1.2.5.1 Assigning Privileges to a User Group for a Record of a User-Created Form

To assign administrative privileges to a user group for a record of a user-created form:

1. Click **Assign**.
The Assignment dialog box is displayed.
2. Select the user group, and assign it to the record of the user-created form.
3. Click **OK**.
The user group is displayed in the **Administrators** tab.
4. If you want this user group to be able to create and modify information for the current record of the user-created form, double-click the corresponding **Write** check box. Otherwise, go to Step 5.
5. If you want this user group to be able to remove information from the current record of the user-created form, double-click the associated **Delete** check box. Otherwise, go to Step 6.
6. Click **Save**.
The user group now has administrative privileges for this record of the user-created form.

13.1.2.5.2 Removing User Group Privileges for a Record of a User-Created Form

To remove administrative privileges from a user group for a record of a user-created form:

1. Select the user group that you want to remove.
2. Click **Delete**.

The user group no longer has administrative privileges for this record of the user-created form.

13.1.2.6 Usage Tab

In this tab, you can see the resource objects and processes to which the current form has been assigned.

For example, the **Solaris** form (represented by the **UD_SOLARIS** name in the **Table Name** field) was created and assigned to both the Solaris resource object and provisioning process.

Note: The table name contains the **UD_** prefix, followed by the form name. For this example, because the name of the form is Solaris, its table name is **UD_SOLARIS**.

This tab will be populated with information only after you click **Make Version Active**.

13.1.2.7 Pre-Populate Tab

You use this tab is to do the following:

- Attach a pre-populate adapter to a data field of the user-created form.
- Select the rule that will determine if this adapter will be executed to populate the designated data field with information.
- Set the priority number for the selected rule.
- Map the adapter variables of the prepopulate adapter to their correct locations.

See Also: [Chapter 3, "Using Adapters"](#) for more information about prepopulate adapters, attaching pre-populate adapters to fields of user-created forms, or mapping the variables of a pre-populate adapter

13.1.2.8 Default Columns Tab

A form that is created by using the Form Designer form is composed of two types of data fields:

- Data fields that are created by a user (by using the **Additional Columns** tab)
- Data fields that are created by Oracle Identity Manager, and added to the form, once the form is created

Through the **Default Columns** tab, you can see the names, variant types, and lengths of the data fields, which are added, by default, to a user-created form. As a result, by viewing these data fields, you can see all data fields for this type of form, without starting SQL*Plus, or a similar database application.

13.1.2.9 User Defined Fields Tab

This tab is used to view and access any user-defined fields that were created for the Form Designer form. Once a user-defined field has been created, it is displayed on this tab and is able to accept and supply data.

See Also: See [Section 15.3, "User Defined Field Definition Form"](#) for instructions about how to create fields for user-created forms

13.1.3 Creating an Additional Version of a Form

Sometimes, when you create a form and populate the tabs of the Form Designer form with information, so the form will work with the process or resource object to which it will be assigned, you might want to create a different version of the form. This way, you can modify this version, without changing the original version of the form.

To create an additional version of a form:

1. Open the Form Designer form.
2. Search for the specific form of which you want to create a different version.
3. Click the **Current Version** box.

From the drop-down menu that is displayed, select the version of the form of which you are creating an additional version.

4. Click the **Create New Version** button.

The Create a New Version window is displayed.

5. In the **Label** field, enter the name of the additional version of the form.
6. From the Create a New Version window's toolbar, click **Save**.
7. From this toolbar, click **Close**.

The additional version of the form is created. When you click the **Current Version** box, the version's name, which you entered into the **Label** field in Step 5, is displayed. By selecting this version, you can populate the tabs of the Form Designer form with information, without changing the original version of the form.

13.2 Error Message Definition Form

The Error Message Definition form, as shown in [Figure 13-9](#), is in the Development Tools folder. It is used to:

- Create the error messages that are displayed in dialog boxes when certain problems occur.
- Define the error messages that users can access when they create error handler tasks by using the Adapter Factory form.

The error messages you create are displayed on the Administrative and User Console if they are added to an adapter definition while creating a new adapter by using an error handler logic task based on a failure condition.

Note: If an entity adapter is attached to a process form or an object form for validation of field values, these adapters will run if you edit data in these forms after completing direct or request provisioning.

Oracle Identity Manager 11g Release 1 (11.1.1) does not support creating new entity adapters.

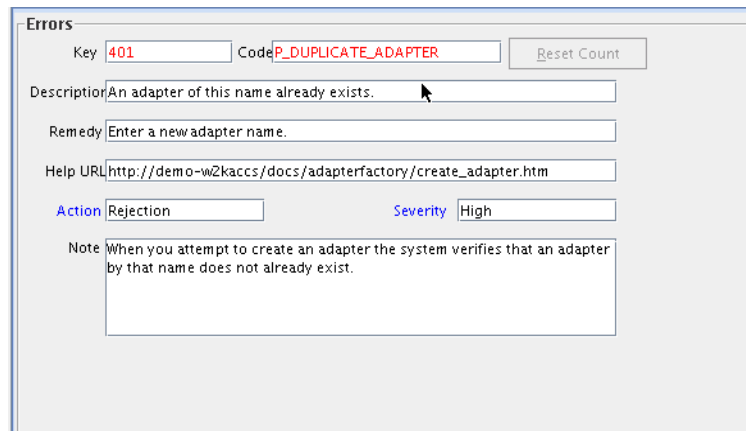
Figure 13–9 Error Message Definition Form


Table 13–5 describes the data fields of the Error Message Definition form.

Table 13–5 Fields of the Error Message Definition Form

Field Name	Description
Key	The error message definition's unique, system-generated identification number.
Code	The code that represents the error message definition.
Reset Count	When you click this button, Oracle Identity Manager resets the counter to zero. This counter is the number of times the error message is displayed.
Description	A description of the error message.
Remedy	A description of how to correct the condition that caused the error message to be displayed.
Help URL	The link to the URL that contains an online Help topic for this error message.
Action	A one-letter code, representing the seriousness of the condition that causes the error message to be displayed. An error message has three levels of seriousness: Error (E), Rejection (R), and Fatal Rejection (F).
Severity	For classification purposes, you can categorize the seriousness of the condition that results in the error message being displayed, even further. An error message has five sub-levels of severity: None (N), Low (L), Medium (M), High (H), and Crash (C).
Note	Explanatory information about the error message.

13.2.1 Creating an Error Message

When you create an error message, Oracle Identity Manager populates the **Key** field with a unique identification number. When a condition occurs that causes the error message to be displayed, the text in the **Description** field is displayed in a dialog box.

Note: After you create an error message definition, to reset the count of how many times the error message is displayed, click the **Reset Count** button. This resets the count to zero.

To create an error message:

1. Open the Error Messaging Definition form.
2. In the **Code** field, enter the code that represents the error message definition.
3. In the **Description** field, enter a description for the error message.
4. In the **Remedy** field, you can enter a description for how to correct the condition that causes the error message to be displayed.
5. In the **Help URL** field, you can enter the link to the URL that contains an online Help topic for this error message.
6. (Optional) Double-click the **Action Lookup** field.

From the Lookup dialog box that is displayed, you can select a code that represents the seriousness of the condition that causes the error message to be displayed. These codes, listed by degree of seriousness (from lowest to highest), are:

- Error (E). Oracle Identity Manager stores the error message, and stops any related operations from being triggered. Instead, the operation rolls back to the previous operation.
 - Reject (R). Oracle Identity Manager stores the rejection message, but it does not prevent subsequent operations from being executed.
 - Fatal Reject (F). Oracle Identity Manager stores the rejection message, and it stops any subsequent operations from being triggered. However, it stores all operations that were executed up to the fatal rejection.
7. (Optional) Double-click the **Severity Lookup** field. From the Lookup dialog box that is displayed, you can select a code (None (N), Low (L), Medium (M), High (H), or Crash (C)). This code presents a detailed classification of the code that is displayed in the **Action** lookup field.
 8. In the **Note** field, enter explanatory information about the error message.
 9. Click **Save**.

The error message is created.

After creating error messages by using the Error Message Definition form, you must add new error codes and advice messages in the Oracle Identity Manager `customResources.properties` resource bundle. These localized error codes and advice messages will be shown in the Administrative and User Console.

Customizing Reconciliation Operations

Oracle Identity Manager provides connectors for reconciliation of users/accounts from various target systems, such as Microsoft Active Directory, Sun Java System Directory, Oracle Internet Directory, and Oracle E-Business Suite. For information about these connectors, see Oracle Identity Manager Connectors Documentation in the Oracle Technology Network (OTN) Web site at the following URL:

<http://www.oracle.com/technetwork/indexes/documentation/index.html>

However, to create a custom connector, you must develop a new scheduled task that performs the following:

1. Retrieve user/account information from the target system.
2. Use reconciliation APIs to create reconciliation events to submit event data.
3. Create events for creating, modifying, or deleting an entity.

See Also: [Chapter 6, "Developing Scheduled Tasks"](#) for information about developing a scheduled task

14.1 Developing Reconciliation Scheduled Tasks

To connect to a specific target system, you must:

- Create a new IT resource type
- Define a new IT resource
- Use the IT resource as an input parameter for the scheduled task

See Also: *Oracle Fusion Middleware Java API Reference for Oracle Identity Manager* for information about the APIs to lookup IT resource definition

In Oracle Identity Manager, a provisioning process and a process instance is associated with activities related to users or accounts. This provides a hook or point to add customizations upon various actions.

Changes to the user state or the account state can occur via direct APIs or reconciliation. The changes can be of many types, such as:

See Also: ["Understanding Reconciliation APIs"](#) on page 14-2 for information about the reconciliation APIs

- Data change in the user or account profile

- Status change, such as enable or disable
- Changes to user based on attestation processes
- Organization change
- Attribute propagation
- Password propagation

For each of these changes, the process definition provides a facility to add hooks to be run upon any of these changes. For reconciliation, the process definition provides the hooks in the form of the following conditional tasks:

- **Reconciliation Insert Received:** This conditional task is inserted when an account is created via reconciliation.
- **Reconciliation Update Received:** This conditional task is inserted when an existing account linked to a user is updated via reconciliation. Data in the process form or status of the account are updated.
- **Reconciliation Delete Received:** This conditional task is inserted when an existing account is revoked via reconciliation.

These tasks provide starting points for the workflows. You can create custom workflows in the provisioning process, and create a dependency between the reconciliation trigger tasks and the workflows. This causes the workflows to be run upon the respective triggers.

Every reconciliation event that is successfully linked to a user or an account inserts a single trigger from the conditional tasks. All the data in the user profile and the account profile is available as context-sensitive data for any adapter that is attached to one of these dependant tasks.

See Also: [Part V, "Requests and Approval Processes"](#) and [Part I, "Concepts"](#) for details about creating conditional tasks, adapters, and dependencies

14.2 Understanding Reconciliation APIs

The reconciliation APIs are a set of published APIs that can be used to create reconciliation events with single-valued and multi-valued attribute data and other features.

Reconciliation connector developers must use these APIs to push data to the reconciliation event repository.

See Also: [Chapter 31, "Using APIs"](#) for more information about using APIs in Oracle Identity Manager

Most of these APIs existed in earlier versions of Oracle Identity Manager. However, in 11g Release 1 (11.1.1), the implementation has changed and is based on the new reconciliation architecture introduced in the release.

Existing standard connectors also use these APIs; since the earlier APIs continue to be supported, no changes are necessary to those connectors.

`callingEndOfJobAPI` is the only new reconciliation API in 11g Release 1 (11.1.1).

Each run of a connector is known as a job. In 11g Release 1 (11.1.1), reconciliation events are submitted to the reconciliation engine in batches. At the end of a job, the scheduler (which executes the connector scheduled task) executes a listener, which in

turn invokes the `callingEndOfJOBAPI`. This API submits any open batch for processing to the reconciliation engine.

The API calls are similar for Multilanguage Supported (MLS) and non-MLS data. The connector passes in data to be reconciled as a `HashMap`. The difference is that if an attribute is MLS-enabled, then the key is the attribute name, while the value is another `HashMap` of MLS data. The keys of this MLS-specific `HashMap` are language codes, and the values are the corresponding locale-specific data obtained from target system. If there is any MLS data that does not have a locale defined with it in the target system, that data is passed with key "base" in the MLS input data `HashMap`.

14.3 Postprocessing for Trusted Reconciliation

If the user login is not passed for trusted reconciliation, then the login handler generates the user login. The password is generated in postprocessing event handler. You can configure Oracle Identity Manager to send notification for the same.

Notification is sent only when the value of the `Recon.SEND_NOTIFICATION` system property is set to true. See "System Properties in Oracle Identity Manager" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager* for information about the `Recon.SEND_NOTIFICATION` system property.

In SSO disabled environment, for user creation via reconciliation, both the user login and password are generated in postprocess handlers and a single notification is sent for both user login and password.

In SSO enabled environment, because the password is not to be generated, if login is generated in postprocess handler, then notification is sent only for the user login.

14.4 Troubleshooting Reconciliation

Before troubleshooting issues related to reconciliation, change the reconciliation logging level to INFO. To do so, add the following line in the `logging.xml` file, and restart Oracle Identity Manager.

```
<LOGGER NAME="ORACLE.IAM.RECONCILIATION" LEVEL="INFO"/>
```

This section describes troubleshooting reconciliation issues in the following sections:

- [Troubleshooting General Reconciliation Issues](#)
- [Troubleshooting Trusted Source Reconciliation](#)
- [Troubleshooting Target Resource Reconciliation](#)
- [Troubleshooting Database-Related Reconciliation Issues](#)

14.4.1 Troubleshooting General Reconciliation Issues

[Table 14–1](#) lists the troubleshooting steps that you can perform if you encounter reconciliation errors:

Table 14–1 Troubleshooting Reconciliation

Problem	Solution
Failure in processing events	<p>The error details can be obtained from the reconciliation tables, such as:</p> <ul style="list-style-type: none"> For batch processing, the exception is stored in RECON_BATCHES.RB_NOTE column For single event processing, the exception is stored in RECON_EVENTS.RE_NOTE column
Failure occurring in kernel orchestration handler	<p>The orchestration ID can be tracked from the reconciliation table, which can further be used to check the status of related handlers, such as:</p> <ul style="list-style-type: none"> For batch processing, the postprocess only orchestration ID can be read from the RECON_BATCHES.RB_NOTE column For single event processing, end-to-end orchestration ID can be read from the RECON_EVENTS.RE_NOTE column <p>There is no UI that displays LDAP synchronization during reconciliation. Therefore, you can only track LDAP success or failure by checking the status of LDAP sync event handlers in orchestration based on the ID available in RB_NOTE/RE_NOTE columns.</p>
After a job run, a lot of events are in Data Received status	<p>Check if related batches are in Ready For Processing status by using the following statement:</p> <pre>select rb_batch_status, rb_note from recon_batches where rb_batch_status = 'Ready For Processing' and rj_key = JOB_ID_ON_UI</pre> <p>In addition, in the RECON_BATCHES.RB_NOTE, there is some generic exception, such as Connection issue. Fix the issue, and then perform any one of the following:</p> <ul style="list-style-type: none"> If there is a lot of data, then rerun the reconciliation job. There is a scheduled task provided for manual retry of such failed batches Retry Reconciliation Batch. This can be used for retrying specific batches only. Multiple comma-separated batches are supported. There is no predefined job associated with this schedule task. A job can be created as required.
The following error is generated when performing user update for trusted source reconciliation:	<p>For of trusted source reconciliation, if the matching rule is based on Usr_login, then the matching rule must not be case-sensitive. Otherwise, updating users work as creating users, and the error might be generated.</p> <pre>ORA Error Code =>ORA-00001: unique constraint (.) violated</pre>

14.4.2 Troubleshooting Trusted Source Reconciliation

This section describes the following issues related to trusted source reconciliation:

No Reconciliation Profile

For missing reconciliation profile, the following error is logged:

```
[EXEC] ORACLE.IAM.RECONCILIATION.EXCEPTION.RECONCILIATIONEXCEPTION: EXCEPTION
OCCURRED WHILE INSERTING DATA INTO TABLE RA_TESTRORECON7 DUE TO RA_TESTRORECON7
[EXEC] AT
ORACLE.IAM.RECONCILIATION.IMPL.RECONOPERATIONSSERVICEIMPL.RECONEVENT(RECONOPERATIO
```

```

NSSERVICEIMPL.JAVA)
[EXEC] AT
ORACLE.IAM.RECONCILIATION.IMPL.RECONOPERATIONSSERVICEIMPL.CREATECONCILIATIONEVENT(RECONOPERATIONSSERVICEIMPL.JAVA)
[EXEC] AT
ORACLE.IAM.RECONCILIATION.IMPL.RECONOPERATIONSSERVICEIMPL.CREATECONCILIATIONEVENT(RECONOPERATIONSSERVICEIMPL.JAVA)
[EXEC] AT
ORACLE.IAM.RECONCILIATION.IMPL.RECONOPERATIONSSERVICEIMPL.CREATECONCILIATIONEVENT(RECONOPERATIONSSERVICEIMPL.JAVA)
[EXEC] AT
ORACLE.IAM.RECONCILIATION.IMPL.RECONOPERATIONSSERVICEIMPL.CREATECONCILIATIONEVENT(RECONOPERATIONSSERVICEIMPL.JAVA)
[EXEC] AT
ORACLE.IAM.RECONCILIATION.API.RECONOPERATIONSSERVICEEJB.CREATECONCILIATIONEVENTX(UNKNOWN SOURCE)
.....
.....
.....
[EXEC] CAUSED BY: ORACLE.IAM.PLATFORM.ENTITYMGR.NOSUCHENTITYEXCEPTION:
RA_TESTRORECON7
[EXEC] AT
ORACLE.IAM.PLATFORM.ENTITYMGR.IMPL.ENTITYMANAGERCONFIGIMPL.EXISTS(ENTITYMANAGERCONFIGIMPL.JAVA)
[EXEC] AT
ORACLE.IAM.PLATFORM.ENTITYMGR.IMPL.ENTITYMANAGERCONFIGIMPL.GETDATAPROVIDER(ENTITYMANAGERCONFIGIMPL.JAVA)
[EXEC] AT
ORACLE.IAM.PLATFORM.ENTITYMGR.IMPL.ENTITYMANAGERIMPL.GETENTITYCAPABILITY(ENTITYMANAGERIMPL.JAVA)
[EXEC] AT
ORACLE.IAM.PLATFORM.ENTITYMGR.IMPL.ENTITYMANAGERIMPL.GETENTITYCAPABILITY(ENTITYMANAGERIMPL.JAVA)
[EXEC] AT
ORACLE.IAM.PLATFORM.ENTITYMGR.IMPL.ENTITYMANAGERIMPL.CREATEENTITY(ENTITYMANAGERIMPL.JAVA)
[EXEC] AT
ORACLE.IAM.PLATFORM.ENTITYMGR.IMPL.ENTITYMANAGERIMPL.CREATEENTITY(ENTITYMANAGERIMPL.JAVA)
[EXEC] AT
ORACLE.IAM.RECONCILIATION.IMPL.RECONOPERATIONSSERVICEIMPL.RECONEVENT(RECONOPERATIONSSERVICEIMPL.JAVA)
[EXEC] AT
ORACLE.IAM.RECONCILIATION.IMPL.RECONOPERATIONSSERVICEIMPL.CREATECONCILIATIONEVENT(RECONOPERATIONSSERVICEIMPL.JAVA)
[EXEC] AT
ORACLE.IAM.RECONCILIATION.IMPL.RECONOPERATIONSSERVICEIMPL.CREATECONCILIATIONEVENT(RECONOPERATIONSSERVICEIMPL.JAVA)
[EXEC] AT
ORACLE.IAM.RECONCILIATION.IMPL.RECONOPERATIONSSERVICEIMPL.CREATECONCILIATIONEVENT(RECONOPERATIONSSERVICEIMPL.JAVA)
[EXEC] AT
ORACLE.IAM.RECONCILIATION.IMPL.RECONOPERATIONSSERVICEIMPL.CREATECONCILIATIONEVENT(RECONOPERATIONSSERVICEIMPL.JAVA)
[EXEC] AT
ORACLE.IAM.RECONCILIATION.API.RECONOPERATIONSSERVICEEJB.CREATECONCILIATIONEVENTX(UNKNOWN SOURCE)
.....
.....
.....

```

Missing Reconciliation Field Mapping or Inactive Owner Rule

The following error is logged when:

- There are no reconciliation fields defined.
- Reconciliation fields are added but no mapping is defined.
- There is no matching rule defined or the rule is inactive.

```
[EXEC] ORACLE.IAM.PLATFORM.UTILS.SUPERRUNTIMEEXCEPTION: -100: ERROR OCCURED IN
XL_SP_RECONBLKUSRRQDCVALDNMTCH WHILE PROCESSING BATCH ID 4 ONE OR MORE INPUT
PARAMETER PASSED AS NULL
[EXEC] AT
ORACLE.IAM.RECONCILIATION.DAO.RECONACTIONDAO.EXECUTEBULKUSERMATCHCRUD (RECONACTIOND
AO.JAVA)
[EXEC] AT
ORACLE.IAM.RECONCILIATION.IMPL.USERHANDLER.EXECUTEBULKCUD (USERHANDLER.JAVA)
[EXEC] AT
ORACLE.IAM.RECONCILIATION.IMPL.BASEENTITYTYPEHANDLER.PROCESS (BASEENTITYTYPEHANDLER
.JAVA)
[EXEC] AT
ORACLE.IAM.RECONCILIATION.IMPL.ACTIONENGINE.PROCESSBATCH (ACTIONENGINE.JAVA)
[EXEC] AT ORACLE.IAM.RECONCILIATION.IMPL.ACTIONENGINE.EXECUTE (ACTIONENGINE.JAVA)
[EXEC] AT ORACLE.IAM.RECONCILIATION.IMPL.ACTIONTASK.EXECUTE (ACTIONTASK.JAVA)
[EXEC] AT
ORACLE.IAM.PLATFORM.ASYNC.IMPL.TASKEXECUTOR.EXECUTEUNMANAGEDTASK (TASKEXECUTOR.JAVA
)
[EXEC] AT ORACLE.IAM.PLATFORM.ASYNC.IMPL.TASKEXECUTOR.EXECUTE (TASKEXECUTOR.JAVA)
[EXEC] AT
ORACLE.IAM.PLATFORM.ASYNC.MESSAGING.MESSAGERECEIVER.ONMESSAGE (MESSAGERECEIVER.JAVA
)
.....
.....
.....
[EXEC] CAUSED BY: ORACLE.IAM.PLATFORM.UTILS.SUPERRUNTIMEEXCEPTION: -100: ERROR
OCCURED IN XL_SP_RECONBLKUSRRQDCVALDNMTCH WHILE PROCESSING BATCH ID 4 ONE OR
MORE INPUT PARAMETER PASSED AS NULL
[EXEC] AT
ORACLE.IAM.RECONCILIATION.DAO.RECONACTIONDAO.EXECUTEBULKUSERMATCHCRUD (RECONACTIOND
AO.JAVA)
[EXEC] AT
ORACLE.IAM.RECONCILIATION.IMPL.USERHANDLER.EXECUTEBULKCUD (USERHANDLER.JAVA)
[EXEC] AT
ORACLE.IAM.RECONCILIATION.IMPL.BASEENTITYTYPEHANDLER.PROCESS (BASEENTITYTYPEHANDLER
.JAVA)
[EXEC] AT
ORACLE.IAM.RECONCILIATION.IMPL.ACTIONENGINE.PROCESSBATCH (ACTIONENGINE.JAVA)
[EXEC] AT ORACLE.IAM.RECONCILIATION.IMPL.ACTIONENGINE.EXECUTE (ACTIONENGINE.JAVA)
[EXEC] AT ORACLE.IAM.RECONCILIATION.IMPL.ACTIONTASK.EXECUTE (ACTIONTASK.JAVA)
[EXEC] AT
ORACLE.IAM.PLATFORM.ASYNC.IMPL.TASKEXECUTOR.EXECUTEUNMANAGEDTASK (TASKEXECUTOR.JAVA
)
```

Missing Reconciliation Action Rule

If proper reconciliation fields along with mapping are defined and the matching rule is also valid and active, but still the event status is No User Match Found, then probably

there are no action rules defined. After you add the reconciliation action rule, the user is created successfully.

14.4.3 Troubleshooting Target Resource Reconciliation

This section describes the following issues related to target resource reconciliation:

Missing Process Form

When you remove all process data fields mapping and regenerate the profile, the following error is logged:

```
[EXEC] <APR 19, 2011 11:13:48 PM PDT> <ERROR> <ORACLE.IAM.RECONCILIATION.IMPL>
<IAM-5010000> <GENERIC INFORMATION: {0}>
[EXEC] ORACLE.IAM.PLATFORM.UTILS.SUPERRUNTIMEEXCEPTION: -1: ERROR OCCURRED IN
XL_SP_RECONINPUTPARAMSVALDN ORA-20001: COMMA-SEPARATED LIST INVALID NEAR D
[EXEC] AT
ORACLE.IAM.RECONCILIATION.DAO.RECONACTIONDAO.EXECUTEBULKUSERMATCHCRUD (RECONACTIOND
AO.JAVA)
[EXEC] AT
ORACLE.IAM.RECONCILIATION.IMPL.USERHANDLER.EXECUTEBULKUCUD (USERHANDLER.JAVA)
[EXEC] AT
ORACLE.IAM.RECONCILIATION.IMPL.BASEENTITYTYPEHANDLER.PROCESS (BASEENTITYTYPEHANDLER
.JAVA)
[EXEC] AT
ORACLE.IAM.RECONCILIATION.IMPL.ACTIONENGINE.PROCESSBATCH (ACTIONENGINE.JAVA)
[EXEC] AT ORACLE.IAM.RECONCILIATION.IMPL.ACTIONENGINE.EXECUTE (ACTIONENGINE.JAVA)
[EXEC] AT ORACLE.IAM.RECONCILIATION.IMPL.ACTIONTASK.EXECUTE (ACTIONTASK.JAVA)
```

The possible cause of this error is that the process form is missing.

Missing Process Form Mapping or Missing Where Clause Selection

When you remove all process data fields mapping, add process form, and regenerate the profile, the following error is logged:

```
[EXEC] <APR 19, 2011 11:25:31 PM PDT> <WARNING> <JNDI> <BEA-050007> <AN ATTEMPT
WAS MADE TO LOOK UP NON-VERSIONED GLOBAL RESOURCE "QUEUE" FROM AN APPLICATION
VERSION "OIM [VERSION=11.1.1.5.0]". THIS CAN POTENTIALLY CAUSE CONFLICT OF THE
GLOBAL RESOURCE USAGES AMONG MULTIPLE APPLICATION VERSIONS.>
[EXEC] <APR 19, 2011 11:25:31 PM PDT> <ERROR> <ORACLE.IAM.RECONCILIATION.IMPL>
<IAM-5010000> <GENERIC INFORMATION: {0}>
[EXEC] JAVA.LANG.NULLPOINTEREXCEPTION
[EXEC] AT
ORACLE.IAM.RECONCILIATION.IMPL.CONFIG.PROFILE.GETACCOUNTMATCHINGRULESWHERECLAUSE (P
ROFILE.JAVA)
[EXEC] AT
ORACLE.IAM.RECONCILIATION.IMPL.PROFILEDATA.GETACCOUNTMATCHINGRULES (PROFILEDATA.JAV
A)
[EXEC] AT
ORACLE.IAM.RECONCILIATION.DAO.RECONACTIONDAO.EXECUTEBULKACCOUNTMATCHCRUD (RECONACTI
ONDAO.JAVA)
[EXEC] AT
ORACLE.IAM.RECONCILIATION.IMPL.ACCOUNTHANDLER.EXECUTEBULKUCUD (ACCOUNTHANDLER.JAVA)
[EXEC] AT
ORACLE.IAM.RECONCILIATION.IMPL.BASEENTITYTYPEHANDLER.PROCESS (BASEENTITYTYPEHANDLER
.JAVA)
[EXEC] AT
ORACLE.IAM.RECONCILIATION.IMPL.ACTIONENGINE.PROCESSBATCH (ACTIONENGINE.JAVA)
[EXEC] AT ORACLE.IAM.RECONCILIATION.IMPL.ACTIONENGINE.EXECUTE (ACTIONENGINE.JAVA)
[EXEC] AT ORACLE.IAM.RECONCILIATION.IMPL.ACTIONTASK.EXECUTE (ACTIONTASK.JAVA)
```

The possible causes of this error are missing process form mapping or missing where clause selection.

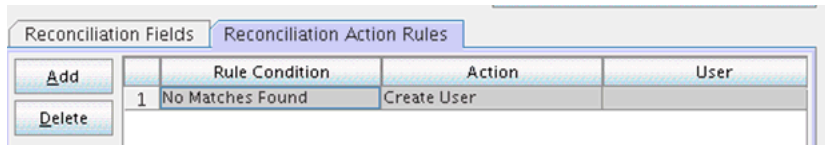
Missing Valid Action Rule

When you remove all process data fields mapping, add process form with mapping, add where clause selection, and regenerate the profile, the following error is logged:

```
[EXEC] <APR 19, 2011 11:36:20 PM PDT> <WARNING> <ORACLE.IAM.PLATFORM.ASYNC>
<BEA-000000> <EXCEPTION FOR ABOVE MESSAGE I.E IAM-0050000
[EXEC] ORACLE.IAM.PLATFORM.UTILS.SUPERRUNTIMEEXCEPTION:
ORACLE.IAM.PLATFORM.UTILS.SUPERRUNTIMEEXCEPTION: -15: ERROR IN
XL_SP_RECONBLKACNTRQDCMTCRUD WHILE PROCESSING BATCH ID 18 ERROR OCCURED IN
XL_SP_RECONBLKCHILDMTACNTRUD WHILE PROCESSING BATCH ID - 18 ACTION RULE 'CREATE
USER' FOR NO ENTITY MATCH FOUND IS INVALID. YOU MUST PASS A VALID NO USER MATCH
FOUND ACTION RULE. -20101 -ERROR- ORA-20101:
[EXEC] AT
ORACLE.IAM.RECONCILIATION.IMPL.ACTIONENGINE.PROCESSBATCH (ACTIONENGINE.JAVA)
[EXEC] AT ORACLE.IAM.RECONCILIATION.IMPL.ACTIONENGINE.EXECUTE (ACTIONENGINE.JAVA)
[EXEC] AT ORACLE.IAM.RECONCILIATION.IMPL.ACTIONTASK.EXECUTE (ACTIONTASK.JAVA)
```

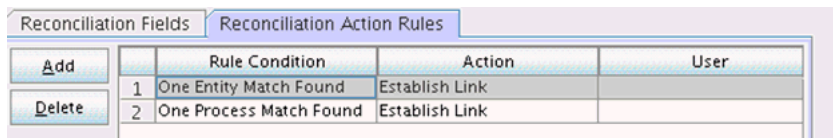
The possible cause of this error is that a valid action rule has not been added. [Figure 14–1](#) shows the Design Console screen with invalid action rule:

Figure 14–1 Invalid Action Rule



When you add a valid action rule, the account is created successfully. [Figure 14–2](#) shows the Design Console screen with valid action rule:

Figure 14–2 Valid Action Rule



Profile Not Generated

When you remove all process data fields mapping, add process form with mapping, add where clause selection, update the matching rule, and regenerate the profile, the profile is not generated and the same error as "Missing Valid Action Rule" on page 14-8 is logged.

Matching Rule is NULL or Inactive

When the matching rule is NULL or inactive, the following error is logged:

```
Thor.API.Exceptions.tcAPIException:
oracle.iam.reconciliation.exception.ReconciliationException: Matching rule where
clause is null
```

Update the profile XML file. If you manually update the profile XML file, use the following example XML for a well-defined and active matching rule in the profile:


```

<ReconUserMatchingRule repo-type="RDBMS" name="AS400TrustedUser Recon
Rule">
<RRL_UPDATE>1301687006000</RRL_UPDATE>
<RRL_VALID>1</RRL_VALID>
<RRL_OPERATOR>AND</RRL_OPERATOR>
<RRL_DESCRIPTION>AS400TrustedUser Recon Rule</RRL_DESCRIPTION>
<RRL_ACTIVE>1</RRL_ACTIVE>
<ReconRuleElement repo-type="RDBMS" id="RRE1">
<RRE_SEQUENCE>1</RRE_SEQUENCE>
<RRE_UPDATE>1301689575000</RRE_UPDATE>
<RRE_FIELDNAME>User Login</RRE_FIELDNAME>
<RRE_CASESENSITIVE>0</RRE_CASESENSITIVE>
<RRE_VALID>1</RRE_VALID>
<RRE_TRANSFORM>None</RRE_TRANSFORM>
<RRE_OPERATOR>Equals</RRE_OPERATOR>
<ORF_KEY Resource="AS400TrustedUser" ReconField="User Login"/>
</ReconRuleElement>
</ReconUserMatchingRule>

```

If changing through the Design Console, then update the rule and regenerate the profile.

Missing Child Data for an Event

No data is deleted when you do not pass any child data for an event, which happens with some connectors if all children are deleted on the target for an account.

Multiple Account Matches When Ad Hoc Linking an Account to a User

If you ad hoc link an account to a user, then based on the matching rule for account, there can be multiple account matches even though a particular user's account has been specified. This can be overcome by selecting an account from matched accounts list, and clicking **LINK** on the Administrative and User Console.

14.4.4 Troubleshooting Database-Related Reconciliation Issues

This section describes the following database-related issues for reconciliation:

Missing Critical Oracle Database 11g Release 1 Interim Patches

When the RDBMS interim patch# 7614692 is missing, the following error is logged:

```

ORA-02291: INTEGRITY CONSTRAINT (FK_RECON_EVENTS_USR) VIOLATED - PARENT KEY NOT
FOUND
[EXEC] ORA-06512: AT "OIM_SP_RECONBLKUSERCRUD"
[EXEC] ORA-06512: AT "OIM_SP_RECONBLKUSRMLSWRAPPER"
[EXEC] ORA-06512:

```

To resolve this issue, the following patches must be installed on Oracle Database 11g Release 1 (11.1.0.7.0):

- p7614692_111070
- p7000281_111070
- p8327137_111070
- p8617824_111070

Note: You can download all interim patches from the following URL:

<http://support.oracle.com>

Missing Critical Oracle Database 11g Release 2 Interim Patches

Running some SQL scripts might generate incorrect or inconsistent results on Oracle Database 11g Release 2 (11.2.0.2.0), which do not cause problems in earlier release of Oracle Database.

To resolve this issue, patch# 10259620 for Oracle Database 11g Release 2 must be installed.

Slow Reconciliation and Similar Traces in Error Log

When the SQL scripts having matching rules involving large volume, the entity tables are slow probably because of FULL table scans or unoptimized SQL plans in the database.

Reconciliation can be slow when the matching rule columns are not properly indexed or schema statistics is outdated. The slowness results in error logs similar to the following:

```
oracle.iam.platform.utils.SuperRuntimeException: java.sql.SQLException:
ORA-01013: user requested cancel of current operation
ORA-06512: at "XL_SP_RECONBLKROLEMATCH"
ORA-06512: at "OIM_SP_RECONBLKROLEMMSWRAPPER"
ORA-06512:

at weblogic.jms.client.JMSSession$UseForRunnable.run(JMSSession.java)
at weblogic.work.SelfTuningWorkManagerImpl$WorkAdapterImpl.run(SelfTuningWorkManagerImpl.java)
at weblogic.work.ExecuteThread.execute(ExecuteThread.java)
at weblogic.work.ExecuteThread.run(ExecuteThread.java)
Caused by: java.sql.SQLException: ORA-01013: user requested cancel of current operation
ORA-06512: at "XL_SP_RECONBLKROLEMATCH"
ORA-06512: at "OIM_SP_RECONBLKROLEMMSWRAPPER"
ORA-06512:
.
at oracle.jdbc.driver.SQLStateMapping.newSQLException(SQLStateMapping.java)
at oracle.jdbc.driver.DatabaseError.newSQLException(DatabaseError.java)
at oracle.jdbc.driver.DatabaseError.throwSQLException(DatabaseError.java)
at oracle.jdbc.driver.T4CTTIoer.processError(T4CTTIoer.java)
at oracle.jdbc.driver.T4CTTIoer.processError(T4CTTIoer.java)
```

To resolve this issue:

1. Verify that all the appropriate indexes are created over matching rule columns.
2. Verify that the database schema statistics are collected according to the guidelines.

See Also: "Connector for Reconciliation" in the *Oracle Fusion Middleware User's Guide for Oracle Identity Manager* for information about creating indexes for reconciliation and collecting database statistics

Developing Lookup Definitions, UDFs, and Remote Manager

This chapter describes how to use the Design Console to administer Oracle Identity Manager. It contains the following topics:

- [Overview](#)
- [Lookup Definition Form](#)
- [User Defined Field Definition Form](#)
- [Remote Manager Form](#)

15.1 Overview

The Design Console Administration folder provides system administrators with tools for managing Oracle Identity Manager administrative features. This folder contains the following forms:

- **Lookup Definition:** You use this form to create and manage lookup definitions. A lookup definition represents a lookup field and the values you can access from that lookup field.
- **User Defined Field Definition:** You use this form to create and manage user-defined fields. A user-defined field enables you to store additional information for the Design Console forms.
- **Remote Manager:** You use this form to display information about the servers that Oracle Identity Manager uses to communicate with third-party programs. These servers are known as remote managers.
- **Password Policies:** You use this form to set password restrictions for the users and view the rules and resource objects that are associated with a password policy. See "Managing Password Policies" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager* for detailed information about password policies.

Note: With this release, the System Configuration Form and the Task Scheduler Form are part of the Administrative and User Console. Refer to the *Oracle Identity Manager Administrator's Guide* for details.

15.2 Lookup Definition Form

A lookup definition represents one of the following:

- The name and description of a text field

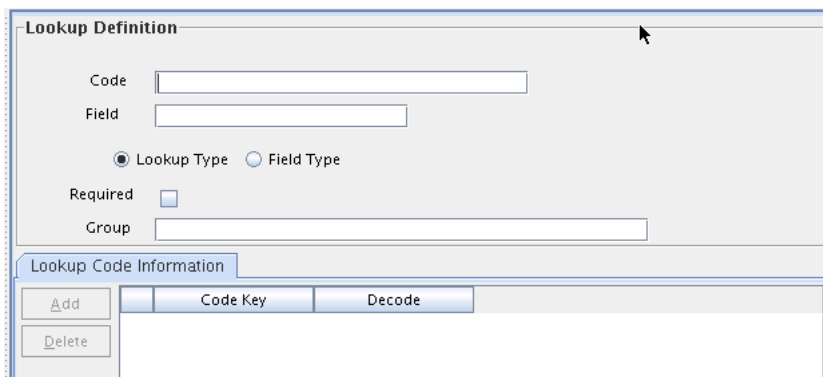
- A lookup field and the values that are accessible from that lookup field by double-clicking it
- A box, and the commands that can be selected from that box

These items, which contain information pertaining to the text field, lookup field, or box, are known as lookup values. Users can access lookup definitions from one of two locations:

- A form or tab that comes packaged with Oracle Identity Manager
- A user-created form or tab built by using the Form Designer form

The Lookup Definition form shown in [Figure 15–1](#) is in the Design Console Administration folder. You use this form to create and manage lookup definitions.

Figure 15–1 *Lookup Definition Form*



[Table 15–1](#) describes the data fields of the Lookup Definition form.

Table 15–1 *Fields of the Lookup Definition Form*

Field Name	Description
Code	The name of the lookup definition.
Field	The name of the table column of the form or tab from which the text field, lookup field, or box field will be accessible.
Lookup Type/Field Type	<p>These options designate if the lookup definition is to represent a text field, a lookup field, or a box.</p> <p>If you select the Field Type option, the lookup definition will represent a text field.</p> <p>If you select the Lookup Type option, the lookup definition is to represent either a lookup field or a box, along with the values that are to be accessible from that lookup field or box.</p> <p>Note: For forms or tabs that come packaged with Oracle Identity Manager, the lookup definition has already been set as either a lookup field <i>or</i> a box. This cannot be changed. However, you can add or modify the values that are accessible from the lookup field or box.</p> <p>For forms or tabs that are user defined, the user determines whether the lookup definition represents a lookup field or a box through the Additional Columns tab of the Form Designer form.</p> <p>For more information about specifying the data type of a lookup definition, see "Additional Columns Tab".</p>

Table 15–1 (Cont.) Fields of the Lookup Definition Form

Field Name	Description
Required	By selecting this check box, the lookup definition is designated as required. As a result, Oracle Identity Manager will not allow the contents of the corresponding form or tab to be saved to the database until the field or box, represented by the lookup definition, is supplied with data.
Group	The name of the Oracle Identity Manager or user-defined form on which the lookup definition is to be displayed.

The following sections describe how to create a lookup definition.

15.2.1 Creating a Lookup Definition

To create a lookup definition:

1. Open the Lookup Definition form.
2. In the **Code** field, enter the name of the lookup definition.
3. In the **Field** field, enter the name of the table column of the Oracle Identity Manager or user-created form or tab, from which the text field, lookup field, or box field will be accessible.
4. If the lookup definition is to represent a lookup field or box, select the **Lookup Type** option.

If the lookup definition is to represent a text field, select the **Field Type** option.

5. Optional. To save the contents of this form or tab only when the field or box represented by the lookup definition is supplied with data, select the **Required** check box. Otherwise, go to Step 6.
6. In the **Group** field, enter the name of the Oracle Identity Manager or user-defined form on which the lookup definition is displayed.

You must follow naming conventions for the text you enter into the **Code**, **Field**, and **Group** fields.

See Also: See "[Lookup Definition Form](#)" on page 15-1 for more information about naming conventions

7. Click **Save**.

The lookup definition is created. The associated text field, lookup field, or box will be displayed in the Oracle Identity Manager or user-defined form or tab you specified.

15.2.2 Lookup Code Information Tab

The Lookup Code Information tab is in the lower half of the Lookup Definition form. You use this tab to create and manage detailed information about the selected lookup definition. This information includes the names, descriptions, language codes, and country codes of a value pertaining to the lookup definition. These items are known as **lookup values**.

The following procedures show how to create, modify, and delete a lookup value.

15.2.2.1 Creating and Modifying a Lookup Value

To create or modify a lookup value:

Note: For internationalization purpose, you must provide both a language and country code for a lookup value.

When creating a new lookup definition, you must save it before adding lookup values to it.

1. Open the Lookup Definition form.

2. Access a lookup definition.

3. If you are creating a lookup value, click **Add**.

A blank row is displayed in the **Lookup Code Information** tab.

If you are modifying a lookup value, select the lookup value that you want to edit.

4. Add or edit the information in the **Code Key** field.

This field contains the name of the lookup value.

In addition, if the **Lookup Type** option is selected, this field also represents what is displayed in the lookup field or box once the user makes a selection.

5. Add or edit the information in the **Decode** field.

This field contains a description of the lookup value.

Note: The decode value is a humanly readable description of the field. The encode value is the actual code value that is used for provisioning. For example, decode value can be an LDAP group name, and encode value is the LDAP group GUID.

If the **Lookup Type** option is selected, this field also represents one of the following:

- The items that is displayed in a lookup window after the user double-clicks the corresponding lookup field
- The commands that are to be displayed in the associated box

6. Click **Save**.

The lookup value you created or modified now reflects the settings you have entered.

15.2.2.2 Deleting a Lookup Value

To delete a lookup value:

Caution: Deleting a lookup value might cause problems depending on what the lookup represents. For example, if a lookup value represents an entitlement and it is deleted, then it must be removed from various locations, such as any access policy with that entitlement or any user account having that entitlement granted. Therefore, Oracle recommends that you check all the possible effects before deleting a lookup value.

1. Open the Lookup Definition form.
2. Search for a lookup definition.
3. Select the lookup value that you want to remove.
4. Click **Delete**. The selected lookup value is deleted.

15.2.3 Configuring Challenge Questions for the User

You can configure challenge questions for the users by using the Lookup Definition Form. These challenge questions are prompted if the user forgets the password and tries to retrieve it. The user must enter the same answers provided while creating a password.

To configure challenge questions for the user:

1. Login to Oracle Identity Manager Design Console.
2. Navigate to **Administration, Lookup Definition**.
3. Search for the Lookup for challenge questions, that is, lookup Code = Lookup.WebClient.Questions.
4. In the Lookup Code Information tab, add questions by entering the appropriate values in the Code Key and Decode fields.
5. Click **Add**.
6. Add this key to the custom resource bundle.

15.3 User Defined Field Definition Form

You might want to augment the fields that Oracle Identity Manager provides by default. You can create new fields and add them to various Oracle Identity Manager forms. These fields are known as user-defined fields (UDFs). In other words, Oracle Identity Manager provides the administrator the capability to extend the schema of some Oracle Identity Manager tables. This is provided in the form of UDFs.

User-defined fields are displayed on the **User Defined Fields** tab of the form that is displayed in the **Form Name** data field. For example, [Figure 15–2](#) shows an **Access Code Number** user-defined field added to the **User Defined Fields** tab of the Organizations form.

The User Defined Field Definition form is displayed, as shown in [Figure 15–2](#), in the Design Console Administration folder. You use this form to create and manage user-defined fields for the organizations, resource objects, roles, and Form Designer forms.

[Table 15–2](#) describes the data fields of the User Defined Field Definition form.

Table 15–2 *Fields of the User Defined Field Definition Form*

Field Name	Description
Form Name	The name of the form that contains the user-defined fields. These fields are displayed in the User Defined Columns tab. Note: Because the user-defined fields for a user pertain to the user's profile information, they are displayed in the User Profile tab of the Users form.
Description	Additional information about the user-defined field.

Table 15–2 (Cont.) Fields of the User Defined Field Definition Form

Field Name	Description
Auto Pre-Population	<p>This check box designates if user-defined fields for a form that have prepopulated adapters attached to them will be populated by Oracle Identity Manager or a user.</p> <p>Select the Auto Pre-Population check box if these fields are populated by Oracle Identity Manager.</p> <p>Deselect this check box if these fields must be populated by a user by clicking the Pre-Populate button on the toolbar or by manually entering the data.</p> <p>Note: This setting does not control triggering of the pre-populate adapter. It only determines if the contents resulting from the execution of the adapter are displayed in the associated user-defined field or fields because of Oracle Identity Manager or a user.</p> <p>For more information about prepopulate adapters, see <i>Oracle Identity Manager Tools Reference</i>.</p> <p>Note: This check box is relevant only if you have created a user-defined field, and a prepopulate adapter is associated with that field.</p>

The following section describes how to select a target form for user-defined fields.

15.3.1 Selecting the Target Form for a User-Defined Field

To select the target form for a user-defined field:

1. Open the User Defined Field Definition form.
2. Double-click the **Form Name** lookup field.

From the Lookup window that is displayed, select the Oracle Identity Manager form (Resource Objects, Organizations, Roles, or Form Designer) that will display the user-defined field you will be creating.

3. Click **OK**.

The form to which you will be adding the user-defined field is selected.

15.3.2 Tabs on the User Defined Field Definition Form

After you start the User Defined Field Definition form and select a target form for the user-defined fields, the tabs of this form become functional.

The User Defined Field Definition form contains the following tabs:

- [User Defined Columns Tab](#)
- [Properties Tab](#)
- [Administrators Tab](#)

Each of these tabs is covered in greater detail in the sections that follow.

15.3.2.1 User Defined Columns Tab

You use this tab to do the following:

- Create a user-defined field.
- Set the variant type, length, and field type for the user-defined field.

- Specify the order in which the user-defined field is displayed on the **User Defined Fields** tab of the target form.

The field's order number determines the order in which a user-defined field is displayed on a form. In [Figure 15-2](#), the **Access Code Number** user-defined field has an order number of 1, so it is displayed first on the **User Defined Fields** tab of the Organizations form.

- Determine if the information that is associated with the user-defined field is encrypted when it is exchanged between the client and the server.
- Remove a user-defined field.

[Figure 15-2](#) shows the User Defined Columns tab of the User Defined Field Definition Form.

Figure 15-2 User Defined Columns Tab of the User Defined Field Definition Form

	Label	Variant Ty...	Len...	Column Name	Order	Field Type	Encrypted
1	Access Cod...	String	25	ACT_UDF_ACN	1	TextField	0

The following sections describe how to add and remove a user-defined field in an Oracle Identity Manager form.

Adding a User-Defined Field to an Oracle Identity Manager Form

To add a user-defined field:

1. Click **Add**.

The User Defined Fields dialog box is displayed, as shown in [Figure 15-3](#).

Figure 15-3 User Defined Fields Dialog Box

The following table describes the fields in the User Defined Fields dialog box.

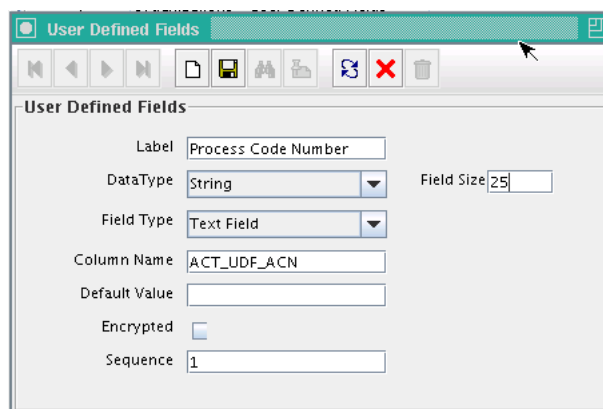
Table 15–3 Fields of the User Defined Fields Dialog Box

Field Name	Description
Label	<p>The label for the user-defined field. This label is displayed next to the user-defined field on the User Defined Fields tab of the target form.</p> <p>The maximum length for a label is 30 characters.</p>
Data Type	<p>From this box, select one of the following data types for the user-defined field:</p> <ul style="list-style-type: none"> ■ String. A user can enter a series of alphanumeric characters in this field. ■ Date. When a user double-clicks this field, the Date and Time dialog box is displayed. ■ Integer. A user can enter a number without a decimal point (for example, 3) in this user-defined field. ■ Boolean. A user can enter two values into this field: True (1) or False (0). ■ Double. A user can enter a double-precision floating-point number (or a double number) in this field.
Field Size	<p>The Field Size text field is enabled only for the String data type.</p> <p>In this field, enter the maximum amount of numbers or characters that a user can enter in the field. If the size is 4000 characters or less, then it is a varchar2 field. If the size is more than 4000, then it is CLOB.</p>
Field Type	<p>From this box, select one of the following field types for the user-defined field:</p> <ul style="list-style-type: none"> ■ Text Field. The field is displayed on the User Defined Fields tab of the target form as a text field. ■ Lookup Field. The field is displayed on the User Defined Fields tab of the target form as a lookup field. ■ Combo Box. The field is displayed on the User Defined Fields tab of the target form as a box. ■ Text Area. The field is displayed on the User Defined Fields tab of the target form as a text area. ■ Password Field. The field is displayed on the User Defined Fields tab of the target form as a text field. When you query for the entity and get the entity details, the password value, which is encrypted and displayed as a series of asterisks [*], is retrieved. ■ Check Box. The field is displayed on the User Defined Fields tab of the target form as a check box. ■ Date Field with Dialog. This field is displayed on the User Defined Fields tab of the target form as a lookup field. Once the user double-clicks this lookup field, a Date & Time window is displayed. Oracle Identity Manager will populate the data field with the date and time that the user selects from this window. <p>Note: The field types that are displayed in this box reflect the data type that is displayed in the Data Type box.</p>

Table 15–3 (Cont.) Fields of the User Defined Fields Dialog Box

Field Name	Description
Column Name	<p>The name of the user-defined field that is recognized by the database.</p> <p>Note: This name consists of a <code>TABLE_NAME_UDF_</code> prefix, followed by the label that is associated with the user-defined field.</p> <p>For example, if the Table Name field of the Organizations form is ACT, and the name for the data field is ACN, the name of the user-defined field, which the database recognizes, would be ACT_UDF_ACN.</p> <p>Note: The name in Column Name field cannot contain any spaces.</p>
Default Value	<p>This value is displayed in a user-defined field on the target form. Oracle recommends that you do not specify default values for passwords and encrypted fields.</p>
Encrypted	<p>This check box determines if the information that is displayed in the associated user-defined field is encrypted when it is exchanged between the client and the server.</p> <p>Select this check box to encrypt the information displayed in the user-defined field.</p> <p>Deselect this check box to not encrypt the information in the user-defined field.</p> <p>Note: Here, encrypted means that the field is encrypted in the database, but is displayed as clear text in the UI. A password field means that the field is encrypted in the database, and is displayed as asterisk characters (***) in the UI.</p>
Sequence	<p>This field represents the order in which the user-defined field is displayed on the form. For example, if a 2 is displayed in the Sequence field, it is displayed below the user-defined field with a sequence number of 1.</p> <p>Note: The same sequence number cannot be assigned to two user-defined fields.</p>

2. Set the parameters for the user-defined field you are adding to a form, as shown in [Figure 15–4](#).

Figure 15–4 User Defined Fields Dialog Box - Filled


The screenshot shows the 'User Defined Fields' dialog box with the following configuration:

- Label:** Process Code Number
- DataType:** String
- Field Size:** 25
- Field Type:** Text Field
- Column Name:** ACT_UDF_ACN
- Default Value:** (empty)
- Encrypted:**
- Sequence:** 1

In [Figure 15-4](#), the **Process Code Number** user-defined field is displayed first on the **User Defined Fields** tab of the Organizations form. The data type of this field is String, and a user can enter up to 25 characters into it.

3. From this window, click **Save**.
4. Click **Close**.

The user-defined field is displayed in the **User Defined Columns** tab. Once the target form is started, this user-defined field usually is displayed in the **User Defined Fields** tab of that form. Because the user-defined fields for a user pertain to the user's profile information, they are displayed in the **User Profile** tab of the **Users** form.

Removing a User-Defined Field from an Oracle Identity Manager Form

To remove a user-defined field:

1. Select the desired user-defined field.
2. Click **Delete**.

The user-defined field is removed.

15.3.2.2 Properties Tab

You use this tab to assign properties and property values to the data fields that are displayed on the **User Defined Fields** tabs of various Oracle Identity Manager forms.

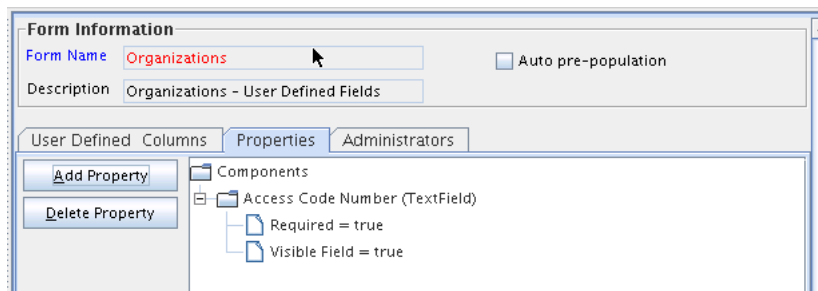
For this example, the **User Defined Fields** tab of the Requests form displays one data field: **Issue Tracking Item**. This data field contains the following properties:

- **Required**, which determines whether or not the data field must be populated for the **Requests** form to be saved. The default property value for the **Required** property is **false**.
- **Visible Field**, which determines whether or not the data field is displayed on the **Requests** form. The default property value for the **Visible Field** property is **true**.

Because the property values for the **Required** and **Visible Field** properties are **true** for this data field, once the **Requests** form is started, the **Issue Tracking Item** data field is displayed in the **User Defined Fields** tab. In addition, this field must be populated for the form to be saved.

[Figure 15-5](#) shows the Properties tab of the User Defined Field Definition form.

Figure 15-5 Properties Tab of the User Defined Field Definition Form



The following section describes how to add and remove a property and property value for a data field.

See Also: See "Form Designer Form" for more information about how to add a property and property value to a data field, or remove a property and property value from a data field

15.3.2.3 Administrators Tab

Figure 15–6 shows the Administrators tab of the User Defined Field Definition form.

Figure 15–6 Administrators Tab of the User Defined Field Definition Form

You use this tab to specify the roles that have administrative privileges over the current record of the User Defined Field Definition form. The **Write** and **Delete** check boxes on this form designate if these administrative roles can modify or delete information about the current user-defined field (UDF) definition.

15.4 Remote Manager Form

The Remote Manager is a lightweight network server that enables you to integrate with target systems whose APIs cannot communicate over a network, or that have network awareness but are not secure. The Remote Manager works as a server on the target system, and an Oracle Identity Manager server works as its client. The Oracle Identity Manager server sends a request for the Remote Manager to instantiate the target system APIs on the target system itself, and invokes methods on its behalf.

The Remote Manager form shown in Figure 15–7 is in the Design Console Administration folder. It displays the following:

- The names and IP addresses of the remote managers that communicate with Oracle Identity Manager
- Whether or not the remote manager is running
- Whether or not it represents IT resources that Oracle Identity Manager can use

Figure 15–7 Remote Manager Form

	Service	Url	Running	IT Resource
1	RManager	rmi://10.228.220.159:1	<input type="checkbox"/>	<input checked="" type="checkbox"/>

For this example, you can define only one remote manager that can communicate with Oracle Identity Manager: RManager.

Although this remote manager can handshake with Oracle Identity Manager, it is unavailable because the **Running** check box is deselected. Since the **IT Resource** check

box is selected, this remote manager represents an IT resource or resources that can be used by Oracle Identity Manager.

Part III

Identity Connector Framework

This part contains information regarding the Identity Connector Framework and how to use it to create an identity connector.

This part contains the following chapters:

- [Chapter 16, "Understanding the Identity Connector Framework"](#)
- [Chapter 17, "Developing Identity Connectors"](#)

Understanding the Identity Connector Framework

Identity connectors are components developed to link Oracle Identity Manager with external stores of applications, directories, and databases. This release of Oracle Identity Manager provides support for developing and building identity connectors by using the Identity Connector Framework (ICF). Using the ICF decouples Oracle Identity Manager from the other applications to which it connects. Therefore, you can build and test an identity connector before integrating it with Oracle Identity Manager. This chapter contains conceptual information and sample code in the following sections:

- [Introducing the ICF Architecture](#)
- [Using the ICF API](#)
- [Introducing the ICF SPI](#)
- [Extending an Identity Connector Bundle](#)
- [Using an Identity Connector Server](#)

Note: Earlier releases of Oracle Identity Manager have other options for building identity connectors. These options are still supported, but it is recommended that you build new identity connectors by using the ICF.

16.1 Introducing the ICF Architecture

Identity connectors allow Oracle Identity Manager to carry out user provisioning and reconciliation operations on target systems in the enterprise. ICF decouples any calling application, such as Oracle Identity Manager, from the implementation of the connector. ICF also decouples the implementation of the connector from the calling application. The same connector implementation can work with several different calling applications. [Figure 16–1](#) illustrates how this is accomplished by situating the ICF API and SPI between Oracle Identity Manager and the target system.

The API implementation always post-processes the results returned by the SPI Search operation. This double-checks the SPI implementation if the connector bundle does not implement all Filter types, or does not implement them properly for all attributes. If the implementation of Search in the SPI returns every object of the specified type, then the API implementation discards every object that does not match the specified Filter. Post-processing in the API implementation is expensive in terms of processing-time and network-bandwidth, and therefore, it is more efficient if each connector-bundle supports every type of filter (search predicate or logical operator)

that the target application can support natively. See the details for Filter Translator in "Common Classes" on page 16-18.

Figure 16-1 illustrates that the calling application sees only the ICF API. The ICF API dedicates a classloader to each connector bundle, so that the calling application is not exposed to the classes and libraries in the implementation of the connector-bundle (SPI). Bundle classloader also ensures isolation between the bundles as well as making any bundled library available to the connector bundle only, thereby avoiding conflicts between dependencies.

Figure 16-1 Identity Connector Framework Deployment

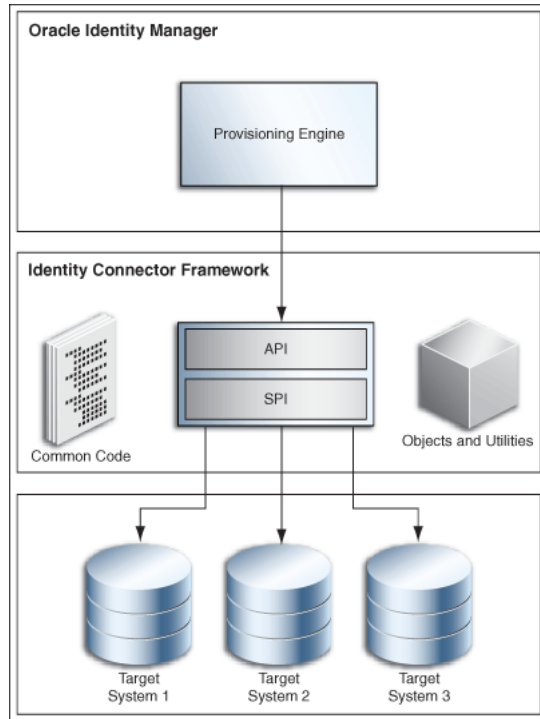
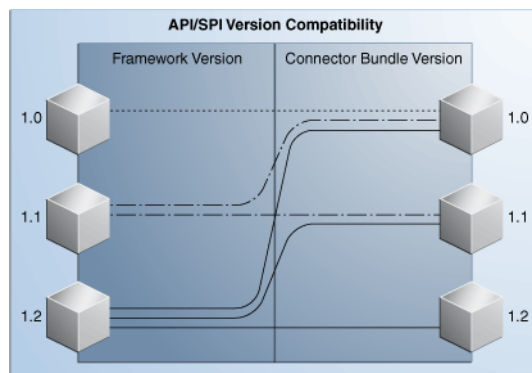


Figure 16-2 illustrates the backwards compatibility of the ICF. Newer bundles may be deployed without affecting existing ones. In addition, newer versions of the ICF are generally backward-compatible with existing bundles.

Figure 16-2 Compatibility Between the ICF and Connector Bundles



Identity connectors are stateless by design. An identity connector stores nothing. The calling application supplies to the connector the values for its configuration, including

the information required to connect to the target application. Because identity connectors are stateless, each bundle implementation are kept as simple as possible, and coupling the implementation with that of the calling application is also prevented.

16.2 Using the ICF API

The `org.identityconnectors.framework.api` package contains the ICF API. Oracle Identity Manager uses the API to call Connector implementations. The API provides a consistent view of any implemented Connector, regardless of the supported operations. The following sections explain these interfaces and classes.

- [The ConnectorInfoManagerFactory Class](#)
- [The ConnectorInfoManager Interface](#)
- [The ConnectorKey Class](#)
- [The ConnectorInfo Interface](#)
- [The APIConfiguration Interface](#)
- [The ConfigurationProperties Interface](#)
- [The ConnectorFacadeFactory Class](#)
- [The ConnectorFacade Interface](#)

16.2.1 The ConnectorInfoManagerFactory Class

The `ConnectorInfoManagerFactory` class allows Oracle Identity Manager to load Connector classes from a set of bundles. The static `getInstance` method returns an object of type `ConnectorInfoManagerFactory`. This object can then be used to get a reference to the `ConnectorInfoManager`. (See [Section 16.2.2, "The ConnectorInfoManager Interface"](#) for more information.) [Example 16–1](#) illustrates the `ConnectorInfoManagerFactory` implementation.

Example 16–1 ConnectorInfoManagerFactory Implementation

```
//create ConnectorInfoManagerFactory
ConnectorInfoManagerFactory cInfoManagerFactory =
    ConnectorInfoManagerFactory.getInstance();
```

16.2.2 The ConnectorInfoManager Interface

The `ConnectorInfoManager` interface maintains a list of `ConnectorInfo` instances. Each instance describes an identity connector. `ConnectorInfoManager` can be obtained by calling the `getLocalManager` method on the `ConnectorInfoManagerFactory`, and a list of bundle URLs is passed to it. `ConnectorInfoManager` can also be obtained by calling `getRemoteManager` method on the `ConnectorInfoManagerFactory`. The `getRemoteManager` method accepts an instance of `RemoteFrameworkConnectionInfo`, which is used for getting information about connectors deployed on Connector Server.

In [Example 16–2](#), `cInfoManagerFactory` is the instance of the `ConnectorInfoManagerFactory` and `bundleURL` is a list of bundle URLs that may point to directories consisting of JAR-ed or un-JAR-ed bundles.

Example 16–2 ConnectorInfoManager Implementation

```
//get the ConnectorInfoManager
```

```
ConnectorInfoManager cInfoManager =  
    cInfoManagerFactory.getLocalManager(bundleURL);
```

16.2.3 The ConnectorKey Class

A ConnectorKey uniquely identifies a Connector instance within an installation. The ConnectorKey class takes a bundleName (name of the Connector bundle), a bundleVersion (version of the Connector bundle) and a connectorName (name of the Connector bundle) as illustrated in [Example 16–3](#).

Example 16–3 ConnectorKey Implementation

```
//get the ConnectorKey reference  
ConnectorKey flatFileConnectorKey =  
    new ConnectorKey(bundleName, bundleVersion, connectorName);
```

16.2.4 The ConnectorInfo Interface

The ConnectorInfo interface contains information about a specific identity connector. It contains the display name, key and message details regarding the particular identity connector. [Example 16–4](#) illustrates how to implement the ConnectorInfo.

Example 16–4 ConnectorInfo Implementation

```
//get the ConnectorInfo  
ConnectorInfo info =  
    cInfoManager.findConnectorInfo(flatFileConnectorKey);
```

In the example, cInfoManager is the ConnectorInfoManager and flatFileConnectorKey is the identity connector key.

16.2.5 The APIConfiguration Interface

The APIConfiguration interface shows the configuration properties from both the SPI and the API sides. The getConfigurationProperties method returns a ConfigurationProperties instance based on the connector Configuration implementation, initialized to the defaults. Caller can then modify the properties, as required. [Example 16–5](#) illustrates this.

Example 16–5 APIConfiguration Definition

```
APIConfiguration apiConfig =  
    info.createDefaultAPIConfiguration();
```

16.2.6 The ConfigurationProperties Interface

The ConfigurationProperties interface encapsulates the SPI Configuration and uses reflection to identify the individual properties that are available for an application to manipulate. Set all of the identity connector's configuration properties using the setPropertyValue method as defined in [Example 16–6](#).

Example 16–6 setPropertyValue Method Signature

```
public void setPropertyValue  
    (java.lang.String name, java.lang.Object value)
```

[Example 16–7](#) illustrates an implementation of the `ConfigurationProperties` interface.

Example 16–7 ConfigurationProperties Implementation

```
//get the default APIConfiguration
ConfigurationProperties flatFileConfigProps =
    apiConfig.getConfigurationProperties();
```

16.2.7 The ConnectorFacadeFactory Class

The `ConnectorFacadeFactory` class allows an application to get a `Connector` instance and to manage a pool of `Connector` instances. [Example 16–8](#) illustrates the `ConnectorFacadeFactory` definition.

Example 16–8 ConnectorFacadeFactory Definition

```
//get a reference to ConnectorFacadeFactory
ConnectorFacadeFactory facadeFactory =
    ConnectorFacadeFactory.getInstance();
```

16.2.8 The ConnectorFacade Interface

The `ConnectorFacade` interface is used by the target system to invoke identity connector operations by representing a specific identity connector on the API side. [Example 16–9](#) illustrates the `ConnectorFacade` implementation.

Example 16–9 ConnectorFacade Implementation

```
//create a ConnectorFacade (nothing but a reference to Connector on SPI side)
ConnectorFacade connectorFacade = facadeFactory.newInstance(apiConfig)
```

16.3 Introducing the ICF SPI

Developers implement the ICF SPI to create identity connectors. The ICF SPI is made up of many interfaces but the developer need only implement those supported by the target system. SPI can again be classified into required, operation, and feature-based interfaces. Required interfaces must be implemented irrespective of the operations supported and they help to create the connector and maintain the connection with the target system, while operation interfaces help the connector to support various operations. Feature-based interfaces support certain features supported by the ICF.

The following sections have more information.

- [Implementing the Required Interfaces](#)
- [Implementing the Feature-based Interfaces](#)
- [Implementing the Operation Interfaces](#)
- [Common Classes](#)

16.3.1 Implementing the Required Interfaces

All identity connectors are required to provide an implementation of two interfaces. These two interfaces declare and initialize the identity connector with the target system. The following sections have more information.

- [org.identityconnectors.framework.spi.Connector](#)
- [org.identityconnectors.framework.spi.Configuration](#)

16.3.1.1 org.identityconnectors.framework.spi.Connector

This is the main interface to declare an identity connector. Many connectors create the connection to the target system when the connection is required, removing the connection when finished with it, and disposing of any resources it has used. The interface provides the init and dispose life cycle methods for this purpose.

Note: Connector implementations must be annotated with type `org.identityconnectors.framework.spi.ConnectorClass` by providing the `configurationClass` and `displayNameKey` information. The `displayNameKey` must be a key defined in the `Messages.properties` file.

Every connector implementation must be annotated with `@ConnectorClass`. This is required because the ICF would scan all top level `.class` files in the connector bundle looking for classes that have the `@ConnectorClass` annotation, therefore, autodiscovering connectors that are defined in the bundle. This annotation requires the following elements:

- **configurationClass:** This is the configuration class for this connector. This class has all the information about the target that can be used by the connector to connect and perform various provisioning and reconciliation operations. See section "[org.identityconnectors.framework.spi.Configuration](#)" on page 16-8 for more information on how to implement the configuration class.
- **displayNameKey:** Display name key that must be present in the message catalog.

[Example 16–10](#) is a sample connector implementation.

Example 16–10 Flat File Connector Implementation

```
/**
 * Flat file connector implementation. This connector supports create,
 * delete, search and update operations.
 */
@ConnectorClass
(configurationClass=FlatFileConfigurationImpl.class,
 displayNameKey="FLAT_FILE_CONNECTOR")
public class FlatFileConnector implements Connector,
    CreateOp, DeleteOp, SearchOp<Map<String, String>>, UpdateOp{
```

In [Example 16–10](#):

- **CreateOp:** Helps the connector to create an entity on the target system
- **DeleteOp:** Helps the connector to delete an entity on the target system
- **SearchOp:** Helps the connector to search an entity on the target system
- **UpdateOp:** Helps the connector to update an existing entity on the target system

See "[Implementing the Operation Interfaces](#)" on page 16-12 for more information.

The following sections contain information and sample code that illustrates how you might implement the Connector methods. For complete code regarding a Connector implementation, see "[Developing a Flat File Connector](#)" on page 17-1.

- [Implementing the init Method](#)
- [Implementing the dispose Method](#)
- [Implementing the getConfiguration Method](#)

16.3.1.1.1 Implementing the init Method

The init method initializes the connector. The connector initializes itself with the configuration instance as provided with the annotation `@ConnectorClass`. The init method takes a Configuration object as an argument. The Configuration object has all the information required by the Connector to connect to the target system.

[Example 16–11](#) illustrates how to implement the init method of interfaces in JDK 1.6.

Note: In this document, all code samples use the methods implementing interfaces in JDK 1.6.

Example 16–11 *init Method Implementation*

```
@Override
public void init(Configuration config) {
    this.flatFileConfig = (FlatFileConfiguration) config;

    FlatFileIOFactory flatFileIOFactory =
        FlatFileIOFactory.getInstance(flatFileConfig);
    this.flatFileMetadata = flatFileIOFactory.getMetadataInstance();
    this.flatFileParser = flatFileIOFactory.getFileParserInstance();
    this.flatFileWriter = flatFileIOFactory.getFileWriterInstance();
    log.ok("Initialization done");
}
```

Note: FlatFileIOFactory, FlatFileMetadata, FlatFileParser and FlatFileWriter are supporting classes and are not part of the ICF. An implementation of these classes is illustrated in "[Developing a Flat File Connector](#)" on page 17-1.

The init method implementation shown in [Example 16–11](#) does the following:

- Stores the configuration information of the target system. This can be used later while performing an operation.
- Initializes all the supporting classes it uses while performing any provisioning and reconciliation operations.

16.3.1.1.2 Implementing the dispose Method

The dispose method disposes of any resources held by this Connector instance. Once the method is called, the Connector instance is discarded and can not be used.

[Example 16–12](#) illustrates how to implement the dispose method.

Example 16–12 *dispose Method Implementation*

```
/**
 * Disposes any resource used by the connector.
 */
@Override
public void dispose() {
```

```
//close any open FileReader or FileWriter instances.  
  
//close connection with the target  
  
//close connection if any with database  
}
```

16.3.1.1.3 Implementing the getConfiguration Method

The getConfiguration method returns the Configuration instance passed to the Connector when the init method was used. [Example 16–13](#) illustrates how to implement the getConfiguration method.

Example 16–13 getConfiguration Method Implementation

```
/**  
 * returns the Configuration of this connector  
 */  
@Override  
public Configuration getConfiguration() {  
    return this.flatFileConfig;  
}
```

Note: Sometimes, components must be able to access the Configuration instance after initialization. This is supported by the accessor method getConfiguration().

16.3.1.2 org.identityconnectors.framework.spi.Configuration

The implementation of this interface encapsulates the configuration of a connector. Configuration implementation includes all the necessary information of the target system, which is used by the Connector implementation to connect to the target system and perform various reconciliation and provisioning operations. The implementation should have a default Constructor with setters and getters defined for its properties. Every property declared may not be required but if a property is required, then it should be marked required using the annotation `org.identityconnectors.framework.spi.ConfigurationProperty`. Configuration implementation is a Java bean and all the instance variables (mandatory or not) do have default values. For example, a string `userName` is used to connect to the target system and this is a mandatory attribute. This has a default value of null. When `userName` is a mandatory attribute, ICF expects a value to be provided by Oracle Identity Manager. In other words, Oracle Identity Manager cannot miss out this parameter. If missed, then the connector throws `ConfigurationException`.

The implementation should check that all required properties are available and validated before passing itself to the Connector. The interface provides a `validate` method for this purpose. For example, there are three mandatory configuration parameters, such as the IP address of the target, the username to connect to the target, and the password for the user. The `validate` method implementation can check for non-NULL values and valid IP address by using regex.

Note: ICF also provides a convenient base class `org.identityconnectors.framework.spi.AbstractConfiguration` for configuration objects to extend.

Example 16–14 Configuration Implementation

```
/**
 * Configuration implementation for the flat file connector.
 */
public class FlatFileConfigurationImpl extends AbstractConfiguration{
```

The following sections contain information and sample code that illustrates how you might implement the Configuration methods.

The Configuration implementation must provide implementation for the following methods:

- [The validate\(\) Method](#)
- [The setConnectorMessages\(\) Method](#)
- [The getConnectorMessages\(\) Method](#)

16.3.1.2.1 The validate() Method

The validate method checks that the values of all required properties are set. It also validates that all values of configuration properties are valid. In other words, it validates that all values of the configuration properties are in the expected range and have the expected format. If the configuration is not valid, then the implementations generate the most specific RuntimeException available. When no specific exception is available, the implementations can throw ConfigurationException. [Example 16–15](#) illustrates how to implement the validate method.

Example 16–15 validate Method Implementation

```
@Override
public void validate() {
    // Validate if file exists and is usable
    boolean validFile = (this.storeFile.exists() &&
        this.storeFile.canRead() &&
        this.storeFile.canWrite() &&
        this.storeFile.isFile());
    if (!validFile)
        throw new ConfigurationException("User store file not valid");
    FlatFileIOFactory.getInstance(this);
}
```

Here, if the target flat file provided is valid or not is checked, such as is a file, is writeable, is readable. If not valid, then an exception is generated.

Implementations of the validate method should NOT connect to the target system to validate the properties.

Note: This implementation depends on an instance variable (private File storeFile) and a supporting class (FlatFileIOFactory). A complete implementation is illustrated in "[Developing a Flat File Connector](#)" on page 17-1.

16.3.1.2.2 The setConnectorMessages() Method

The setConnectorMessages method sets the org.identityconnectors.framework.common.objects.ConnectorMessages message catalog instance, allowing the Connector to localize messages. [Example 16–16](#) illustrates the setConnectorMessages method definition.

Example 16–16 setConnectorMessages Method Definition

```
public final void setConnectorMessages(ConnectorMessages messages) {
    _connectorMessages = messages;
}
```

16.3.1.2.3 The getConnectorMessages() Method

The `getConnectorMessages` method returns the `ConnectorMessages` set by the `setConnectorMessages` method. [Example 16–17](#) illustrates the `getConnectorMessages` method definition.

Example 16–17 getConnectorMessages Method Definition

```
public final ConnectorMessages getConnectorMessages() {
    return _connectorMessages;
}
```

16.3.2 Implementing the Feature-based Interfaces

The following sections contain information on the interfaces used to enable identity connector pooling and attribute normalizing.

- [org.identityconnectors.framework.spi.PoolableConnector](#)
- [org.identityconnectors.framework.spi.AttributeNormalizer](#)

16.3.2.1 org.identityconnectors.framework.spi.PoolableConnector

Connection pooling by ICF is a feature provided by the ICF in which the framework maintains a pool of connector instances and uses them while performing provisioning and reconciliation operations. Connectors can make use of pooling by implementing the `PoolableConnector` interface instead of plain `Connector` interface. To make use of this feature, implement the `PoolableConnector` interface. If you implement the `Connector` interface, then ICF creates a new connector instance for every operation, creates a new connection with the target, completes the provisioning/reconciliation operation, removes the connection with the target system, and finally disposes this connector instance. Therefore, the advantages of implementing `PoolableConnector` is that a pool of configurable connector instances are maintained and are reused for many operations.

Some of configurable options are:

- Maximum connector objects in the pool that are idle and active (`_maxObjects`)
- Maximum connector objects that are idle (`_maxIdle`)
- Max time to wait if the pool is waiting for a free object to become available before failing (`_maxWait`)
- Minimum time to wait before evicting an idle object (`_minEvictableIdleTimeMillis`)
- Minimum number of idle objects (`_minIdle`)

These values must be set by connector API developer, and if not provided, then the following default values are used:

- `_maxObjects` = 10
- `_maxIdle` = 10

- `_maxWait = 150 * 1000 ms`
- `_minEvictableIdleTimeMillis = 120 * 1000 ms`
- `_minIdle = 1`

The `PoolableConnector` interface extends the `Connector` interface. It is implemented to enable identity connector pooling that ICF provides. ICF must make sure that the `Connector` instance is alive before being used. For this purpose, the interface provides a `checkAlive` method. [Example 16–18](#) is a sample flat file `PoolableConnector` implementation.

Example 16–18 Flat File Poolable Connector Implementation

```
/**
 * Flat file connector implementation. This is a poolable connector
 * which supports create, delete, search and update operations.
 */
@ConnectorClass
(configurationClass=FlatFileConfigurationImpl.class,
 displayNameKey="FLAT_FILE_CONNECTOR")
public class FlatFileConnector implements PoolableConnector,
CreateOp, DeleteOp, SearchOp<Map<String, String>>, UpdateOp{
```

To implement the `PoolableConnector` interface, provide an implementation of the `checkAlive` method along with all the methods discussed in [Section 16.3.1.1](#), "[org.identityconnectors.framework.spi.Connector](#)." The `checkAlive` method determines if the `Connector` instance is alive and can be used for operations on the target system. `checkAlive` can be called often thus the developer should make sure the implementation is fast. The method should throw a specific `RuntimeException` (if available) when the `Connector` is no longer alive. [Example 16–19](#) illustrates how to implement the `checkAlive` method.

Example 16–19 checkAlive Method Implementation

```
/**
 * Checks if this connector is alive, if not throws a RuntimeException
 */
@Override
public void checkAlive() {
//check if the connector is still connected to target
}
```

16.3.2.2 org.identityconnectors.framework.spi.AttributeNormalizer

This interface must be implemented by a `Connector` that needs to normalize any attributes passed to it. A normalizer converts values to a standard form for the purpose of display, consumption, or comparison. For example, a normalizer might convert text values to a specific case, trim whitespace, or order the elements of a DN in a specific way.

The interface defines a `normalizeAttribute` method for this purpose. This method takes an `ObjectClass` and an `Attribute` to be normalized as arguments and returns the normalized `Attribute`. Attribute normalization is applied during many operations including:

- Filters that are passed to `SearchOp`
- Results returned from `SearchOp`

- Results returned from SyncOp
- Attributes passed to UpdateAttributeValuesOp
- Uids returned from UpdateAttributeValuesOp
- Attributes passed to UpdateOp
- Uids returned from UpdateOp
- Attributes passed to CreateOp
- Uids returned from CreateOp
- Uids passed to DeleteOp

[Example 16–20](#) illustrates the `normalizeAttribute` method definition.

Example 16–20 `normalizeAttribute` Method Definition

```
public Attribute normalizeAttribute (ObjectClass oClass, Attribute attribute) {
    if (attribute instanceof Uid) {
        return new Uid(LdapUtil.createUniformUid((String)newValues.get(0),
            configuration.getSuffix()));
    }
}
```

16.3.3 Implementing the Operation Interfaces

Each operation interface defines an action that the Connector may perform on a target system, if supported by it. The operation interfaces belong to the `org.identityconnectors.framework.spi.operations` package. The names of these operation interfaces are listed below, but subsequent sections elaborate on each interface:

- `AuthenticateOp`
- `CreateOp`
- `DeleteOp`
- `ResolveUsernameOp`
- `SchemaOp`
- `ScriptOnConnectorOp`
- `ScriptOnResourceOp`
- `SearchOp<T>SyncOp`
- `TestOp`
- `UpdateAttributeValuesOp`
- `UpdateOp`

The following sections contain more information on some of these operations.

- [Implementing the SchemaOp Interface](#)
- [Implementing the CreateOp Interface](#)
- [Implementing the DeleteOp Interface](#)
- [Implementing the SearchOp Interface](#)
- [Implementing the UpdateOp Interface](#)

16.3.3.1 Implementing the SchemaOp Interface

The SchemaOp interface is implemented to allow the connector to describe the objects it can handle on the target system. The schema that a connector returns describes the object-classes that it exposes for management. Each object-class has a name, a description, and a set of attribute definitions. Each attribute definition has a name, a syntax, and certain flags that describe its properties, such as multi-valued, single-valued, readable, or writeable.

The schema that a connector returns describes the attributes of each type of object that the connector exposes. Sometimes, this requires translation from an internal representation to this Schema format. In other instances, the Schema presents as an attribute; something that is natively available only via calls to the target API. Irrespective of how the SPI implementation accomplishes the mapping between the native representation and the corresponding ConnectorObject, the Schema provides the metadata that describes what a client can expect to find in a ConnectorObject of each type, which is objectClass.

To implement this interface, provide an implementation for the schema method as defined in [Example 16–21](#).

Example 16–21 *schema Method Signature*

```
public Schema schema
```

The implementation should return the schema containing the types of objects that this identity connector supports.

Example 16–22 *schema Method Implementation*

```
@Override
public Schema schema() {
    SchemaBuilder flatFileSchemaBldr = new SchemaBuilder(this.getClass());
    Set<AttributeInfo> attrInfos = new HashSet<AttributeInfo>();
    for (String fieldName : flatFileMetadata.getOrderedTextFieldNames()) {
        AttributeInfoBuilder attrBuilder = new AttributeInfoBuilder();
        attrBuilder.setName(fieldName);
        attrBuilder.setCreateable(true);
        attrBuilder.setUpdateable(true);
        attrInfos.add(attrBuilder.build());
    }

    // Supported class and attributes
    flatFileSchemaBldr.defineObjectClass
        (ObjectClass.ACCOUNT.getDisplayNameKey(), attrInfos);
    return flatFileSchemaBldr.build();
}
```

Note: The Uid should not appear in the returned schema.

16.3.3.2 Implementing the CreateOp Interface

The CreateOp interface is implemented to enable creating objects on the target system. To implement this interface, provide an implementation of the create() method, as shown in [Example 16–23](#).

Example 16–23 create Method Signature

```
public Uid create
    (ObjectClass objectClass, Set<Attribute> attributes,
     OperationOptions options)
```

This method takes an `ObjectClass` (for example, `account` or `group`), a set object attributes, and operation options. The implementation creates an object on the target system by using passed object attributes and object type defined by `ObjectClass`. The `ObjectClass` argument specifies the class of object to create. The class of object to be created is one of the inputs to the create operation. `ObjectClass` is the first argument to the `create()` method, as shown in [Example 16–24](#).

Example 16–24 create Method Implementation

```
@Override
    public Uid create(ObjectClass arg0, Set<Attribute> attrs,
        OperationOptions ops) {

        System.out.println("Creating user account " + attrs);
        assertUserObjectClass(arg0);
        try {
            FlatFileUserAccount accountRecord = new FlatFileUserAccount(attrs);
            // Assert uid is there
            assertUidPresence(accountRecord);

            // Create the user
            this.flatFileWriter.addAccount(accountRecord);

            // Return uid
            String uniqueAttrField = this.flatFileConfig
                .getUniqueAttributeName();
            String uniqueAttrVal = accountRecord
                .getAttributeValue(uniqueAttrField);
            System.out.println("User " + uniqueAttrVal + " created");

            return new Uid(uniqueAttrVal);
        } catch (Exception ex) {

            // If account exists
            if (ex.getMessage().contains("exists"))
                throw new AlreadyExistsException(ex);

            // For all other causes
            System.out.println("Error in create " + ex.getMessage());
            throw ConnectorException.wrap(ex);
        }
    }
}
```

If the operation is successful, `Uid` instance representing object identifier on the target system is supposed to be created and returned. The caller can then use the `Uid` to refer to the created object.

16.3.3.3 Implementing the DeleteOp Interface

The `DeleteOp` interface is implemented to enable deleting objects from the target system. To implement this interface, provide an implementation for the `delete` method as defined in [Example 16–25](#).

Example 16–25 delete Method Signature

```
public void delete
    (ObjectClass objectClass, Uid uid, OperationOptions options)
```

This method takes an `ObjectClass` (for example, `account` or `group`), the `Uid` of the object being deleted from the target system, and operation options. The implementation deletes the object identified by the provided `Uid` from the target system. If the object does not exist on the target system, then an `org.identityconnectors.framework.common.exceptions.UnknownUidException` is generated. [Example 16–26](#) illustrates how to implement the `delete` method.

Example 16–26 delete Method Implementation

```
@Override
public void delete(ObjectClass arg0, Uid arg1, OperationOptions arg2) {
    final String uidVal = arg1.getUidValue();
    this.flatFileWriter.deleteAccount(uidVal);
    log.ok("Account {0} deleted", uidVal);
}
```

Note: If the `delete` operation fails, then ICF generates subclasses of `RuntimeException`. See *Oracle Fusion Middleware Java API Reference for Identity Connector Framework* for details.

16.3.3.4 Implementing the SearchOp Interface

The `SearchOp` interface is implemented to enable searching objects on the target system. Here, the search operation consists of:

- Creation of a native filter to implement search conditions that are specified generically.
- Executing the actual query.

Implementing these methods in the SPI allows the API to support search. The API performs (by post-processing the result) any filtering that the connector does not perform, for example, by translating any specified filter conditions into native search conditions.

To implement this interface, provide an implementation for the `createFilterTranslator` and `executeQuery` methods as documented in the following sections.

- [Implementing the createFilterTranslator Method](#)
- [Implementing the executeQuery Method](#)

16.3.3.4.1 Implementing the createFilterTranslator Method

The `createFilterTranslator` method returns an instance of implementation of `FilterTranslator`, which converts the ICF `Filter` object passed to it from the API side into a native query. Following the conversion, ICF passes the query to the `executeQuery` method. [Example 16–27](#) illustrates the `createFilterTranslator` method definition.

Example 16–27 createFilterTranslator Method Signature

```
public FilterTranslator createFilterTranslator
    (ObjectClass oClass, OperationsOptions options)
```

Note: The return value should not be null.

[Example 16–28](#) illustrates an implementation of the `createFilterTranslator` method.

Example 16–28 `createFilterTranslator` Method Implementation

```
@Override
public FilterTranslator<Map<String, String>> createFilterTranslator
    (ObjectClass arg0, OperationOptions arg1) {
    return new ContainsAllValuesImpl() {
    };
}
```

This example supports only a single type of search predicate, which is `ContainsAllValues`. See ["Implementation of AbstractFilterTranslator<T>"](#) on page 17-8 for an example of an implementation of `ContainsAllValuesImpl`. The implementation of `ContainsAllValues` translates into native form a condition of the form: Attribute A contains all of the values V(1), V(2) ... V(N).

For information on the `org.identityconnectors.framework.common.objects.filter.FilterTranslator`, see ["Common Classes"](#) on page 16-18.

16.3.3.4.2 Implementing the `executeQuery` Method

The `executeQuery` method is called for every query produced by the `FilterTranslator` implementation (as documented in ["Implementing the createFilterTranslator Method"](#) on page 16-15). It takes an `ObjectClass` (for example, `account` or `group`), the query, an instance of `ResultsHandler` used as a callback to handle found objects, and operation options, as illustrated in [Example 16–29](#).

Example 16–29 `executeQuery` Method Signature

```
public void executeQuery
    (ObjectClass oClass, T query,
    ResultsHandler handler, OperationOptions options)
```

The implementation of the `executeQuery` method searches for the target objects by using the passed query, creates instances of `ConnectorObject` for each target object found, and uses `ResultsHandler` to handle `ConnectorObjects`. `ConnectorObject` is ICF representation of target resource object. It contains information such as `ObjectClass`, `Uid`, `Name`, and `Set of Attributes`. `ConnectorObject` is central to search. `executeQuery` streams `ConnectorObjects` into the `ResultsHandler`, and therefore, to the client.

[Example 16–30](#) illustrates how to implement the `executeQuery` method.

Example 16–30 `executeQuery` Method Implementation

```
@Override
    public void executeQuery(ObjectClass objectClass,
        Map<String, String> matchSet, ResultsHandler resultHandler,
        OperationOptions ops) {

    // searches the flat file for accounts which fulfil the condition 'matchSet'
    // created by FilterTranslator
    Iterator<FlatFileUserAccount> userAccountIterator = this.flatFileParser
        .getAccountIterator(matchSet);
```



```

boolean handleMore = true;
while (userAccountIterator.hasNext() && handleMore) {
    FlatFileUserAccount userAcc = userAccountIterator.next();
    ConnectorObject userAccObject = convertToConnectorObject(userAcc);
    // Let the client handle the result by doing callback
    handleMore = resultHandler.handle(userAccObject);
}
while (userAccountIterator.hasNext()) {
    FlatFileUserAccount userAcc = userAccountIterator.next();
    ConnectorObject userAccObject = convertToConnectorObject(userAcc);
    if (!resultHandler.handle(userAccObject)) {
        System.out.println("Not able to handle " + userAcc);
        break;
    }
}
}
}

```

16.3.3.5 Implementing the UpdateOp Interface

The UpdateOp interface is implemented to enable updating objects on the target system. To implement this interface, provide an implementation of the update method as defined in [Example 16–31](#).

Example 16–31 update Method Signature

```

public Uid update(ObjectClass oClass, Uid uid,
    Set<Attribute> attributes, OperationOptions options)

```

This method takes an ObjectClass (for example, account or group), Uid of the object being updated, a set of object attributes being updated, and operation options. The implementation updates the object on the target system identified by the Uid with the new values of attributes. If the object identified by the Uid does not exist on the target system, then an UnknowUidException is generated. [Example 16–32](#) illustrates how to implement the update method.

Example 16–32 update Method Implementation

```

@Override
public Uid update(ObjectClass arg0, Uid arg1,
    Set<Attribute> arg2, OperationOptions arg3) {
    String accountIdentifier = arg1.getUidValue();
    // Fetch the account
    FlatFileUserAccount accountToBeUpdated = this.flatFileParser
        .getAccount(accountIdentifier);

    // Update
    accountToBeUpdated.updateAttributes(arg2);
    this.flatFileWriter
        .modifyAccount(accountIdentifier, accountToBeUpdated);
    log.ok("Account {0} updated", accountIdentifier);

    // Return new uid
    String newAccountIdentifier = accountToBeUpdated
        .getAttributeValue
            (this.flatFileConfig.getUniqueAttributeName());
    return new Uid(newAccountIdentifier);
}

```

16.3.4 Common Classes

There are many ICF classes mentioned in the previous sections. The most important classes are:

- `org.identityconnectors.framework.common.objects.Attribute`

An `Attribute` is a named collection of values within a target system object. A target system object may have many attributes and each may have many values. In its simplest form, an `Attribute` can be considered a name-value pair of a target system object. Empty and null values are supported. The developer should use `org.identityconnectors.framework.common.objects.AttributeBuilder` to construct `Attribute` instances.

Note: All attributes are syntactically multivalued in this model. A particular attribute being singlevalued is only a semantic restriction.

- `org.identityconnectors.framework.common.objects.Uid`

A single-valued `Attribute` (`Uid` is a subclass of `Attribute`) that represents the unique identifier of an object on the target resource. Ideally, it should be immutable.

Note: A singlevalued attribute is particularly relevant to `UID` being a unique identifier.

- `org.identityconnectors.framework.common.objects.ObjectClass`

An `ObjectClass` defines the type of the object on the target system. `Account`, `group`, or `organization` are examples of such types. ICF defines predefined `ObjectClasses` for `account` (`ObjectClass.ACCOUNT`) and `group` (`ObjectClass.GROUP`).

- `org.identityconnectors.framework.common.objects.ConnectorObject`

A `ConnectorObject` represents an object (for example, an `account` or `group`) on the target system. The developer must use `org.identityconnectors.framework.common.objects.ConnectorObjectBuilder` to construct a `ConnectorObject`.

- `org.identityconnectors.common.security.GuardedString`

A guarded string is a secure `String` implementation which solves the problem of storing passwords in memory in a plain `String` format. Passwords are stored as `Bytes` in an encrypted format. The encryption key will be randomly generated.

- `org.identityconnectors.framework.common.objects.filter.FilterTranslator`

A `FilterTranslator` object is responsible for converting all the filters specified on the API side of the ICF into native queries during a search operation. ICF Filters support both search predicates and logical operators:

- Search predicates match objects based on the values of a specified attribute. For example, an `EqualsFilter` returns true when at least one value of an attribute is equal to a specified value.
- Logical operators `AND` and `OR` join search predicates to build complex expressions. For example, an expression of the form "A AND B" is true only if both A and B are true. An expression of the form "A OR B" is true if at least one of A or B is true.

The ICF provides the `AbstractFilterTranslator<T>` base class to make search implementation easier. A `FilterTranslator` sub class should override the following whenever possible.

- `createAndExpression(T, T)`
- `createOrExpression(T, T)`
- `createContainsExpression(ContainsFilter, boolean)`
- `createEndsWithExpression(EndsWithFilter, boolean)`
- `createEqualsExpression(EqualsFilter, boolean)`
- `createGreaterThanExpression(GreaterThanFilter, boolean)`
- `createGreaterThanOrEqualExpression(GreaterThanOrEqualFilter, boolean)`
- `createStartsWithExpression(StartsWithFilter, boolean)`
- `createContainsAllValuesExpression(ContainsAllValuesFilter, boolean)`

For more information see [Section 16.3.3.4, "Implementing the SearchOp Interface."](#)

- `org.identityconnectors.framework.common.objects.ResultsHandler`

This is a callback interface for operations returning one or more results. The sub class should provide an implementation to the `handle` method whereas the caller can decide what to do with the results. Currently, this is used only by the `SearchOp` interface. For more information, see [Section 16.3.3.4, "Implementing the SearchOp Interface."](#)

16.4 Extending an Identity Connector Bundle

An identity connector bundle is the specific implementation for a particular target system. The bundle is a Java archive (JAR) that contains all the files required by the identity connector to connect to the target system and perform operations. It also has special attributes (defined in the MANIFEST file) that are recognized by the ICF. These are:

- **ConnectorBundle-FrameworkVersion** is the minimum version of the ICF required for this identity connector bundle to work. Newer ICF versions will be backwards compatible.
- **ConnectorBundle-Name** is the unique name for this identity connector bundle; it is generally the package name.
- **ConnectorBundle-Version** is the version of this bundle. Within a given deployment of Oracle Identity Manager, the `ConnectorBundle-Name` and `ConnectorBundle-Version` combination should be unique.

You extend an identity connector bundle, for example, to reuse common code. The `AbstractDatabaseConnector` is a good example, because different types of connectors can reuse the same basic logic that accesses database tables using JDBC. A connector for database tables might share this common code with a connector for Oracle Database users, a connector for IBM DB2 database users, and a connector for MySQL users.

A given Connector can be extended by adding the extended bundle to the `/lib` directory of a new bundle and creating a new class that subclasses the target class. This can be illustrated with the `AbstractDatabaseConnector` bundle. The common logic would be in a common bundle as follows:

Note: You do not extend the original bundle. Instead, you extend the connector by embedding the original bundle in a new bundle that wraps the original bundle.

- META-INF/MANIFEST.MF
 - ConnectorBundle-FrameworkVersion: 1.0
 - ConnectorBundle-Name: org.identityconnectors.database.common
 - ConnectorBundle-Version: 1.0
- org.identityconnectors/database/common/AbstractDatabaseConnector.class

Note: This identity connector would not have a `@ConnectorClass` annotation.

- org/identityconnectors/database/common/* (other common source files)
- lib/

There would be as many database (resource) specific bundles as needed. For example:

- META-INF/MANIFEST.MF
 - ConnectorBundle-FrameworkVersion: 1.0
 - ConnectorBundle-Name: org.identityconnectors.database.mysql
 - ConnectorBundle-Version: 1.0
- org/identityconnectors/database/mysql/MySQLConnector.class (subclass of AbstractDatabaseConnector)

Note: This identity connector would have a `@ConnectorClass` annotation.

- org/identityconnectors/database/mysql/* (other MySQL source files)
- lib/org.identityconnectors.database.common-1.0.jar (parent bundle described above)
- lib/* (specific database drivers and libraries as needed)

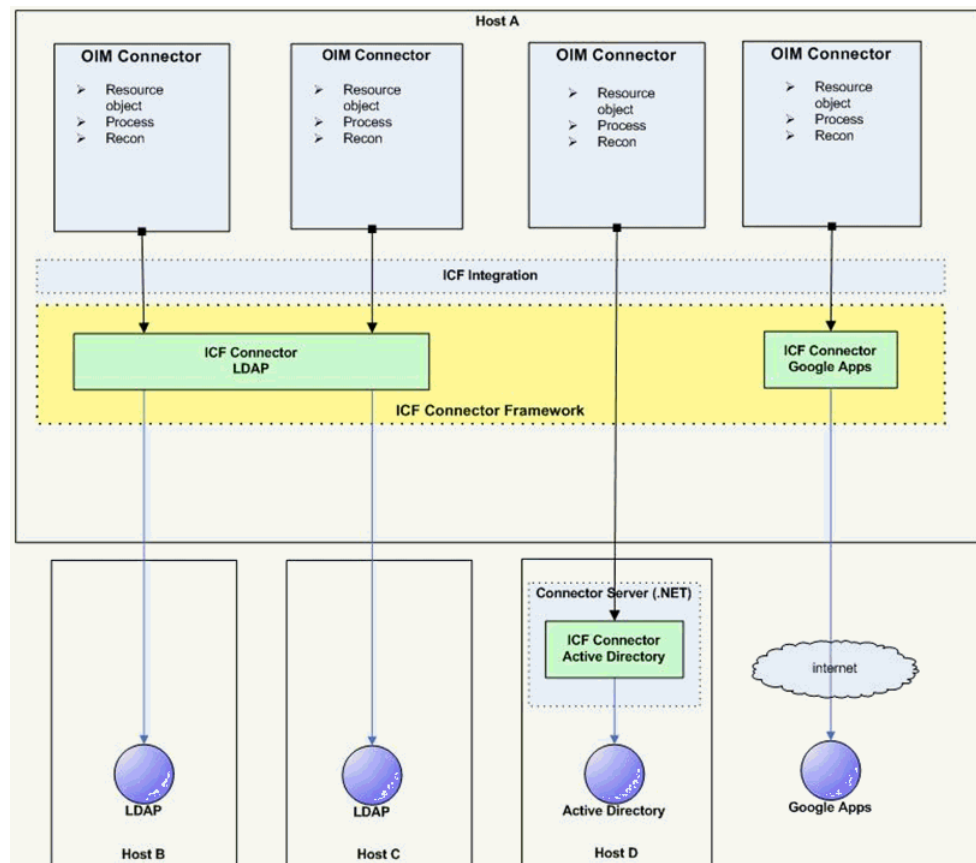
16.5 Using an Identity Connector Server

An identity connector server is required when an identity connector bundle is not directly executed within your application. By using one or more identity connector servers, the ICF architecture permits your application to communicate with externally deployed identity connector bundles. Identity connector servers are available for Java™ and Microsoft .NET Framework applications.

A single connector server can support multiple ICF connectors, and these ICF connectors may be of different connector types. A single ICF connector can be used to communicate with multiple targets.

Figure 16–3 shows how Oracle Identity Manager connectors integrate with resources via ICF connectors:

Figure 16–3 ICF Connectors and Connector Server



In Figure 16–3:

- Oracle Identity Manager connectors do not directly interact with the target resource. Instead, the create, read, update, delete, and query (CRUDQ) operations are performed via the appropriate ICF connector.
- A single ICF Connector can be used to connect to multiple resources of the same resource type. In Figure 16–3, an ICF Connector for LDAP is used to connect to a local LDAP resource, as well as being used to connect to a remote LDAP resource.
- The .NET Connector Server is used to deploy .NET ICF Connectors on the target host. An Active Directory resource is connected in this manner.
- An ICF Connector for Google Apps provides a connection to Google Apps across the Internet.
- While not shown in the diagram, a Connector Server can support multiple ICF Connectors of different resource types.

The types of connector servers are described in the following sections:

- [Using the Java Connector Server](#)
- [Using the Microsoft .NET Framework Connector Server](#)

Tip: Get the following information (defined during installation) for use during either Connector Server configuration.

- Host name and IP address
- Connector Server port
- Connector Server key
- SSL enabled

16.5.1 Using the Java Connector Server

A Java Connector Server is used when you do not want to execute a Java Connector Bundle in the same Java Virtual Machine (JVM) as the application. This deployment may be beneficial in terms of performance as the bundle works faster when deployed on the same host as the managed target system. In addition, use Java Connector Server to eliminate possibility of an application JVM crash because of faulty JNI-based connector.

Using the Java connector server is described in the following sections:

- [Installing and Configuring a Java Connector Server](#)
- [Running the Java Connector Server on Microsoft Windows](#)
- [Running the Java Connector Server on Solaris and Linux](#)
- [Installing an Identity Connector in a Java Connector Server](#)
- [Using SSL to Communicate with a Connector Server](#)

16.5.1.1 Installing and Configuring a Java Connector Server

To install and configure the Java Connector Server:

1. Create a new directory on the computer on which you want to install the Java Connector Server. In this section, `CONNECTOR_SERVER_HOME` represents this directory.
2. Unzip the Java Connector Server package in your new directory from Step 1. Java Connector Server is available for download in the Oracle Technology Network Web site at the following URL:
<http://www.oracle.com/technetwork/index.html>
3. In the `ConnectorServer.properties` file in the `conf/` directory, set the properties as required by your deployment. [Table 16–1](#) lists the properties in the `ConnectorServer.properties` file:

Table 16–1 *Properties in the ConnectorServer.properties File*

Property	Description
<code>connectorserver.port</code>	Port on which the Java Connector Server listens for requests. The default value is <code>8759</code> .
<code>connectorserver.bundleDir</code>	Directory where the connector bundles are deployed. The default value is <code>bundles</code> .
<code>connectorserver.libDir</code>	Directory in which to place dependent libraries. The default value is <code>lib</code> .

Table 16–1 (Cont.) Properties in the ConnectorServer.properties File

Property	Description
connectorserver.usessl	<p>If set to true, the Java Connector Server uses SSL for secure communication. The default value is <code>false</code>.</p> <p>If you specify true, then use the following options on the command line when you start the Java Connector Server:</p> <ul style="list-style-type: none"> ■ <code>-Djavax.net.ssl.keyStore</code> ■ <code>-Djavax.net.ssl.keyStoreType</code> (optional) ■ <code>-Djavax.net.ssl.keyStorePassword</code>
connectorserver.ifaddress	Bind address. To set this property, uncomment it in the file, if required. The bind address can be useful if there are more NICs installed on the computer.
connectorserver.key	Java Connector Server key.

4. Set the properties in the ConnectorServer.properties file, as follows:
 - To set connectorserver.key, run the Java Connector Server with the `/setKey` option. See ["Running the Java Connector Server on Microsoft Windows"](#) on page 16-23 and ["Running the Java Connector Server on Solaris and Linux"](#) on page 16-24 for more information.
 - For all other properties, edit the ConnectorServer.properties file manually.
5. The conf directory also contains the logging.properties file, which you can edit if required by your deployment.

16.5.1.2 Running the Java Connector Server on Microsoft Windows

To run the Java Connector Server on Microsoft Windows, use the ConnectorServer.bat script, as follows:

1. Make sure that you have set the properties required by your deployment in the ConnectorServer.properties file, as described in ["Installing and Configuring a Java Connector Server"](#) on page 16-22.
2. Change to the `CONNECTOR_SERVER_HOME\bin` directory and find the ConnectorServer.bat script.

[Table 16–2](#) lists the options supported by the ConnectorServer.bat script:

Table 16–2 Options Supported by the ConnectorServer.bat Script

Option	Description
<code>/install [serviceName] ["-J java-option"]</code>	<p>Installs the Java Connector Server as a Microsoft Windows service.</p> <p>Optionally, you can specify a service name and Java options. If you do not specify a service name, then the default name is ConnectorServerJava.</p>
<code>/run ["-J java-option"]</code>	<p>Runs the Java Connector Server from the console.</p> <p>Optionally, you can specify Java options. For example, to run the Java Connector Server with SSL:</p> <pre>ConnectorServer.bat /run "-J-Djavax.net.ssl.keyStore=mykeystore.jks" "-J-Djavax.net.ssl.keyStorePassword=password"</pre>

Table 16–2 (Cont.) Options Supported by the ConnectorServer.bat Script

Option	Description
/setKey [key]	Sets the Java Connector Server key. The ConnectorServer.bat script stores the hashed value of the key in the connectorserver.key property in the ConnectorServer.properties file.
/uninstall [serviceName]	Uninstalls the Java Connector Server. If you do not specify a service name, then the script uninstalls the ConnectorServerJava service.

3. If you need to stop the Java Connector Server, then stop the respective Microsoft Windows service.

16.5.1.3 Running the Java Connector Server on Solaris and Linux

To run the Java Connector Server on Solaris and Linux, use the connectorserver.sh script, as follows:

1. Make sure that you have set the properties required by your deployment in the ConnectorServer.properties file, as described in ["Installing and Configuring a Java Connector Server"](#) on page 16-22.
2. Change to the CONNECTOR_SERVER_HOME/bin directory.
3. Use the chmod command to set the permissions to make the connectorserver.sh script executable.
4. Run the connectorserver.sh script.

[Table 16–3](#) lists the options supported by the connectorserver.sh script:

Table 16–3 Options Supported by the connectorserver.sh Script

Option	Description
/run [-Jjava-option]	Runs the Java Connector Server in the console. Optionally, you can specify one or more Java options. For example, to run the Java Connector Server with SSL: <pre>./connectorserver.sh /run -J-Djavax.net.ssl.keyStore=mykeystore.jks -J-Djavax.net.ssl.keyStorePassword=password</pre>
/start [-Jjava-option]	Runs the Java Connector Server in the background. Optionally, you can specify one or more Java options.
/stop	Stops the Java Connector Server, waiting up to 5 seconds for the process to end.
/stop n	Stops the Java Connector Server, waiting up to n seconds for the process to end.
/stop -force	Stops the Java Connector Server. Waits up to 5 seconds, and then uses the kill -KILL command if the process is still running.
/stop n -force	Stops the Java Connector Server. Waits up to n seconds, and then uses the kill -KILL command if the process is still running.
/setKey key	Sets the Java Connector Server key. The connectorserver.sh script stores the hashed value of the key in the connectorserver.key property in the ConnectorServer.properties file.

16.5.1.4 Installing an Identity Connector in a Java Connector Server

This section contains the procedures to deploy a Java Connector Bundle in a Java Connector Server.

1. Change to the bundles directory in your Java Connector Server directory.
2. Copy the Java Connector Bundle JAR to the bundles directory.
3. Add any applicable third party JAR files required by the identity connector to the lib directory.
4. Restart the Java Connector Server.

16.5.1.5 Using SSL to Communicate with a Connector Server

Follow these steps to communicate with a Connector Server using Secure Sockets Layer (SSL).

1. Deploy an SSL certificate to the Connector Server's system.
2. Configure your Connector Server to provide SSL sockets.
3. Configure your application to communicate with the Connector Server using SSL.

Refer to the target system's manual for specific notes on configuring connections to identity connector servers. You will indicate to your application that an SSL connection is required when establishing a connection for each SSL-enabled connector server. Additionally, if any of the SSL certificates used by your connector servers are issued by a non-standard certificate authority, your application must be configured to respect the additional authorities. Refer to your manual for notes regarding certificate authorities.

Note: Java applications may solve the issue of non-standard certificate authorities by expecting the following Java system properties to be passed when launching the application:

- `javax.net.ssl.trustStorePassword`

For example:

```
-Djavax.net.ssl.trustStorePassword=changeit
```

- `javax.net.ssl.trustStore`

For example:

```
-Djavax.net.ssl.trustStore=/usr/myApp_cacerts
```

Alternately, the non-standard certificate authorities may be imported to the standard `${JAVA_HOME}/lib/security/cacerts` directory.

16.5.2 Using the Microsoft .NET Framework Connector Server

The use of a Microsoft .NET Framework (.NET) Connector Server is useful when an application is written in Java but a Connector Bundle is written using C#. Because a Java Platform, Enterprise Edition (JEE™) application cannot load C# classes, you can deploy the C# bundles under a .NET Connector Server. The Java application can then communicate with the C# (.NET) Connector Server over the network. The C# (.NET) Connector Server serves as a proxy to provide any authenticated application access to the C# bundles. The following sections contain additional information.

- [Installing the .NET Connector Server](#)

- [Configuring the .NET Connector Server](#)
- [Configuring Trace Settings](#)
- [Running the .NET Connector Server](#)
- [Installing Multiple Connectors on a .NET Connector Server](#)

16.5.2.1 Installing the .NET Connector Server

The minimum requirements to run a .NET Connector Server are:

- Microsoft Windows Server 2003 or 2008
- Microsoft .NET Framework 3.5 or higher

Refer to the particular .NET identity connector documentation to determine if there are additional requirements.

To install the .NET Connector Server, execute the ServiceInstall.msi file and follow the instructions displayed in the Installation Wizard. Upon completion of the installation, the Connector Server will be installed as a Windows Service.

16.5.2.2 Configuring the .NET Connector Server

Follow this procedure to configure the .NET Connector Server. Common configurations include port, trace and SSL settings as well as the Connector Server key.

1. Start the Microsoft Services Console.
2. Check to see if the Connector Server is currently running. If yes, stop it.
3. Set the key for the Connector Server using the command prompt.

This key is required by any client that connects to this Connector Server.

- a. Change to the directory in which the Connector Server was installed.

By default: \Program Files\Identity Connectors\Connector Server

- b. Execute the following command:

```
ConnectorServer /setkey NEWKEY
```

where *NEWKEY* is the value for the key.

4. Configure additional properties by inspecting the settings in `connectorserver.exe.config`.

The `connectorserver.exe.config` file contains information about the Connector Server. The port, SSL configuration and trace settings are most commonly changed. Port and SSL settings are in a tag called `AppSettings` as follows:

```
<add key="connectorserver.port" value="8759" />
<add key="connectorserver.usessl" value="false" />
<add key="connectorserver.certificatename"
value="ConnectorServerSSLCertificate" />
<add key="connectorserver.ifaddress" value="0.0.0.0" />
```

The port can be set by changing the value of `connectorserver.port`. To use SSL, set the value of `connectorserver.usessl` to true, and set the value of `connectorserver.certificatename` to the name of your certificate store. The listening socket can be bound to a particular address, or can be left as 0.0.0.0. For more information about configuring the Connector Server with SSL, see [Section 16.5.1.5, "Using SSL to Communicate with a Connector Server."](#) For

information on trace setting configurations, see [Section 16.5.2.3, "Configuring Trace Settings."](#)

16.5.2.3 Configuring Trace Settings

The Connector Server uses the standard .NET trace mechanism. Trace settings are defined in the `connectorserver.exe.config` configuration file. [Example 16–33](#) illustrates how they are defined.

Example 16–33 Defined Trace Settings

```
<system.diagnostics>
  <trace autoflush="true" indentsize="4">
    <listeners>
      <remove name="Default" />
      <add name="myListener"
        type="System.Diagnostics.TextWriterTraceListener"
        initializeData="c:\connectorserver2.log"
        traceOutputOptions="DateTime">
      <filter type="System.Diagnostics.EventTypeFilter"
        initializeData="Information" />
      </add>
    </listeners>
  </trace>
</system.diagnostics>
```

The default settings are a good starting point but you may change these settings as follows.

- For less tracing, change the filter type's `initializeData` setting to *Warning* or *Error*.
- For more verbose logging, set the value to *Verbose* or *All*.

Caution: The amount of logging performed has a direct effect on the performance of the Connector Servers.

Any configuration changes require that the Connector Server be stopped and restarted.

Note: For more information about the tracing options, see Microsoft .NET documentation for `System.Diagnostics`.

16.5.2.4 Running the .NET Connector Server

The best way to run the .NET Connector Server is as a Windows Service. During installation, the Connector Server is installed as a Windows service. If this is not adequate for your environment, the Connector Server may be installed or uninstalled as a Windows Service by using the `/install` or `/uninstall` arguments at the command prompt.

To run the Connector Server interactively, issue the command `ConnectorServer /run`.

16.5.2.5 Installing Multiple Connectors on a .NET Connector Server

To install new identity connectors, change to the directory where the Connector Server was installed, extract the new identity connector ZIP into it, and restart the Connector Server.

Developing Identity Connectors

This chapter is a tutorial that walks through the procedures necessary to develop an identity connector using the Identity Connector Framework (ICF) and the Oracle Identity Manager metadata. It includes information about important ICF classes and interfaces, the connector bundle, the connector server, and code samples for implementing a flat file identity connector and creating Oracle Identity Manager metadata for user provisioning and reconciliation processes. It contains the following sections:

- [Developing a Flat File Connector](#)
- [Uploading the Identity Connector Bundle to Oracle Identity Manager Database](#)
- [Provisioning a Flat File Account](#)

17.1 Developing a Flat File Connector

The procedure for developing a flat file connector is to develop an implementation of the Configuration interface followed by the implementation of the Connector class. Before beginning, you must prepare IO representation modules for all flat file connector operations. This might include all or some of the following:

- Read the column names of the flat file and prepare metadata information.
- Add a record to the flat file with the corresponding column values separated by the specified delimiter.
- Delete a record to the flat file based on the UID value.
- Search operations on flat file.

This tutorial is focused on identity connector development, and therefore, these preparations are not discussed in detail.

Note: The following supporting classes are used for file input and output handling during identity connector operations:

- `org.identityconnectors.flatfile.io.FlatFileIOFactory`
- `org.identityconnectors.flatfile.io.FlatFileMetadata`
- `org.identityconnectors.flatfile.io.FlatFileParser`
- `org.identityconnectors.flatfile.io.FlatFileWriter`

See "[Supporting Classes for File Input and Output Handling](#)" on page 17-9 for the implementations of the input and output handling supporting classes.

To develop a flat file connector:

1. Implement the configuration class for the Flat File Connector by extending the `org.identityconnectors.framework.spi.AbstractConfiguration` base class.

[Example 17–1](#) is a sample of this. See [Section 16.3.1.2](#), "[org.identityconnectors.framework.spi.Configuration](#)" for more information.

Example 17–1 Implementation of AbstractConfiguration

```
package org.identityconnectors.flatfile;
import java.io.File;
import org.identityconnectors.flatfile.io.FlatFileIOFactory;
import org.identityconnectors.framework.common.exceptions.ConfigurationException;
import org.identityconnectors.framework.spi.AbstractConfiguration;
import org.identityconnectors.framework.spi.ConfigurationProperty;
/**
 * Class for storing the flat file configuration
 */
public class FlatFileConfiguration extends AbstractConfiguration {
    /**
     * Storage file name
     */
    private File storeFile;
    /**
     * Delimiter used
     */
    private String textFieldDelimiter;
    /**
     * Unique attribute field name
     */
    private String uniqueAttributeName = "";
    /**
     * Change attribute field name. Should be numeric
     */
    private String changeLogAttributeName = "";

    public File getStoreFile() {
        return storeFile;
    }

    public String getTextFieldDelimiter() {
        return textFieldDelimiter;
    }

    public String getUniqueAttributeName() {
        return uniqueAttributeName;
    }

    public String getChangeLogAttributeName() {
        return changeLogAttributeName;
    }

    /**
     * Set the store file
     * @param storeFile
     */
    @ConfigurationProperty(order = 1, helpMessageKey = "USER_ACCOUNT_STORE_HELP",
        displayMessageKey = "USER_ACCOUNT_STORE_DISPLAY")
    public void setStoreFile(File storeFile) {
        this.storeFile = storeFile;
    }
}
```

```

    }

    /**
     * Set the text field delimiter
     * @param textFieldDelimiter
     */
    @ConfigurationProperty(order = 2,
        helpMessageKey = "USER_STORE_TEXT_DELIM_HELP",
        displayMessageKey = "USER_STORE_TEXT_DELIM_DISPLAY")
    public void setTextFieldDelimiter(String textFieldDelimiter) {
        this.textFieldDelimiter = textFieldDelimiter;
    }

    /**
     * Set the field whose values will be considered as unique attributes
     * @param uniqueAttributeName
     */
    @ConfigurationProperty(order = 3, helpMessageKey = "UNIQUE_ATTR_HELP",
        displayMessageKey = "UNIQUE_ATTR_DISPLAY")
    public void setUniqueAttributeName(String uniqueAttributeName) {
        this.uniqueAttributeName = uniqueAttributeName;
    }

    /**
     * Set the field name where change number should be stored
     * @param changeLogAttributeName
     */
    @ConfigurationProperty(order = 3, helpMessageKey = "CHANGELOG_ATTR_HELP",
        displayMessageKey = "CHANGELOG_ATTR_DISPLAY")
    public void setChangeLogAttributeName(String changeLogAttributeName) {
        this.changeLogAttributeName = changeLogAttributeName;
    }
}

@Override
public void validate() {

    // Validate if file exists and is usable
    boolean validFile = (this.storeFile.exists() &&
        this.storeFile.canRead() &&
        this.storeFile.canWrite() &&
        this.storeFile.isFile());

    if (!validFile)
        throw new ConfigurationException("User store file not valid");

    // Validate if there is a field on name of unique attribute field name
    // Validate if there is a field on name of change attribute field name
    FlatFileIOFactory.getInstance(this);
    // Initialization does the validation
}
}
}

```

2. Create connector class for the Flat File Connector by implementing the `org.identityconnectors.framework.spi.Connector` interface.

Example 17–2 implements the `CreateOp`, `DeleteOp`, `SearchOp` and `UpdateOp` interfaces and thus supports all four operations. The `FlatFileMetadata`, `FlatFileParser` and `FlatFileWriter` classes are supporting classes. Their implementation is not shown as they do not belong to the ICF.

Example 17–2 Implementation of PoolableConnector

```

package org.identityconnectors.flatfile;

import java.util.HashSet;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import java.util.Set;

import org.identityconnectors.flatfile.io.FlatFileIOFactory;
import org.identityconnectors.flatfile.io.FlatFileMetadata;
import org.identityconnectors.flatfile.io.FlatFileParser;
import org.identityconnectors.flatfile.io.FlatFileWriter;
import org.identityconnectors.framework.api.operations.GetApiOp;
import org.identityconnectors.framework.common.exceptions.AlreadyExistsException;
import org.identityconnectors.framework.common.exceptions.ConnectorException;
import org.identityconnectors.framework.common.objects.Attribute;
import org.identityconnectors.framework.common.objects.AttributeInfo;
import org.identityconnectors.framework.common.objects.AttributeInfoBuilder;
import org.identityconnectors.framework.common.objects.ConnectorObject;
import org.identityconnectors.framework.common.objects.ConnectorObjectBuilder;
import org.identityconnectors.framework.common.objects.ObjectClass;
import org.identityconnectors.framework.common.objects.OperationOptions;
import org.identityconnectors.framework.common.objects.ResultsHandler;
import org.identityconnectors.framework.common.objects.Schema;
import org.identityconnectors.framework.common.objects.SchemaBuilder;
import org.identityconnectors.framework.common.objects.Uid;
import
org.identityconnectors.framework.common.objects.filter.AbstractFilterTranslator;
import org.identityconnectors.framework.common.objects.filter.FilterTranslator;
import org.identityconnectors.framework.spi.Configuration;
import org.identityconnectors.framework.spi.ConnectorClass;
import org.identityconnectors.framework.spi.PoolableConnector;
import org.identityconnectors.framework.spi.operations.CreateOp;
import org.identityconnectors.framework.spi.operations.DeleteOp;
import org.identityconnectors.framework.spi.operations.SchemaOp;
import org.identityconnectors.framework.spi.operations.SearchOp;
import org.identityconnectors.framework.spi.operations.UpdateOp;

/**
 * The main connector class
 */
@ConnectorClass(configurationClass = FlatFileConfiguration.class, displayNameKey =
"FlatFile")
public class FlatFileConnector implements SchemaOp, CreateOp, DeleteOp,
    UpdateOp, SearchOp<Map<String, String>>, GetApiOp, PoolableConnector {

    private FlatFileConfiguration flatFileConfig;
    private FlatFileMetadata flatFileMetadata;
    private FlatFileParser flatFileParser;
    private FlatFileWriter flatFileWriter;
    private boolean alive = false;

    @Override
    public Configuration getConfiguration() {
        return this.flatFileConfig;
    }

    @Override
    public void init(Configuration config) {

```



```

this.flatFileConfig = (FlatFileConfiguration) config;

FlatFileIOFactory flatFileIOFactory =
    FlatFileIOFactory.getInstance(flatFileConfig);
this.flatFileMetadata = flatFileIOFactory.getMetadataInstance();
this.flatFileParser = flatFileIOFactory.getFileParserInstance();
this.flatFileWriter = flatFileIOFactory.getFileWriterInstance();
this.alive = true;
System.out.println("init called: Initialization done");
}

@Override
public void dispose() {
    this.alive = false;
}

@Override
public Schema schema() {
    SchemaBuilder flatFileSchemaBldr = new SchemaBuilder(this.getClass());
    Set<AttributeInfo> attrInfos = new HashSet<AttributeInfo>();
    for (String fieldName : flatFileMetadata.getOrderedTextFieldNames()) {
        AttributeInfoBuilder attrBuilder = new AttributeInfoBuilder();
        attrBuilder.setName(fieldName);
        attrBuilder.setCreateable(true);
        attrBuilder.setUpdateable(true);
        attrInfos.add(attrBuilder.build());
    }

    // Supported class and attributes
    flatFileSchemaBldr.defineObjectClass
        (ObjectClass.ACCOUNT.getDisplayNameKey(), attrInfos);
    System.out.println("schema called: Built the schema properly");
    return flatFileSchemaBldr.build();
}

@Override
public Uid create(ObjectClass arg0, Set<Attribute> attrs,
    OperationOptions ops) {

    System.out.println("Creating user account " + attrs);
    assertUserObjectClass(arg0);
    try {
        FlatFileUserAccount accountRecord = new FlatFileUserAccount(attrs);
        // Assert uid is there
        assertUidPresence(accountRecord);

        // Create the user
        this.flatFileWriter.addAccount(accountRecord);

        // Return uid
        String uniqueAttrField = this.flatFileConfig
            .getUniqueAttributeName();
        String uniqueAttrVal = accountRecord
            .getAttributeValue(uniqueAttrField);
        System.out.println("User " + uniqueAttrVal + " created");

        return new Uid(uniqueAttrVal);
    } catch (Exception ex) {

        // If account exists

```

```
        if (ex.getMessage().contains("exists"))
            throw new AlreadyExistsException(ex);

        // For all other causes
        System.out.println("Error in create " + ex.getMessage());
        throw ConnectorException.wrap(ex);
    }
}

@Override
public void delete(ObjectClass arg0, Uid arg1, OperationOptions arg2) {
    final String uidVal = arg1.getUidValue();
    this.flatFileWriter.deleteAccount(uidVal);
    System.out.println("Account " + uidVal + " deleted");
}

@Override
public Uid update(ObjectClass arg0, Uid arg1, Set<Attribute> arg2,
    OperationOptions arg3) {
    String accountIdentifier = arg1.getUidValue();
    // Fetch the account
    FlatFileUserAccount accountToBeUpdated = this.flatFileParser
        .getAccount(accountIdentifier);

    // Update
    accountToBeUpdated.updateAttributes(arg2);
    this.flatFileWriter
        .modifyAccount(accountIdentifier, accountToBeUpdated);
    System.out.println("Account " + accountIdentifier + " updated");

    // Return new uid
    String newAccountIdentifier = accountToBeUpdated
        .getAttributeValue(this.flatFileConfig.getUniqueAttributeName());
    return new Uid(newAccountIdentifier);
}

@Override
public FilterTranslator<Map<String, String>> createFilterTranslator(
    ObjectClass arg0, OperationOptions arg1) {
    // TODO: Create a fine grained filter translator

    // Return a dummy object as its not applicable here.
    // All processing happens in the execute query
    return new AbstractFilterTranslator<Map<String, String>>() {
    };
}

@Override
public ConnectorObject getObject(ObjectClass arg0, Uid uid,
    OperationOptions arg2) {
    // Return matching record
    String accountIdentifier = uid.getUidValue();
    FlatFileUserAccount userAcc = this.flatFileParser
        .getAccount(accountIdentifier);
    ConnectorObject userAccConnObject = convertToConnectorObject(userAcc);
    return userAccConnObject;
}

/*
 * (non-Javadoc)

```

```

* This is the search implementation.
* The Map passed as the query here, will map to all the records with
* matching attributes.
*
* The record will be filtered if any of the matching attributes are not
* found
*
* @see
* org.identityconnectors.framework.spi.operations.SearchOp#executeQuery
* (org.identityconnectors.framework.common.objects.ObjectClass,
* java.lang.Object,
* org.identityconnectors.framework.common.objects.ResultsHandler,
* org.identityconnectors.framework.common.objects.OperationOptions)
*/
@Override
public void executeQuery(ObjectClass objectClass,
    Map<String, String> matchSet, ResultsHandler resultHandler,
    OperationOptions ops) {

    System.out.println("Inside executeQuery");

    // Iterate over the records and handle individually
    Iterator<FlatFileUserAccount> userAccountIterator = this.flatFileParser
        .getAccountIterator(matchSet);

    while (userAccountIterator.hasNext()) {
        FlatFileUserAccount userAcc = userAccountIterator.next();
        ConnectorObject userAccObject = convertToConnectorObject(userAcc);
        if (!resultHandler.handle(userAccObject)) {
            System.out.println("Not able to handle " + userAcc);
            break;
        }
    }
}

private void assertUserObjectClass(ObjectClass arg0) {
    if (!arg0.equals(ObjectClass.ACCOUNT))
        throw new UnsupportedOperationException(
            "Only user account operations supported.");
}

private void assertUidPresence(FlatFileUserAccount accountRecord) {
    String uniqueAttrField = this.flatFileConfig.getUniqueAttributeName();
    String uniqueAttrVal = accountRecord.getAttributeValue(uniqueAttrField);

    if (uniqueAttrVal == null) {
        throw new IllegalArgumentException("Unique attribute not passed");
    }
}

private ConnectorObject convertToConnectorObject(FlatFileUserAccount userAcc)
{
    ConnectorObjectBuilder userObjBuilder = new ConnectorObjectBuilder();
    // Add attributes
    List<String> attributeNames = this.flatFileMetadata
        .getOrderedTextFieldNames();
    for (String attributeName : attributeNames) {
        String attributeVal = userAcc.getAttributeValue(attributeName);
        userObjBuilder.addAttribute(attributeName, attributeVal);
    }
}

```

```

        if (attributeName.equals(this.flatFileConfig
            .getUniqueAttributeName())) {
            userObjBuilder.setUid(attributeVal);
            userObjBuilder.setName(attributeVal);
        }
    }
    return userObjBuilder.build();
}

@Override
public void checkAlive() {
    if (!alive)
        throw new RuntimeException("Connection not alive");
}
}
}

```

3. This connector supports only the ContainsAllValuesFilter operation. Implement the ContainsAllValuesFilter operation [Example 17-3](#) illustrates the sample implementation of `org.identityconnectors.framework.common.objects.filter.AbstractFilterTranslator<T>` that defines the filter operation.

Example 17-3 Implementation of AbstractFilterTranslator<T>

```

package org.identityconnectors.flatfile.filteroperations;

import java.util.HashMap;
import java.util.Map;

import org.identityconnectors.framework.common.objects.Attribute;
import
org.identityconnectors.framework.common.objects.filter.AbstractFilterTranslator;
import
org.identityconnectors.framework.common.objects.filter.ContainsAllValuesFilter;

public class ContainsAllValuesImpl extends AbstractFilterTranslator<Map<String,
String>>{
    @Override
    protected Map<String, String> createContainsAllValuesExpression(
        ContainsAllValuesFilter filter, boolean not) {
        Map<String, String> containsAllMap = new HashMap<String, String>();
        Attribute attr = filter.getAttribute();
        containsAllMap.put(attr.getName(), attr.getValue().get(0).toString());
        return containsAllMap;
    }
}

```

4. Create the connector bundle JAR. The MANIFEST.MF file must contain the following entries:
 - ConnectorBundle-FrameworkVersion
 - ConnectorBundle-Name
 - ConnectorBundle-Version

[Example 17-4](#) shows the contents of the MANIFEST.MF file:

Example 17–4 The MANIFEST.MF File

```

Manifest-Version: 1.0
Ant-Version: Apache Ant 1.7.0
Created-By: 14.1-b02 (Sun Microsystems Inc.)
ConnectorBundle-FrameworkVersion: 1.0
ConnectorBundle-Name: org.identityconnectors.flatfile
ConnectorBundle-Version: 1.0
Build-Number: 609
Subversion-Revision: 4582

```

5. Update the connector bundle JAR as created in step 4. To do so:
 - a. Extract the connector bundle JAR into any desired location.
 - b. Create a lib directory in the directory in which you extracted the JAR.
 - c. Add the dependent third-party JARs into the lib directory.
 - d. JAR the entire directory.

Note: The MANIFEST.MF file must contain the entires listed in step 4.

17.1.1 Supporting Classes for File Input and Output Handling

This section shows the implementation of the following supporting classes for file input and output handling:

- [Example 17–5, "FlatFileIOFactory"](#)
- [Example 17–6, "FlatFileMetadata"](#)
- [Example 17–7, "FlatFileParser"](#)
- [Example 17–8, "FlatFileWriter"](#)

[Example 17–5](#) shows the implementation of the FlatFileIOFactory supporting class:

Example 17–5 FlatFileIOFactory

```

package org.identityconnectors.flatfile.io;

import org.identityconnectors.flatfile.FlatFileConfiguration;

public class FlatFileIOFactory {

    private FlatFileMetadata flatFileMetadata;
    private FlatFileConfiguration flatFileConfig;

    /**
     * Provides instance of the factory
     * @param flatfileConfig Configuration bean for the flat file
     */
    public static FlatFileIOFactory getInstance(FlatFileConfiguration fileConfig)
    {
        return new FlatFileIOFactory(fileConfig);
    }

    /**
     * Making it private to avoid public instantiation. Encouraging use of
     * getInstance
     * @param fileConfig

```

```

    */
    private FlatFileIOFactory(FlatFileConfiguration fileConfig) {
        this.flatFileConfig = fileConfig;
        this.flatFileMetadata = new FlatFileMetadata(flatFileConfig);
        System.out.println("Metadata set");
    }

    /**
     * Returns the metadata instance
     * @return
     */
    public FlatFileMetadata getMetadataInstance() {
        return this.flatFileMetadata;
    }

    /**
     * Returns the FlatFileParser instance
     * @return
     */
    public FlatFileParser getFileParserInstance() {
        return new FlatFileParser(this.flatFileMetadata, this.flatFileConfig);
    }

    /**
     * Returns the FlatFileWriter instance
     * @return
     */
    public FlatFileWriter getFileWriterInstance() {
        return new FlatFileWriter(this.flatFileMetadata, this.flatFileConfig);
    }
}

```

[Example 17-6](#) shows the implementation of the FlatFileMetaData supporting class:

Example 17-6 FlatFileMetadata

```

package org.identityconnectors.flatfile.io;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.StringTokenizer;

import org.identityconnectors.flatfile.FlatFileConfiguration;

/**
 * This class contains all the metadata related information Example: Ordering of
 * columns, Number of columns etc.
 *
 * @author harsh
 */
public class FlatFileMetadata {

    private FlatFileConfiguration fileConfig;

    private List<String> orderedTextFieldNames;
}

```

```

private String changeLogFieldName;
private String uniqueAttributeFileName;

/**
 * Instantiates the class with the file configuration.
 * Making it package private to encourage instantiation from Factory class
 * @param fileConfig
 */
FlatFileMetadata(FlatFileConfiguration fileConfig) {
    /**
     * Ideally you should not take connector specific configuration class in
     * flat file resource classes. Change if this has to go to production.
     * Probably make another configuration class for flat file with same
     * signatures.
     */
    this.fileConfig = fileConfig;

    initializeMetadata();
    validateConfigProps();
}

/**
 * Returns the text field names in the order of their storage
 *
 * @return
 */
public List<String> getOrderedTextFieldNames() {
    return this.orderedTextFieldNames;
}

/**
 * Returns the number of columns
 */
public int getNumberOfFields() {
    int numberOfTextFields = this.orderedTextFieldNames.size();
    return numberOfTextFields;
}

/**
 * Specifies if number of tokens are matching with the standard length of
 * metadata
 * @param countTokens
 * @return
 */
public boolean isDifferentFromNumberOfFields(int countTokens) {
    return (getNumberOfFields() != countTokens);
}

/**
 * Reads the header line and sets the metadata
 */
private void initializeMetadata() {
    // Read the file.
    File recordsStore = this.fileConfig.getStoreFile();

    try {
        BufferedReader storeFileReader = new BufferedReader(new FileReader(
            recordsStore.getAbsolutePath()));

        // Read the header line

```

```

String headerString = storeFileReader.readLine();

// Tokenize the headerString
StringTokenizer tokenizer = new StringTokenizer(headerString,
        fileConfig.getTextFieldDelimiter());

this.orderedTextFieldNames = new ArrayList<String>();
while (tokenizer.hasMoreTokens()) {
    String header = tokenizer.nextToken();
    this.orderedTextFieldNames.add(header);
}

System.out.println("Columns read - " + this.orderedTextFieldNames);
} catch (IOException e) {
    throw new RuntimeException("How can I read a corrupted file");
}

// Store the change log and unique attribute field names
this.changeLogFieldName = fileConfig.getChangeLogAttributeName();
this.uniqueAttributeFiledName = fileConfig.getUniqueAttributeName();
}

/**
 * Validate if the attribute names in config props object are present in the
 * column names
 *
 * @throws RuntimeException
 *         if validation fails
 */
private void validateConfigProps() {
    // Check if unique attribute col name is present
    if (!this.orderedTextFieldNames.contains(this.changeLogFieldName))
        throw new RuntimeException("Change log field name "
            + this.changeLogFieldName + " not found in the store file ");

    // Check if change col name is present
    if (!this.orderedTextFieldNames.contains(this.uniqueAttributeFiledName))
        throw new RuntimeException("Unique attribute field name "
            + this.uniqueAttributeFiledName
            + " not found in the store file");
}
}

```

[Example 17-7](#) shows the implementation of the FlatFileParser supporting class:

Example 17-7 FlatFileParser

```

package org.identityconnectors.flatfile.io;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.Map;

import org.identityconnectors.flatfile.FlatFileConfiguration;
import org.identityconnectors.flatfile.FlatFileUserAccount;

```



```

import org.identityconnectors.flatfile.utils.AccountConversionHandler;

public class FlatFileParser {

    private File recordsStore;
    private FlatFileConfiguration fileConfig;
    private FlatFileMetadata metadata;
    private AccountConversionHandler accountConverter;

    /**
     * Instantiates the parser class. Making it package private to encourage
     * instantiation from Factory class
     *
     * @param metadata
     * @param fileConfig
     */
    FlatFileParser(FlatFileMetadata metadata, FlatFileConfiguration fileConfig) {
        this.fileConfig = fileConfig;
        this.recordsStore = fileConfig.getStoreFile();
        this.accountConverter = new AccountConversionHandler(metadata,
            fileConfig);
        this.metadata = metadata;
    }

    /**
     * Returns all accounts in the file
     *
     * @return
     */
    public List<FlatFileUserAccount> getAllAccounts() {
        try {
            BufferedReader userRecordReader = new BufferedReader(
                new FileReader(recordsStore.getAbsolutePath()));
            String recordStr;

            // Skip headers
            userRecordReader.readLine();

            // Loop over records and make list of objects
            List<FlatFileUserAccount> allAccountRecords = new
                ArrayList<FlatFileUserAccount>();
            while ((recordStr = userRecordReader.readLine()) != null) {
                try {
                    FlatFileUserAccount accountRecord = accountConverter
                        .convertStringRecordToAccountObj(recordStr);
                    allAccountRecords.add(accountRecord);
                } catch (RuntimeException e) {
                    System.out.println("Invalid entry " + e.getMessage());
                }
            }
            userRecordReader.close();

            return allAccountRecords;
        } catch (IOException e) {
            throw new RuntimeException("How can I read a corrupted file");
        }
    }

    /**
     * Gets the account of matching account identifier

```

```

*
* @param accountIdentifier
* @return
*/
public FlatFileUserAccount getAccount(String accountIdentifier) {

    /*
    * I know its not right to get all account details. Don't want to focus
    * on efficiency and scalability as this is just a sample.
    */
    // Iterate over all records and check for matching account
    Map<String, String> matchSet = new HashMap<String, String>();
    matchSet.put(fileConfig.getUniqueAttributeName(), accountIdentifier);
    for (FlatFileUserAccount userRecord : getAllAccounts()) {
        if (userRecord.hasMatchingAttributes(matchSet))
            return userRecord;
    }

    // Got nothing..
    return null;
}

/**
* Returns all records with matching Attributes If more than attributes are
* passed. it will check all the attributes
*
* @param matchSet
*         Checks if all provided attributes are matched
*/
public List<FlatFileUserAccount> getAccountsByMatchedAttrs(
    Map<String, String> matchSet) {
    /*
    * I know its not right to get all account details. Don't want to focus
    * on efficiency and scalability as this is just a sample.
    */
    // Iterate over all records and check for matching account
    List<FlatFileUserAccount> matchingRecords = new
ArrayList<FlatFileUserAccount>();
    for (FlatFileUserAccount userRecord : getAllAccounts()) {
        if (userRecord.hasMatchingAttributes(matchSet))
            matchingRecords.add(userRecord);
    }

    return matchingRecords;
}

/**
* Returns the records that fall after the specified change number This
* function helps in checking the function of sync
*
* @param changeNumber
*         the change number for the last search
*/
public List<FlatFileUserAccount> getUpdatedAccounts(int changeNumber) {
    /*
    * I know its not right to get all account details. Don't want to focus
    * on efficiency and scalability as this is just a sample.
    */
    // Iterate over all records and check for matching account
    List<FlatFileUserAccount> matchingRecords = new

```

```

ArrayList<FlatFileUserAccount>();
    String changeLogAttrName = fileConfig.getChangeLogAttributeName();
    for (FlatFileUserAccount userRecord : getAllAccounts()) {
        int recordChangeNumber = userRecord
            .getChangeNumber(changeLogAttrName);
        if (recordChangeNumber >= changeNumber)
            matchingRecords.add(userRecord);
    }
    return matchingRecords;
}

/**
 * Returns an iterator that iterates over the records. This is provided for
 * dynamic retrieval of records
 *
 * @param matchSet
 *         Filters the records by matching the given attributes. Use null
 *         or empty set to avoid filtering
 * @return
 */
public Iterator<FlatFileUserAccount> getAccountIterator(
    Map<String, String> matchSet) {
    Iterator<FlatFileUserAccount> recordIterator = new FlatFileLineIterator(
        this.metadata, this.fileConfig, matchSet);

    return recordIterator;
}

/**
 * Gives the next change number. Logic is max of existing change numbers + 1
 * @return
 */
public int getNextChangeNumber() {
    int maximumChangeNumber = 0;

    /*
     * I know its not right to get all account details. Don't want to focus
     * on efficiency and scalability as this is just a sample.
     */
    // Iterate over all records and check for matching account
    String changeLogAttrName = fileConfig.getChangeLogAttributeName();
    for (FlatFileUserAccount userRecord : getAllAccounts()) {
        int changeNumber = userRecord.getChangeNumber(changeLogAttrName);

        if (changeNumber >= maximumChangeNumber) {
            maximumChangeNumber = changeNumber + 1;
        }
    }
    return maximumChangeNumber;
}
}

```

[Example 17-8](#) shows the implementation of the FlatFileWriter supporting class:

Example 17-8 FlatFileWriter

```

package org.identityconnectors.flatfile.io;

import java.io.BufferedReader;

```

```
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

import org.identityconnectors.flatfile.FlatFileConfiguration;
import org.identityconnectors.flatfile.FlatFileUserAccount;
import org.identityconnectors.flatfile.utils.AccountConversionHandler;

/**
 * Class for searching operations on files
 *
 * @author Harsh
 */
public class FlatFileWriter {

    private File recordsStore;
    private FlatFileParser recordParser;
    private FlatFileConfiguration fileConfig;
    private AccountConversionHandler accountConverter;

    /**
     * Initializes the writer with the configuration Making it package private
     * to encourage use of Factory class for global instantiation
     *
     * @param metadata
     * @param fileConfig
     */
    FlatFileWriter(FlatFileMetadata metadata, FlatFileConfiguration fileConfig) {
        this.fileConfig = fileConfig;

        this.recordsStore = fileConfig.getStoreFile();
        recordParser = new FlatFileParser(metadata, fileConfig);
        accountConverter = new AccountConversionHandler(metadata, fileConfig);
    }

    /**
     * Appends the user record at the end of
     *
     * @param accountRecord
     */
    public void addAccount(FlatFileUserAccount accountRecord) {
        try {
            BufferedWriter userRecordWriter = new BufferedWriter(
                new FileWriter(this.recordsStore.getAbsolutePath(), true));

            // Set the latest changelog number
            int latestChangeNumber = recordParser.getNextChangeNumber();
            accountRecord.setChangeNumber(fileConfig
                .getChangeLogAttributeName(), latestChangeNumber);

            // Validate if same account id doesn't exist
            String accountId = accountRecord.getAttributeValue(fileConfig
                .getUniqueAttributeName());
            FlatFileUserAccount accountByAccountId = recordParser
                .getAccount(accountId);

            if (accountByAccountId != null)
                throw new RuntimeException("Account " + accountId
```

```

        + " already exists");

        // Put the user record in formatted way
        String userRecordAsStr = accountConverter
            .convertAccountObjToStringRecord(accountRecord);
        userRecordWriter.write("\n" + userRecordAsStr);

        // Close the output stream
        userRecordWriter.close();
    } catch (IOException e) { // Catch exception if any
        throw new RuntimeException("How can I write on a corrupted file");
    }
}

/**
 * Removes the entry for respective account identifier
 *
 * @param accountIdentifier
 */
public void deleteAccount(String accountIdentifier) {
    String blankRecord = "";
    this.modifyAccountInStore(accountIdentifier, blankRecord);
}

/**
 * Updates the entry with respective account identifier
 *
 * @param accountIdentifier
 * @param updatedAccountRecord
 * @return new accountIdentifier
 */
public String modifyAccount(String accountIdentifier,
    FlatFileUserAccount updatedAccountRecord) {

    // Frame a record string and update back to file
    int nextChangeNumber = recordParser.getNextChangeNumber();

    String changeNumberFieldName = fileConfig.getChangeLogAttributeName();
    updatedAccountRecord.setChangeNumber(changeNumberFieldName,
        nextChangeNumber);

    String newRecordAsStr = accountConverter
        .convertAccountObjToStringRecord(updatedAccountRecord);
    // Update to the file
    this.modifyAccountInStore(accountIdentifier, newRecordAsStr);

    // Return new UID
    String uniqueAttrFieldName = fileConfig.getUniqueAttributeName();
    String newAccountIdentifier = updatedAccountRecord
        .getAttributeValue(uniqueAttrFieldName);
    return newAccountIdentifier;
}

/**
 * Returns the complete flat file as string.
 *
 * @return
 */
private String getCompleteFlatFileAsStr() {
    try {

```

```

        BufferedReader userRecordReader = new BufferedReader(
            new FileReader(recordsStore.getAbsolutePath()));
        String recordStr;

        // Loop over records and make list of objects
        StringBuilder flatFileStr = new StringBuilder();
        while ((recordStr = userRecordReader.readLine()) != null) {
            if (!recordStr.isEmpty())
                flatFileStr.append(recordStr + "\n");
        }
        userRecordReader.close();

        return flatFileStr.toString();
    } catch (IOException e) {
        throw new RuntimeException("How can I read a corrupted file");
    }
}

/**
 * Updates the account with the new record. this can also be used for delete
 *
 * @param accountIdentifier
 * @param updatedRecord
 */
private void modifyAccountInStore(String accountIdentifier,
    String updatedRecord) {
    try {
        // Load the complete flat file
        String completeFlatFile = this.getCompleteFlatFileAsStr();

        // Construct the string to be removed and replace it with blank
        FlatFileUserAccount accountToBeRemoved = recordParser
            .getAccount(accountIdentifier);
        String updatableString = accountConverter
            .convertAccountObjToStringRecord(accountToBeRemoved);
        String updatedFlatFile = completeFlatFile.replaceAll(
            updatableString, updatedRecord);

        // Rewrite the file
        BufferedWriter userRecordWriter = new BufferedWriter(
            new FileWriter(this.recordsStore.getAbsolutePath(), false));
        userRecordWriter.write(updatedFlatFile);

        /*** debug ***/
        System.out.println("Old string " + updatableString);
        System.out.println("New String" + updatedRecord);
        System.out.println("new file - " + updatedFlatFile);

        /***/
        // Close the output stream
        userRecordWriter.close();
    } catch (IOException e) { // Catch exception if any
        throw new RuntimeException("How can I write on a corrupted file");
    }
}
}

```

17.2 Uploading the Identity Connector Bundle to Oracle Identity Manager Database

The identity connector bundle must be available to ICF in Oracle Identity Manager database. Follow the list of sections in order to integrate the ICF identity connector with Oracle Identity Manager. Some of the procedures include configuration by using the Oracle Identity Manager Design Console.

- [Registering the Connector Bundle with Oracle Identity Manager](#)
- [Creating Basic Identity Connector Metadata](#)
- [Creating Provisioning Metadata](#)
- [Creating Reconciliation Metadata](#)

17.2.1 Registering the Connector Bundle with Oracle Identity Manager

The connector bundle must be available for the Connector Server local to Oracle Identity Manager. Following is the procedure to accomplish this:

1. Copy the connector bundle JAR to the machine on which Oracle Identity Manager is installed.
2. Run the following command to upload the JAR.

```
$MW_HOME/server/bin/UploadJars.sh
```

Note: In this chapter, *DW_HOME* represents *\$MW_HOME/Oracle_IDM1*.

3. Select ICFBundle as the JAR type.
4. Enter the location of the connector bundle JAR.
5. Press **Enter**.

17.2.2 Creating Basic Identity Connector Metadata

This metadata configuration is needed for both provisioning and reconciliation. The set of procedures in this section are completed by using the Oracle Identity Manager Design Console.

- [Creating the IT Resource Type Definition](#)
- [Creating the Resource Object](#)
- [Creating the Main Configuration Lookup](#)
- [Creating Object Type Configuration Lookup](#)

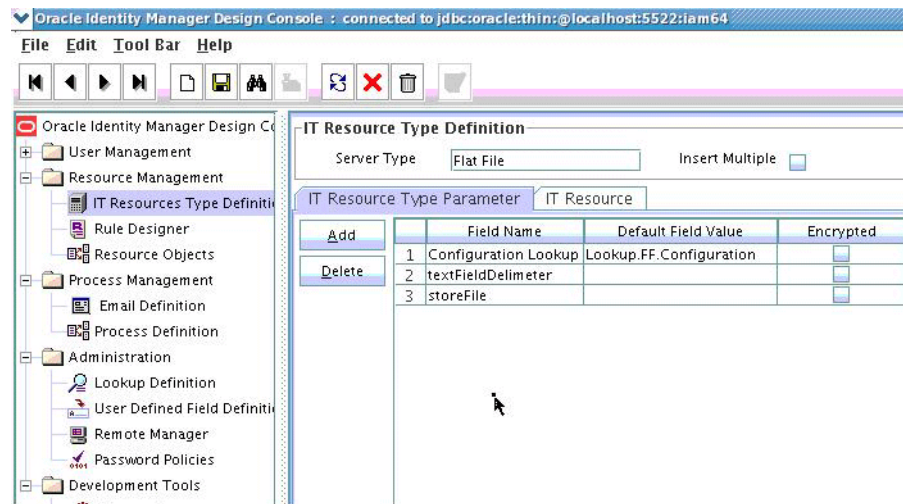
17.2.2.1 Creating the IT Resource Type Definition

An IT resource type definition is the representation of a resource's connection information. The configuration parameters in the IT resource type definition should be matched with the configuration parameters of the connector bundle. The values of the parameters in the IT resource will be set in the bundle configuration.

Note: You may include parameters the bundle configuration is not using. They produce no negative effects on the bundle operations.

1. Log in to the Oracle Identity Manager Design Console.
2. Click IT Resource Type Definition under Resource Management.
3. Create a new IT Resource Type Definition with the Server Type defined as Flat File.
4. Add the following parameters as illustrated in [Figure 17-1](#).
 - **Configuration Lookup** is the marker of the main configuration lookup for the resource. The name of the parameter must be Configuration Lookup. It is a good practice to add a value to Default Field Value.
 - **textFieldDelimiter** maps to the textFieldDelimiter parameter in the bundle configuration. The value of this parameter will be passed.
 - **storeFile** maps to the storeFile parameter in the bundle configuration. The value of this parameter will be passed.

Figure 17-1 IT Resource Type Definition in Design Console

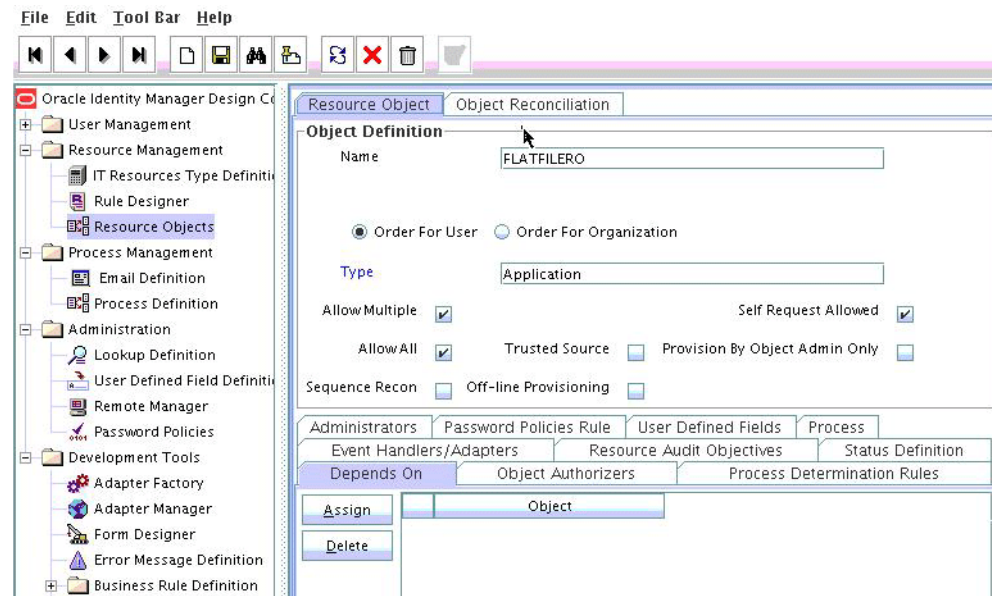


17.2.2.2 Creating the Resource Object

The resource object is the Oracle Identity Manager representation of a resource. The connector bundle is tied to the resource object.

1. Log in to the Oracle Identity Manager Design Console.
2. Click Resource Objects under Resource Management.
3. Create a new resource object with the name FLATFILERO.

As the resource object is a target resource don't check the Trusted Source box as illustrated in [Figure 17-2](#).

Figure 17–2 Resource Objects in Design Console

17.2.2.3 Creating Lookups

Separate lookups have to be defined for different objects supported by the connector bundle. This lookup can contain provisioning and reconciliation related information for those objects. The Main Configuration Lookup is the root for object specific lookups as it contains the pointers to those lookups. The following sections contain information on how to create lookups.

- [Creating the Main Configuration Lookup](#)
- [Creating Object Type Configuration Lookup](#)

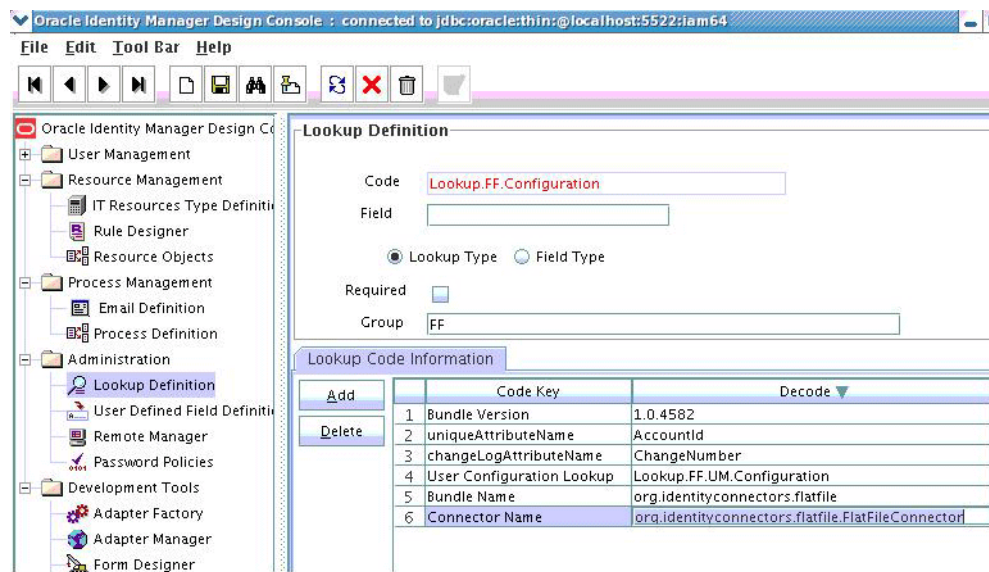
17.2.2.3.1 Creating the Main Configuration Lookup The Configuration Lookup (as defined in [Section 17.2.2.1, "Creating the IT Resource Type Definition"](#)) holds connector bundle configurations that are not counted as connection information. If a configuration parameter is not found in the IT Resource Type Definition, Oracle Identity Manager will look in the Configuration Lookup. The main Configuration Lookup contains bundle properties and bundle configurations. Bundle Property parameters are mandatory as they are needed for identifying the correct bundle. Bundle configurations that are not defined as part of the IT resource type definition (discussed in [Section 17.2.2.1, "Creating the IT Resource Type Definition"](#)) can be declared here.

Note: The values for Code Key should match exactly as illustrated. The values for Decode are specific to the connector bundle.

1. Log in to the Oracle Identity Manager Design Console.
2. Click Lookup Definition under Administration.
3. Create a new lookup and add Lookup.FF.Configuration as the value for Code.
4. Add the following Lookup Code Information as illustrated in [Figure 17–3](#).
 - Add *VERSION* as the required Bundle Version.
 - Add *org.identityconnectors.flatfile* as the required Bundle Name.

- Add `org.identityconnectors.flatfile.FlatFileConnector` as the required Connector Name.
- Add AccountId as the value of uniqueAttributeName. AccountId is a unique string identifier that represents the account to be provisioned or reconciled. It is the name of the column in the flat file. AccountId is unique and is used to represent a user (account detail) uniquely.
- Add ChangeNumber as the value of changeLogAttributeName. When an account is created, a number is attached to it indicating the total accounts created. This value is maintained in the variable called ChangeNumber.
- `OBJECT_TYPE_NAME` Configuration Lookup is the configuration lookup for the particular object type. In this example, the object type is User as User Configuration Lookup is defined.

Figure 17–3 Lookup Definition in Design Console



17.2.2.3.2 Creating Object Type Configuration Lookup Object type configuration lookup contains the parameters specific to the particular object type. Object type is an entity over which an identity connector operates. It is mapped to ICF ObjectClass. In [Section 17.2.2.3.1, "Creating the Main Configuration Lookup,"](#) User Configuration Lookup has been referenced so that User is the object type, in this case mapped to ObjectClass.ACCOUNT. (Roles and UserJobData are two other object types.) The object type name has to match with ObjectClass name supported by the identity connector bundle. The User object type is mapped to predefined ObjectClass.ACCOUNT, the Group object type is mapped to predefined ObjectClass.GROUP. If the identity connector supports multiple objects, then this step must be repeated for each.

Note: Because these use cases cover only the basic functionality, the configuration is kept to the mandatory attribute.

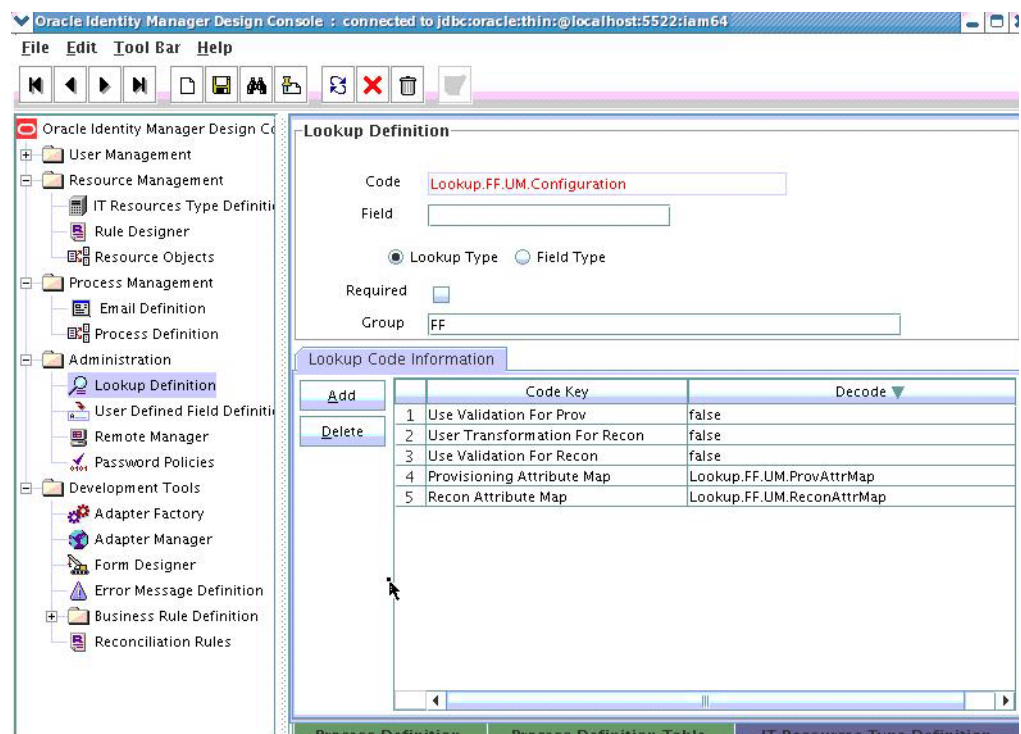
1. Log in to the Oracle Identity Manager Design Console.
2. Click Lookup Definition under Administration.

3. Create a new Lookup and add Lookup.FF.UM.Configuration as the Code.
4. Set the following attributes as illustrated in [Figure 17–4](#).

Note: This tutorial focuses on the minimum configurations needed to run an identity connector.

- **Provisioning Attribute Map** takes a value of Lookup.FF.UM.ProvAttrMap. This lookup contains the mapping between Oracle Identity Manager fields and identity connector attributes. The mapping is used during provisioning.
- **Reconciliation Attribute Map** takes a value of Lookup.FF.UM.ReconAttributeMap. This lookup contains the mapping between Oracle Identity Manager reconciliation fields and identity connector attributes. The mapping is used during reconciliation.

Figure 17–4 *Second Lookup Definition in Design Console*



17.2.3 Creating Provisioning Metadata

The following sections should be followed in order to configure Oracle Identity Manager for flat file provisioning.

- [Creating a Process Form](#)
- [Creating Adapters](#)
- [Creating A Process Definition](#)
- [Creating a Provisioning Attribute Mapping Lookup](#)

17.2.3.1 Creating a Process Form

A process form is used as the representation of object attributes on Oracle Identity Manager. This facilitates user input to set object attributes before passed to the connector bundle for an operation.

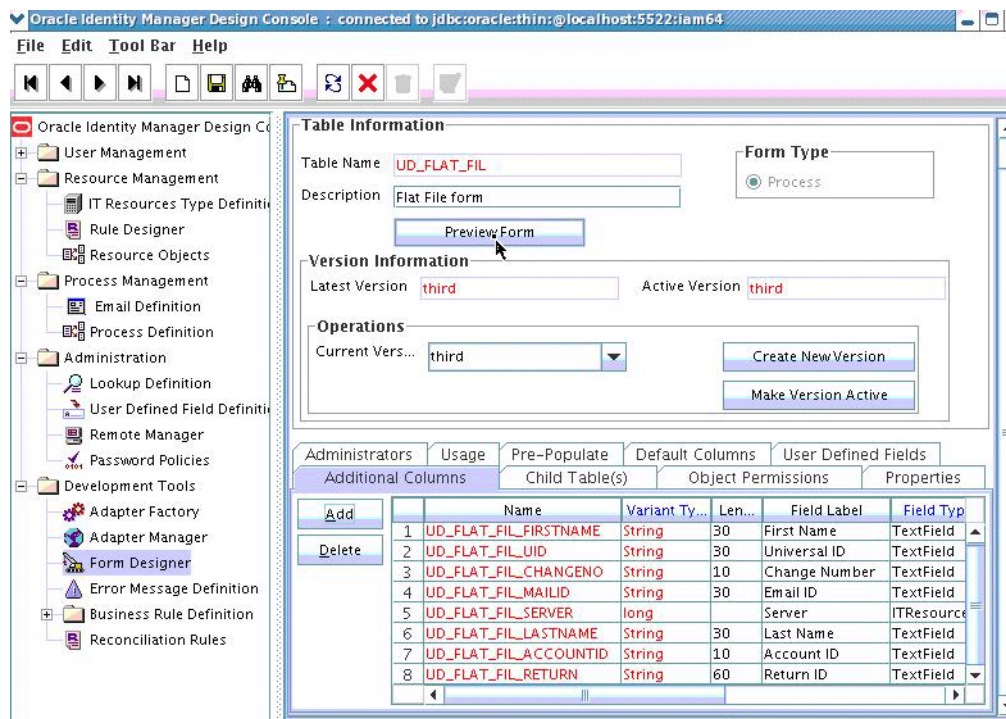
Attributes defined in the process form are not conventions. The form is a way to challenge the attributes that need to be passed to the identity connector. In general, define an attribute for each supported attribute in the identity connector.

Note: It is good practice to have a one to one mapping on the identity connector attributes.

There should be a field for querying the IT resource that should be associated with the respective IT Resource Type Definition. Variable type of each field should map to the type of the object attribute.

1. Log in to the Oracle Identity Manager Design Console.
2. Click Form Designer under Development Tools.
3. Create a new form with the Table Name UD_FLAT_FIL as illustrated in [Figure 17-5](#).

Figure 17-5 Form Designer in Design Console



4. Add the attributes defined in the connector schema, as listed in [Table 17-1](#).

Table 17-1 Form Designer Fields

Name	Variant	Field Label	Field Type
UD_FLAT_FIL_FIRSTNAME	String	First Name	TextField

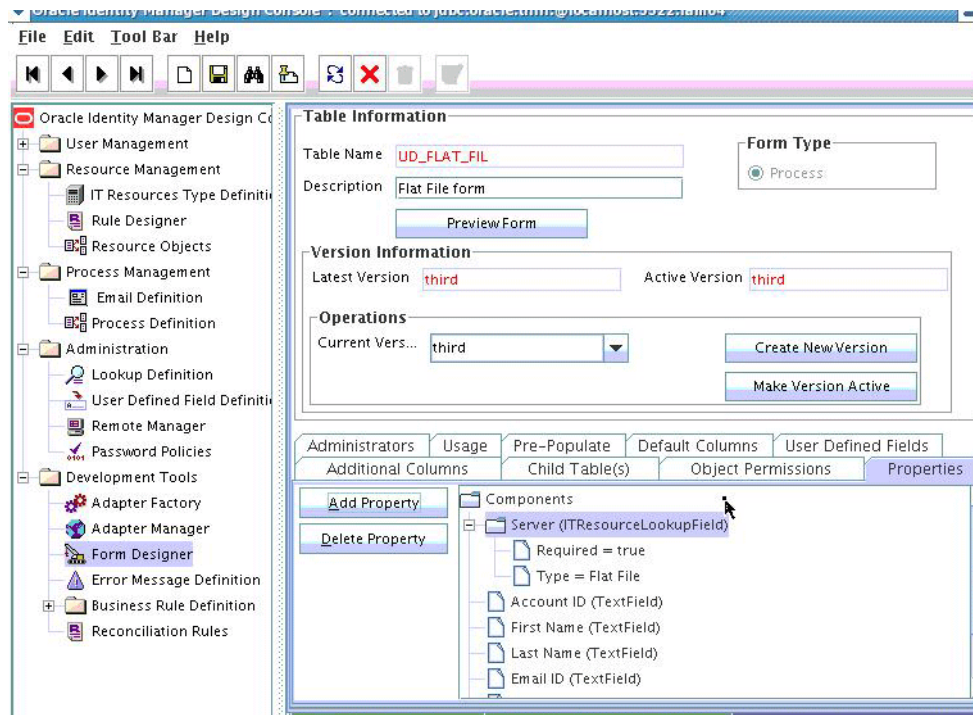
Table 17-1 (Cont.) Form Designer Fields

Name	Variant	Field Label	Field Type
UD_FLAT_FIL_UID	String	Universal ID	TextField
UD_FLAT_FIL_CHANGENO	String	Change Number	TextField
UD_FLAT_FIL_MAILID	String	Email ID	TextField
UD_FLAT_FIL_SERVER	long	Server	ITResource
UD_FLAT_FIL_LASTNAME	String	Last Name	TextField
UD_FLAT_FIL_ACCOUNTID	String	Account ID	TextField
UD_FLAT_FIL_RETURN	String	Return ID	TextField

Note: The flat file column names are FirstName, ChangeNo, EmailID, Server, LastName, and AccountID.

5. Click the Properties tab.
6. Add the following properties to Server(ITResourceLookupField) as illustrated in Figure 17-6.
 - Required = true
 - Type = Flat File

Figure 17-6 Properties of Form Designer in Design Console



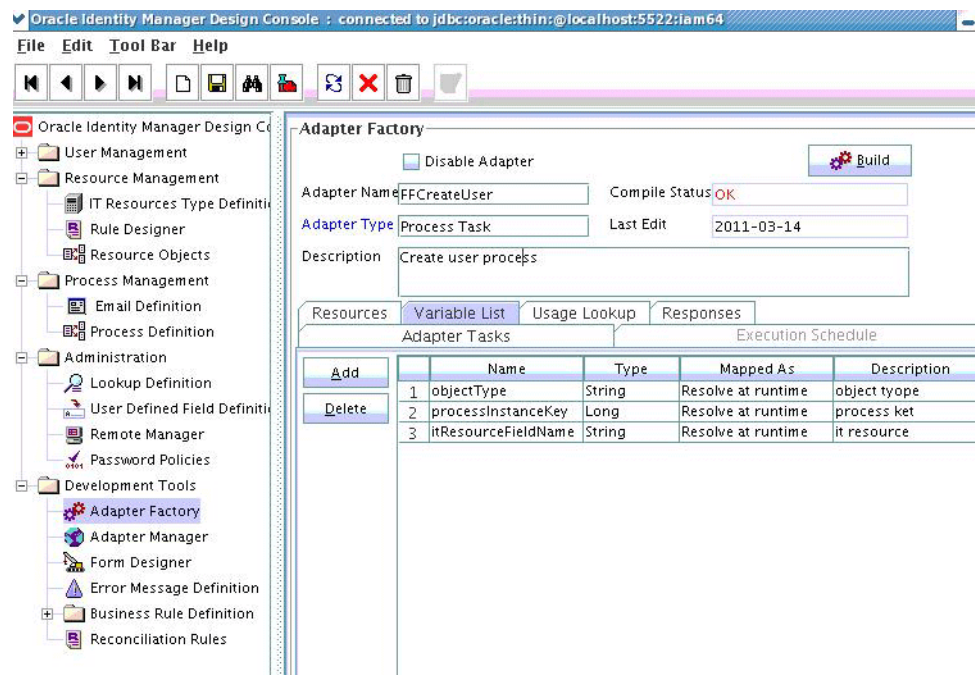
7. Save the form.
8. Click Make Version Active.

17.2.3.2 Creating Adapters

An adapter has to be created for all operations supported by the connector bundle, including Create, Update, and Delete.

1. Log in to the Oracle Identity Manager Design Console.
2. Click **Adapter Factory** under Development Tools.
3. Create a new adapter and add FFCreateUser as the Adapter Name.
4. Add Process Task as the Adapter Type.
5. Save the adapter.
6. Click the **Variable List** tab and add the following variables, as shown in [Figure 17-7](#).
 - objectType with Type String and Mapped as Resolve at runtime.
 - processInstanceKey with Type long and Mapped as Resolve at runtime.
 - itResourceFieldName with Type String and Mapped as Resolve at runtime.

Figure 17-7 Adapter Factory Variable List in Design Console

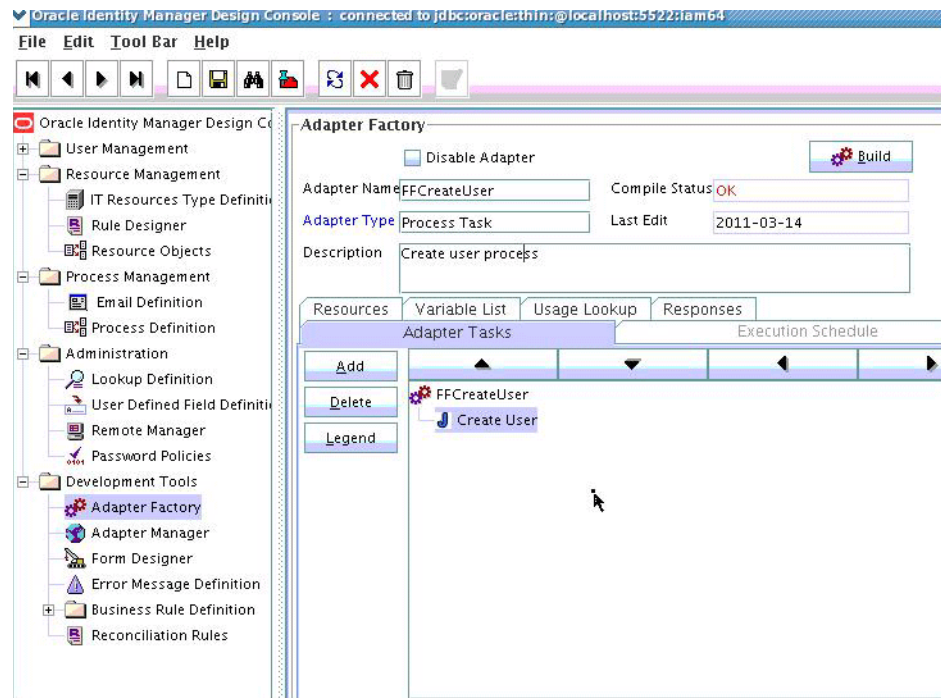


7. Add a Java functional task to the adapter by following this sub procedure, as shown in [Figure 17-8](#).
 - a. Click the Adapter Tasks tab.
 - b. Select the adapter and click Add.
 - c. Select Java from the task options.
 - d. Select **icf-oim-intg.jar** from the API source.
 - e. Select **oracle.iam.connetors.icfcommon.prov.ICProvisioninManager** as the API Source.
 - f. Select **createObject** as the method for the task.

- g. Save the configurations.
- h. Map the variables (previously added to the Variables List) against the appropriate method inputs and outputs.
- i. Map the configuration parameters against the appropriate method inputs and outputs.

Database Reference maps to Database Reference (Adapter References) and Return Variable maps to Return Variable (Adapter Variables).

Figure 17–8 Adapter Factory in Design Console



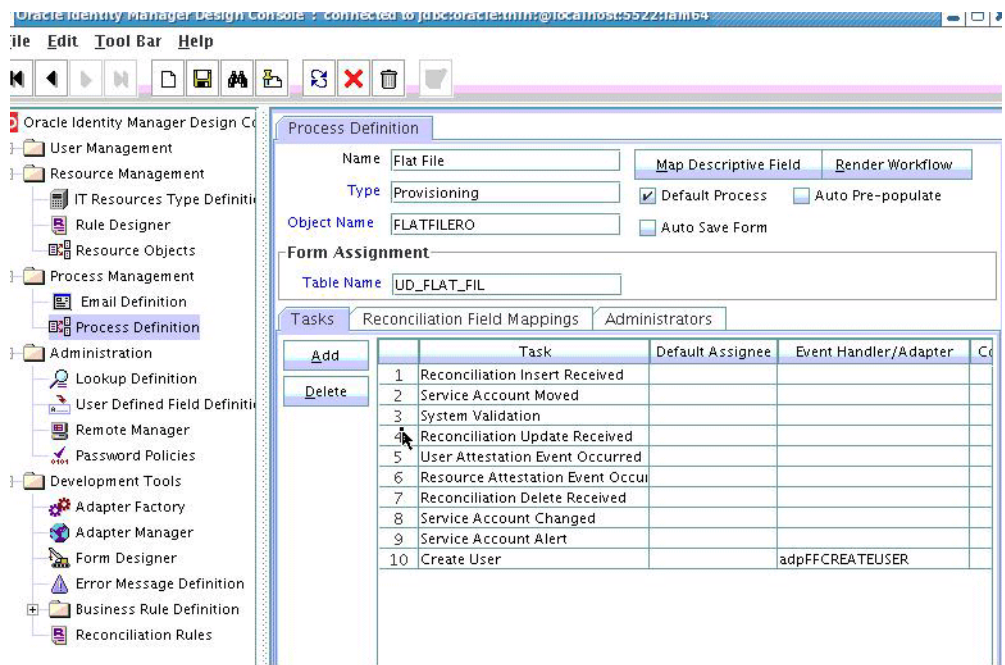
- 8. Save and build the adapter.

17.2.3.3 Creating A Process Definition

Process Definition defines the behavior of the connector bundle for a particular operation. Every operation has a corresponding task associated with it. This procedure will configure the process definition and integration of the process task for the Create operation.

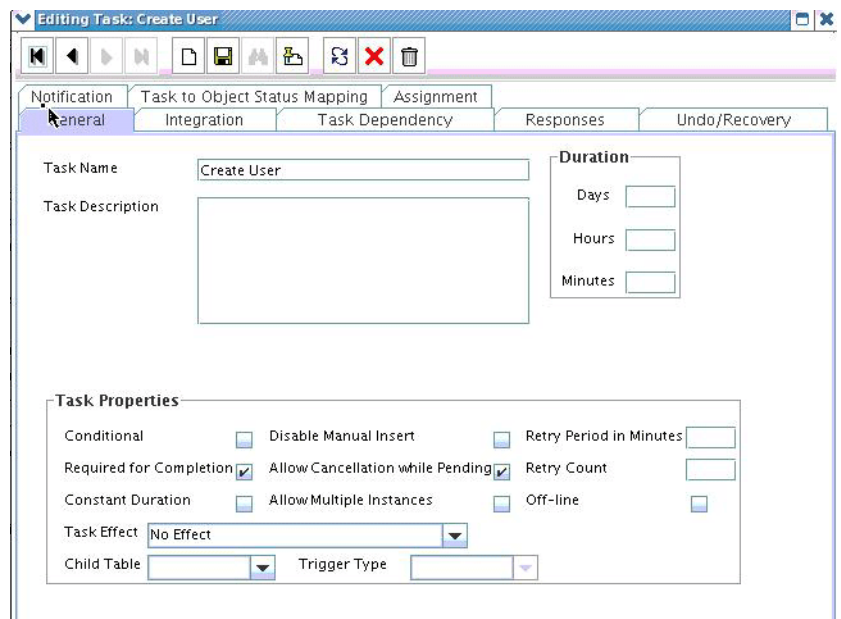
1. Log in to the Oracle Identity Manager Design Console.
2. Click Process Definition under the Process Management tab.
3. Create a new process definition and name it Flat File as illustrated in [Figure 17–9](#).

Figure 17–9 Process Definition in Design Console



4. Select Provisioning as the Type of process.
5. Provide the resource Object Name for the identity connector; in this example, FLATFILERO.
6. Provide the process form Table Name; in this example, UD_FLAT_FIL.
7. Add a process task and name it Create User.
8. Double click **Create User** to edit as illustrated in Figure 17–10.

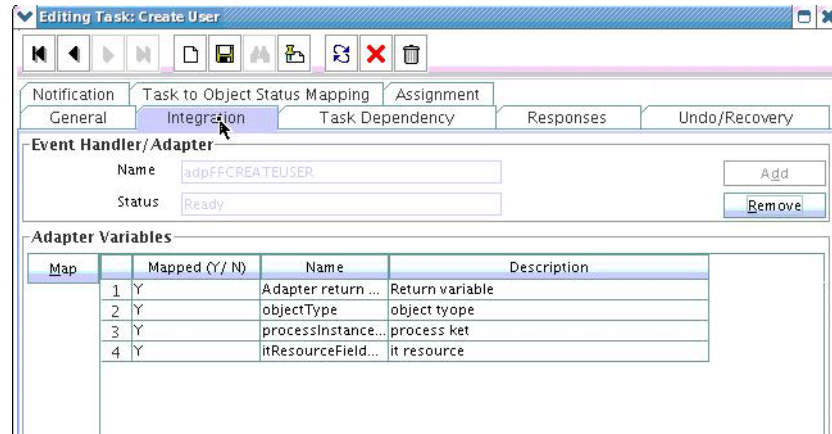
Figure 17–10 Editing Task Screen in Design Console



9. Click the **Integration** tab.
10. Click Add and select the FFCreateUser adapter from the list as illustrated in [Figure 17–11](#).

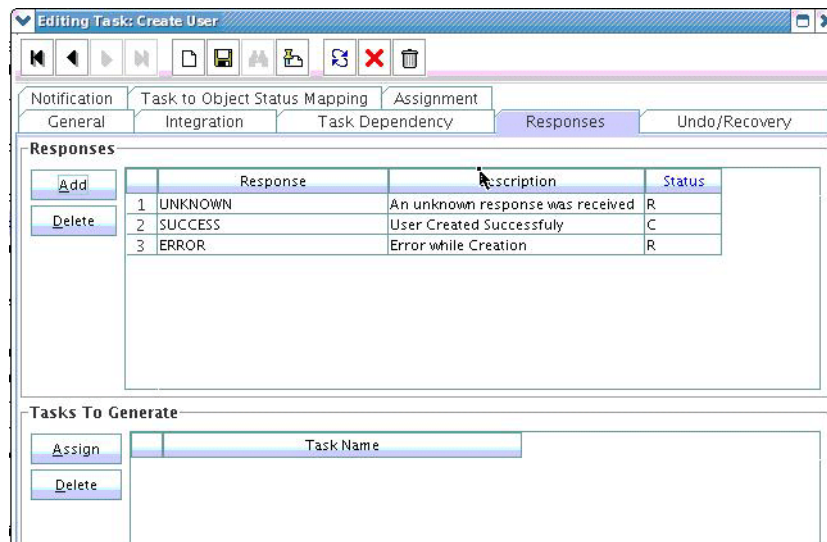
The adapter will be available only after it is compiled.

Figure 17–11 Integration Tab in Design Console



11. Map the variables as follows to set the response code returned by the identity connector.
 - Adapter Return Variable – Response Code
 - Object Type – [Literal:String] User (Name of the object type)
 - Process Instance Key – [Process Data] Process Instance
 - IT Resource Field Name – [Literal:String] UD_FLAT_FIL_SERVER (Form field name that contains the IT resource information)
12. Click the Responses tab and configure the responses as illustrated in [Figure 17–12](#).
 - UNKNOWN can be described as *Unknown response received* with a status of R (Rejected).
 - SUCCESS can be described as *Operation completed* with a status of C (Completed).
 - ERROR can be described as *Error occurred* with a status of R.

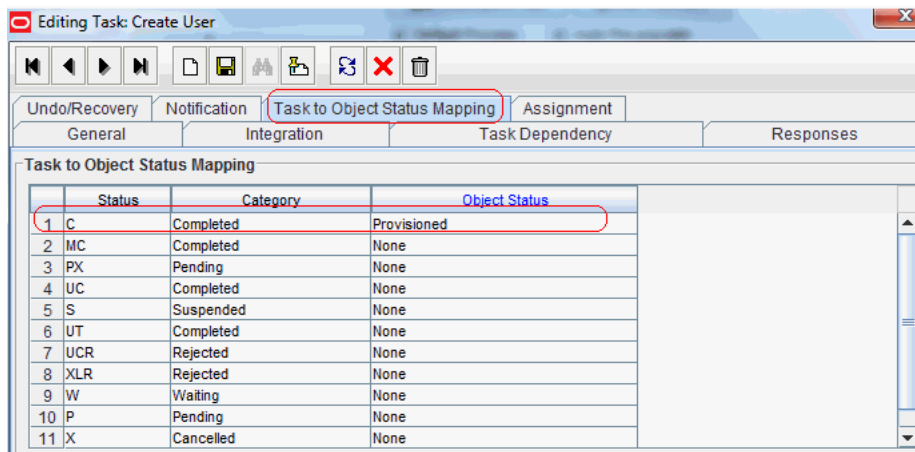
Figure 17–12 Configure Responses in Design Console



13. Click the **Task to Object Status Mapping** tab.

14. Update the Object Status to **Provisioned** for Status C, as shown in [Figure 17–13](#):

Figure 17–13 Task to Object Status Mapping



15. Save the process task.

17.2.3.4 Creating a Provisioning Attribute Mapping Lookup

Provisioning Attribute Mapping Lookup contains mappings of Oracle Identity Manager fields to identity connector bundle attributes. In the Provisioning Attribute Mapping Lookup:

- Code keys are Field Labels of the process form.
- Decodes are identity connector bundle attributes.
- Child form attributes can be configured as embedded objects in inputs.
- The identity connector's provisioning operation returns the UID in response. This can be set in a form field by coding it against the identity connector bundle attribute.

Following is the procedure to create a Provisioning Attribute Mapping Lookup.

1. Log in to the Oracle Identity Manager Design Console.
2. Click Lookup Definition under the Administration tab.
3. Create a new lookup and name it Lookup.FF.UM.ProvAttrMap.
The name of this lookup is referred from the object type configuration lookup. See [Section 17.2.2.3.2, "Creating Object Type Configuration Lookup."](#)
4. Add the form Field Labels as the code keys and identity connector bundle attributes as the decode.
 - Return ID : __UID__
 - Account ID: AccountId
 - Change Number: ChangeNumber
 - First Name: FirstName
 - Last Name: LastName
 - Email ID: MailId

17.2.3.4.1 Field Flags Used in the Provisioning Attributes Map

For provisioning attributes mapping, the following field flags can be appended to the code key:

- **LOOKUP:** This must be specified for all fields whose values are obtained by running a lookup reconciliation job. The values obtained from lookup reconciliation job have IT Resource Name/Key appended to it. Specifying this flag helps ICF integration to remove the appended value just before passing them onto the bundle. For example, the code key for a field with label Database whose value is obtained by running a lookup reconciliation job looks similar to Database[LOOKUP].

Note: The LOOKUP flag can be specified for both Provisioning and Reconciliation Attribute Map. For provisioning, IT Resource Name/IT Resource Key prefix must be removed. For reconciliation, IT Resource Name/IT Resource Key prefix must be added.

- **IGNORE:** This must be specified for all fields whose values are to be ignored and not sent to bundle. For example, the code key for a field with label Database whose value need not be sent to bundle looks similar to Database[IGNORE].
- **WRITEBACK:** This must be specified for all fields whose values need to be written back into the process form right after the create or update operation. Adding this flag makes the ICF integration layer call ICF Get API to get values of attributes marked with the WRITEBACK flag. For example, the code key for a field with label Database whose value needs to be written back to the process form right after create/update looks similar to Database[WRITEBACK]. For this to work, the connector must implement the GetApiOp interface and provide an implementation for the ConnectorObject getObject(ObjectClass objClass,Uid uid,OperationOptions options) API. This API searches the target for the account whose Uid is equal to the passed in Uid, and builds a connector object containing all the attributes (and their values) that are to be written back to process form.

Note: If the connector does not implement the GetApiOp interface, then the WRITEBACK flag does not work and an error is generated.

- **DATE:** This must be specified for fields whose type need to be considered as Date, without which the values are considered as normal strings. For example, the code key for a field with label Today whose value needs to be displayed in the date format looks similar to Today[DATE].
- **PROVIDEONPSWDCHANGE:** This must be specified for all fields that need to be provided to the bundle(target) when a password update happens. Some targets expect additional attributes to be specified on every password change. Specifying the PROVIDEONPSWDCHANGE flag, tells ICF integration to send all the extra fields or attributes whenever a password change is requested. For example, the code key for a field with label Extra Attribute Needed for Password Change whose value needs to be provided to bundle(target) while password update looks similar to Extra Attribute Needed for Password Change[PROVIDEONPSWDCHANGE].

17.2.4 Creating Reconciliation Metadata

This section contains the procedures to configure the reconciliation of records from the flat file. We will use the target reconciliation as an example; trusted reconciliation can also be configured in a similar fashion. Do the procedures in the listed order.

1. [Creating a Reconciliation Schedule Task](#)
2. [Creating a Reconciliation Profile](#)
3. [Setting a Reconciliation Action Rule](#)
4. [Creating Reconciliation Mapping](#)
5. [Defining a Reconciliation Matching Rule](#)

17.2.4.1 Creating a Reconciliation Schedule Task

By default, reconciliation uses a Search operation on the connector bundle. This operation is invoked with a schedule task configured using Oracle Identity Manager. This procedure is comprised of the following sub procedures.

1. [Defining the Schedule Task](#)
2. [Creating a Scheduled Task](#)

17.2.4.1.1 Defining the Schedule Task To define the scheduled task:

1. Create a Deployment Manager XML file containing the scheduled task details as shown in [Example 17-9](#). Make sure to update database value to your database.

Example 17-9 Deployment Manager XML with Scheduled Task Details

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<x1-ddm-data version="2.0.1.0" user="XELSYSADM"
database="jdbc:oracle:thin:@localhost:5524/estView.regress.rdbms.dev.mycompany.com
" exported-date="1307546406635" description="FF">
<scheduledTask repo-type="MDS" name="Flat File Connector User Reconciliation"
mds-path="/db" mds-file="Flat File Connector User Reconciliation.xml">
  <completeXml>
    <scheduledTasks xmlns="http://xmlns.oracle.com/oim/scheduler">
      <task>
```

```

        <name>Flat File Connector User Reconciliation</name>
        <class>oracle.iam.connectors.icfcommon.recon.SearchReconTask</class>
        <description>Flat File Connector User Reconciliation</description>
        <retry>0</retry>
        <parameters>
            <string-param required="false" encrypted="false"
helpText="Filter">Filter</string-param>
            <string-param required="false" encrypted="false"
helpText="Incremental Recon Date Attribute">Incremental Recon Date
Attribute</string-param>
            <string-param required="false" encrypted="false" helpText="IT
Resource Name">IT Resource Name</string-param>
            <string-param required="false" encrypted="false" helpText="Object
Type">Object Type</string-param>
            <string-param required="false" encrypted="false" helpText="Latest
Token">Latest Token</string-param>
            <string-param required="false" encrypted="false" helpText="Resource
Object Name">Resource Object Name</string-param>
        </parameters>
    </task>
</scheduledTasks>
</completeXml>
</scheduledTask>
</xl-ddm-data>

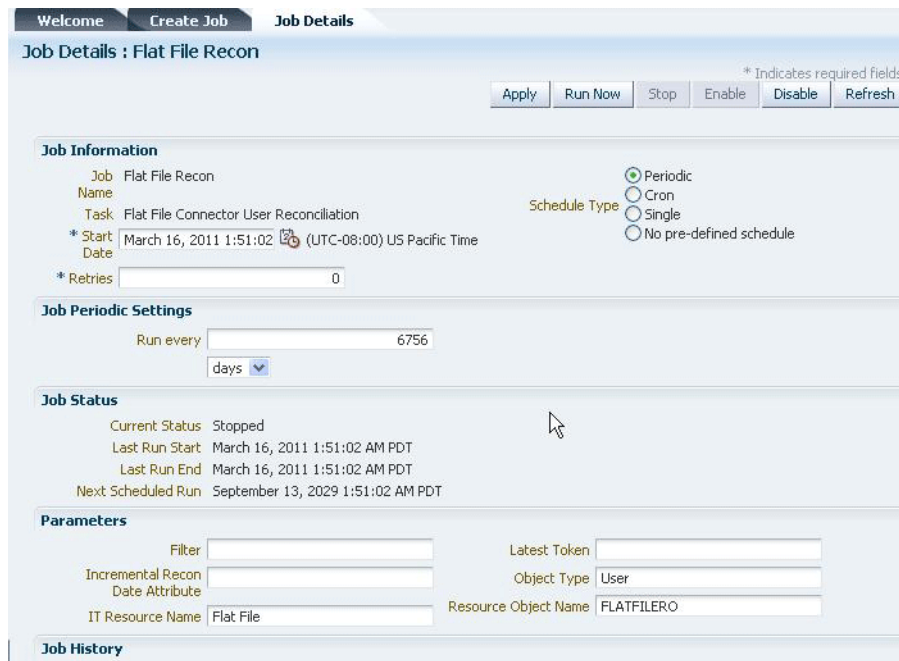
```

2. Save the file as Flat File Connector User Reconciliation.xml.
3. Login into the Administrative and User Console. Click **Import Deployment Manager File**.
4. Select the Flat File Connector User Reconciliation.xml file, and click **Import**.
5. Complete the steps in the wizard.

17.2.4.1.2 Creating a Scheduled Task This procedure explains how to create a scheduled task.

1. Log in to the Oracle Identity Manager Advanced Administration.
2. Click **Scheduler** under the System Management tab.
3. Add a schedule task and add Flat File Connector User Reconciliation as the type as illustrated in [Figure 17-14](#).

Figure 17–14 Schedule Task Screen in Advanced Console

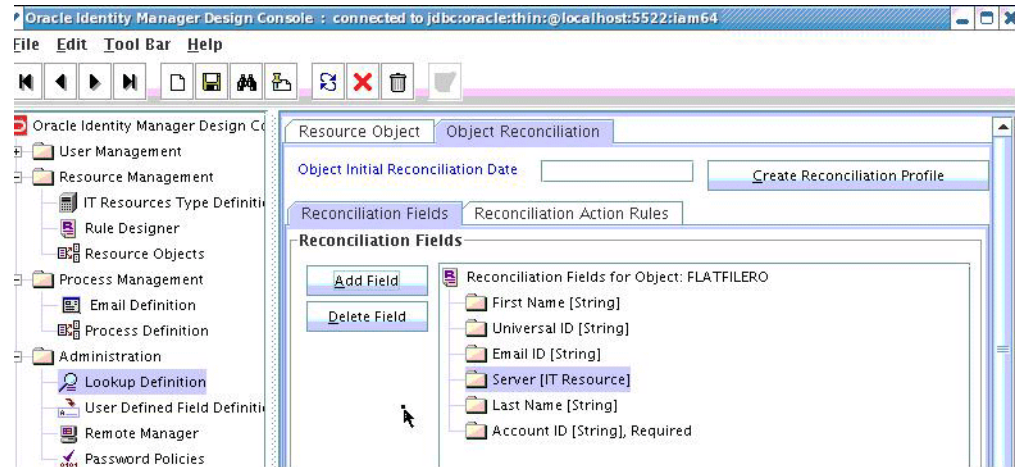


4. Set the parameters as follows:
 - IT Resource Name takes a value of Flat File.
 - Resource Object Name takes a value of FLATFILERO.
 - Object Type takes a value of User.
5. Click **Apply**.

17.2.4.2 Creating a Reconciliation Profile

A reconciliation profile defines the structure of the object attributes while reconciliation. The reconciliation profile should contain all the attributes that have reconciliation support.

1. Log in to the Oracle Identity Manager Design Console.
2. Click **Resource Objects** under Resource Management.
3. Open the FLATFILERO resource object.
4. Click the **Object Reconciliation** tab as illustrated in [Figure 17–15](#).

Figure 17–15 Object Reconciliation in Design Console

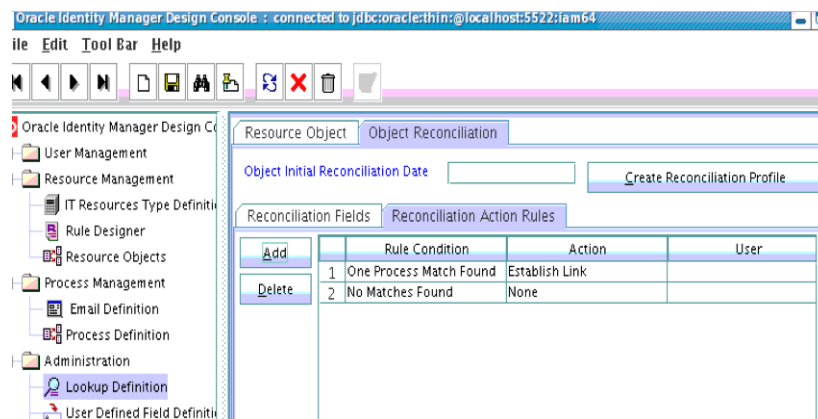
5. Add following reconciliation fields:
 - First Name [String]
 - Universal ID [String]
 - Email ID [String]
 - IT Resource Name [String]
 - Last Name [String]
 - Account ID [String], Required
6. Save the configuration.

17.2.4.3 Setting a Reconciliation Action Rule

A Reconciliation Action Rule defines the behavior of reconciliation. In this procedure, define the expected action when a match is found. This procedure assumes you are logged into the Oracle Identity Manager Design Console.

1. Open the FLATFILERO resource object.
2. Click the **Object Reconciliation** tab.
3. Click the **Reconciliation Action Rules** tab in the right frame.

Figure 17–16 Reconciliation Action Rules in Design Console

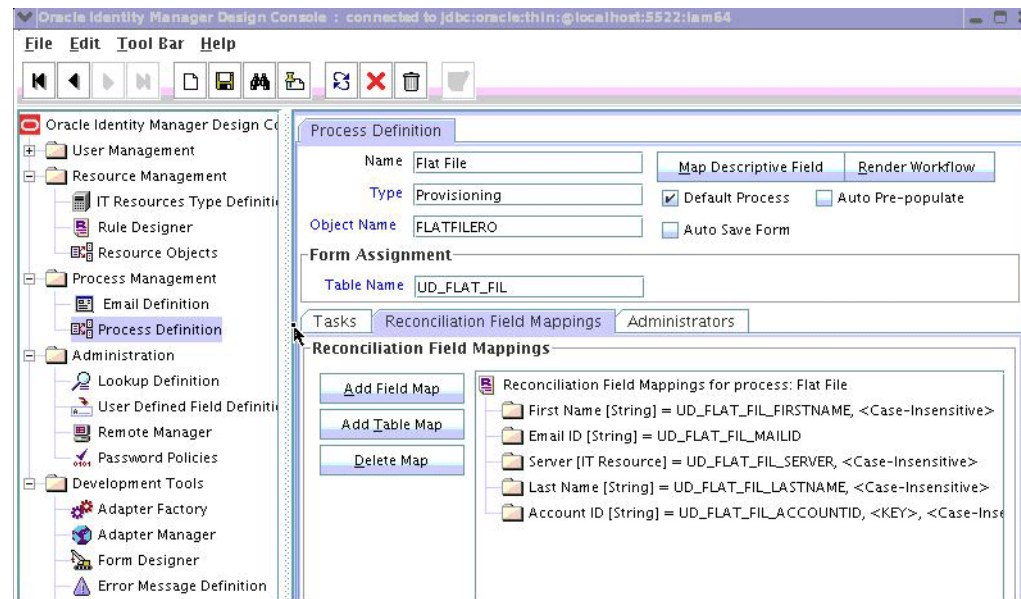


4. Add an action rule defined as One Process Match Found (Rule Condition) and Establish Link (Action).
5. Add an action rule defined as One Entity Match Found (Rule Condition) and Establish Link (Action).
6. Click **Create Reconciliation Profile**.
7. Click **Save**.

17.2.4.4 Creating Reconciliation Mapping

The reconciliation mapping has to be done in the process definition. This is to map the supported reconciliation fields (from resource object) to the process form fields. This mapping is needed only for configuring target reconciliation.

1. Log in to the Oracle Identity Manager Design Console.
2. Click Process Definition under Process Management.
3. Open the Flat File process definition.
4. Click the Reconciliation Field Mappings tab as illustrated in [Figure 17–17](#).

Figure 17–17 Reconciliation Field Mapping in Design Console


5. Add mappings between the reconciliation profile fields and the process form fields.

- First Name[String] = UD_FLAT_FIL_FIRSTNAME
- Email ID[String] = UD_FLAT_FIL_MAILID
- IT Resource Name[String] = UD_FLAT_FIL_SERVER
- Last Name[String] = UD_FLAT_FIL_LASTNAME
- Account ID [String] = UD_FLAT_FIL_ACCOUNTID <KEY>
<KEY> sets Account ID as a key field.

6. Save the configuration.

17.2.4.4.1 Field Flags Used in the Reconciliation Attributes Map

For reconciliation attributes mapping, the following field flags can be appended to the code key:

- **TRUSTED:** This must be specified in the Recon Attribute Map for the field that represents the status of the account. This flag must be specified only for trusted reconciliation. If this is specified, then the status of the account is either Active or Disabled. Otherwise, the status is either Enabled or Disabled. For example, the code key for a field with label Status whose value needs to be either Active/Disabled must look similar to Status[TRUSTED].
- **DATE:** In Recon Attribute Map, this must be specified for fields whose type need to be considered as Date. For example, the code key for a field with label Today whose value needs to be displayed in the date format must look similar to Today[DATE].

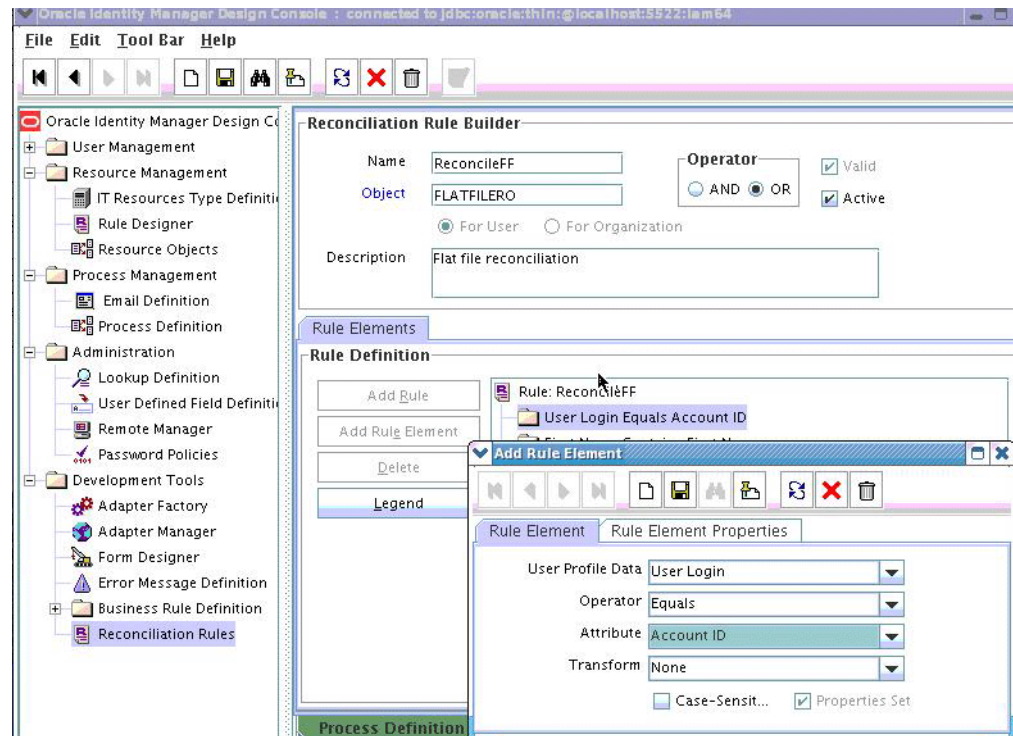
17.2.4.5 Defining a Reconciliation Matching Rule

A reconciliation matching rule defines the equation for calculating the user match.

1. Log in to the Oracle Identity Manager Design Console.

2. Open the Reconciliation Rules form under Development Tools.
3. Click **Add Rule**.

Figure 17–18 Adding Reconciliation Matching Rule



4. Select resource object FLATFILERO.
5. Save and add the rule element.
User Login from the user profile data equals the Account ID resource attribute.
6. Save the rule.

17.3 Provisioning a Flat File Account

The flat file connector is ready to work so now the user needs to log in to Oracle Identity Manager and create an IT resource (target) using the following procedure.

- Create IT resource of type "Flat File".
- Provide the IT resource parameters as appropriate.
- Provide the configuration parameters in Lookup.FF.Configuration as appropriate.

Part IV

Generic Technology Connectors

This part explains how to develop and manage connectors.

It contains the following chapters:

- [Chapter 18, "Understanding Generic Technology Connectors"](#)
- [Chapter 19, "Predefined Providers for Generic Technology Connectors"](#)
- [Chapter 20, "Creating Custom Providers for Generic Technology Connectors"](#)
- [Chapter 21, "Creating and Managing Generic Technology Connectors"](#)
- [Chapter 22, "Troubleshooting Generic Technology Connectors"](#)

Understanding Generic Technology Connectors

This chapter introduces generic technology connectors and the features that Oracle Identity Manager provides for working with generic technology connectors.

This chapter contains the following sections:

- [Requirement for Generic Technology Connectors](#)
- [Functional Architecture of Generic Technology Connectors](#)
- [Features of Generic Technology Connectors](#)
- [Connector Objects Created by the Generic Technology Connector Framework](#)
- [Roadmap for Information on Generic Technology Connectors in This Guide](#)

18.1 Requirement for Generic Technology Connectors

Predefined Oracle Identity Manager connectors are designed for commonly used target systems such as Microsoft Active Directory and PeopleSoft Enterprise Applications. A predefined connector is developed using the Adapter Factory approach, and its architecture is based on either the APIs that the target system supports or the data repository type and schema in which the target system stores user data.

Since they are developed using the Adapter Factory, predefined connectors offer extensive workflow and adapter customization capabilities. The use of a predefined connector is the recommended integration method when such a connector is available for the target system.

There may be scenarios in which you want to integrate Oracle Identity Manager with a target system that has no corresponding predefined connector. The following are examples of such scenarios:

Scenario 1: All employees of Acme Inc. are allotted disk space on a backup server. Employees send requests to the system administrator for managing their accounts on the backup server. The system administrator has developed a Web-based application to capture, review, and act on requests from employees. The front end of this application is a Web service that accepts and stores data in CSV format. Employee account data stored in the back end can be exported as XML files to a specified location.

Scenario 2: Ceeam Travels Inc. owns a custom Web-based application that its customers use to request airline fare quotes. Agents, who are also employees of Ceeam Travels, respond to these requests by using the same application. Customers register

themselves to create accounts in this application. However, Ceeam Travels employees need to have accounts auto-provisioned based on their HR job title. Account management functions (such as create, update, and delete) of the application are available through Java APIs.

In both these scenarios, you need to create a custom connector to link the target system and Oracle Identity Manager. If you are looking for a simple way to create your custom connector and do not need the customization features of the Adapter Factory, you can create the connector by using the Generic Technology Connector feature of Oracle Identity Manager. As described in the [Section 18.2, "Functional Architecture of Generic Technology Connectors"](#), providers are the building blocks of generic technology connectors.

- In Scenario 1, you can use the predefined shared drive reconciliation transport provider and CSV reconciliation format provider to create a generic technology connector that reconciles data stored in a flat file into Oracle Identity Manager.
- In Scenario 2, there is no predefined provider available to integrate the custom application with Oracle Identity Manager. In this case, you can use the instructions provided in [Chapter 20, "Creating Custom Providers for Generic Technology Connectors"](#) to create the custom providers that call the Java APIs exposed by the target application.

18.2 Functional Architecture of Generic Technology Connectors

Like a predefined connector, a generic technology connector acts as the bridge for reconciliation and provisioning operations between Oracle Identity Manager and a target system. Functionally, a generic technology connector can be divided into a reconciliation module and provisioning module. When you create a generic technology connector, you can specify whether you want to include both modules, or include the reconciliation module only, or include the provisioning module only.

A predefined connector provides reconciliation and provisioning functionality in the context of the same target. In contrast, the reconciliation and provisioning modules of a generic technology connector are composed of reusable components that you choose. Each component performs a specific function during provisioning or reconciliation. For example, you can create a connector that performs trusted source reconciliation from flat files and provides target resource provisioning using the SPML protocol to an SPML-enabled target.

In this guide, the components that constitute a generic technology connector are called **providers**.

Each provider performs a transport, format change, validation, or transformation function on the data that it receives as input. In other words, data items processed by a provider are moved to a new location, validated against specified criteria, or undergo modification in structure or value. In this guide, the term **data sets** is used to describe data structures arranged in the form of layers, with data flowing from one layer to another during provisioning and reconciliation.

While creating a generic technology connector, you can specify the fields (user identity metadata) that must be included in each data set. You can also define mappings between fields of different data sets. A mapping serves one of the following purposes:

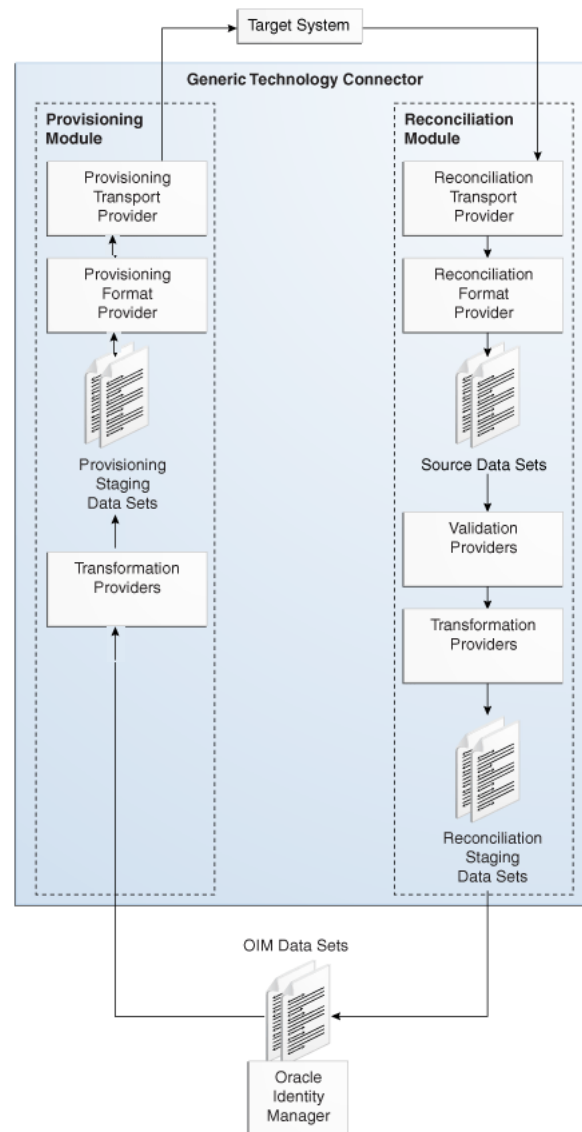
- Establishes a data flow path between fields of two data sets for use either in provisioning or reconciliation.

A mapping of this type forms the basis for validations or transformations to be performed on data that is fetched from the target system.

- Creates a basis for comparing (matching) field values of two data sets.

Figure 18–1 shows the functional architecture of a generic technology connector.

Figure 18–1 Functional Architecture of a Generic Technology Connector



The following sections describe the providers and data sets that constitute a generic technology connector:

- [Providers and Data Sets of the Reconciliation Module](#)
- [Providers and Data Sets of the Provisioning Module](#)
- [Oracle Identity Manager Data Sets](#)

18.2.1 Providers and Data Sets of the Reconciliation Module

The reconciliation module consists of the following providers and data sets:

- Reconciliation Transport Provider

A reconciliation transport provider carries reconciliation data from the target system to Oracle Identity Manager. The manner in which this provider carries the reconciliation data depends on the implementation of the provider. For example, a reconciliation transport provider can read data from a file, or accept data from a Web service, or query a database.

- Reconciliation Format Provider

A reconciliation format provider parses the reconciliation data fetched by the reconciliation transport provider and converts this data into data structures that can be stored in Oracle Identity Manager.

- Source

A Source data set holds the data processed by the reconciliation format provider. This data set can have child data sets.

- Validation Provider

A validation provider checks the data in the source data sets against criteria you specify before passing the data to the reconciliation engine of Oracle Identity Manager.

Note: You can include more than one validation provider in a generic technology connector.

- Transformation Provider

A transformation provider included in the reconciliation module modifies data received from the validation providers before passing on the data for the creation of reconciliation events in Oracle Identity Manager.

The following is an example of a transformation provider function:

Suppose the following are the values of two fields in the target system

First Name: John

Last Name: Doe

A transformation provider can be used to create the following reconciliation field output:

Login ID: John.Doe

- Reconciliation Staging

A reconciliation staging data set holds user data that has been processed by the validation providers and transformation providers. This data set can have child data sets.

18.2.2 Providers and Data Sets of the Provisioning Module

The provisioning module consists of the following providers and data sets:

- Transformation Provider

A transformation provider can be used to modify data items at the following stages:

- A transformation provider included in the provisioning module modifies data entered in Oracle Identity Manager process forms before the data is sent to the target system.

- **Provisioning Staging**
A provisioning staging data set holds user data before it is sent to the provisioning format provider. This data is the output of the transformation functions that are run on the user data for a trusted source or account data for a target system, which are stored in Oracle Identity Manager. This data set can have child data sets.
- **Provisioning Format Provider**
A provisioning format provider converts Oracle Identity Manager provisioning data (received from the transformation provider) into a format that is supported by the target system.
- **Provisioning Transport Provider**
A provisioning transport provider carries provisioning data from the provisioning format provider to the target system. The manner in which this provider carries reconciliation data depends on the implementation of the provider. For example, a provider can copy data into a file, or send data to a Web service, or post data to a database.

18.2.3 Oracle Identity Manager Data Sets

The Oracle Identity Manager data sets represent data that is stored in Oracle Identity Manager. Although these data sets are not part of the reconciliation or provisioning module, they are considered part of the generic technology connector because you can add fields to these data sets and create mappings between fields of these data sets and other data sets. The following are the Oracle Identity Manager data sets:

- **OIM - User**
The OIM - User data set holds the metadata (set of identity fields) that defines the Oracle Identity Manager User. In trusted source reconciliation, this data set receives newly created or modified user account information from the reconciliation staging data set. In target resource reconciliation, the fields of the OIM - User data set can be used to establish a match between target system user accounts and existing Oracle Identity Manager users. This data set does not have child data sets.
- **OIM - Account**
The OIM - Account data set holds the user account information that is stored in the process form fields of Oracle Identity Manager. This user account information is received from the reconciliation staging data sets. The OIM - Account data set can have child data sets.

18.3 Features of Generic Technology Connectors

The following sections discuss the features of generic technology connectors:

- [Features Specific to the Reconciliation Module](#)
- [Other Features](#)

18.3.1 Features Specific to the Reconciliation Module

The following features are specific to the reconciliation module:

- [Trusted Source Reconciliation](#)
- [Account Status Reconciliation](#)

- [Full and Incremental Reconciliation](#)
- [Batched Reconciliation](#)
- [Reconciliation of Multivalued Attribute Data \(Child Data\) Deletion](#)
- [Failure Threshold for Stopping Reconciliation](#)

18.3.1.1 Trusted Source Reconciliation

A generic technology connector can be used for trusted source reconciliation. During reconciliation in trusted mode:

- If the reconciliation engine detects new target system accounts, it creates corresponding Oracle Identity Manager users.
- If the reconciliation engine detects changes to existing target system accounts, the same changes are made in the corresponding Oracle Identity Manager users.

Note: While creating a generic technology connector, if you do not select the Trusted Source reconciliation option, target resource reconciliation is enabled. In target resource reconciliation, only modifications to target system accounts are reconciled. New target system accounts detected during reconciliation are *not* created automatically in Oracle Identity Manager.

A generic technology connector that is used for trusted source reconciliation cannot be used for provisioning. This design feature was incorporated to ensure that you do not create or modify through Oracle Identity Manager user account information on a target system that is designated as a trusted source.

Connector objects, such as IT resources and resource objects, are created automatically at the end of the generic technology connector creation process. By default, the resource object of a generic technology connector is a trusted resource object. In other words, a generic technology connector is already compatible with the Multiple Trusted Source reconciliation feature. This feature is discussed in [Chapter 12, "Developing Provisioning Processes"](#).

Note: In trusted source reconciliation, the reconciliation of multivalued (child) data is not supported.

18.3.1.2 Account Status Reconciliation

User account status information is used to track whether or not the owner of a target system account is to be allowed to access and use the account. If the target system does not store account status information in the format in which it is stored in Oracle Identity Manager, you can use the predefined translation transformation provider to implement account status reconciliation.

Note:

User account status reconciliation can be implemented independently of whether you select trusted source or target resource reconciliation.

The Design Console offers features for implementing account status reconciliation, without using the translation transformation provider. For more information, see [Section 12.3.2.2, "Reconciliation Field Mappings Tab"](#).

18.3.1.3 Full and Incremental Reconciliation

While creating a generic technology connector, you can specify that you want to use the connector for full or incremental reconciliation.

You select incremental reconciliation if the target system supports a method for the reconciliation engine to identify records that have changed since the last reconciliation run. For example, if the target system time stamps the creation of or changes made to user records, the reconciliation engine can identify records that have been added or modified since the last reconciliation run. In incremental reconciliation, only target system records that have changed after the last reconciliation run are reconciled (stored) into Oracle Identity Manager.

You select full reconciliation if any one of the following conditions is true:

- The target system does not support any method for the reconciliation engine to identify records that have changed since the last reconciliation run.
- You want to perform first-time reconciliation of all user account records in the target system.

In full reconciliation, all the reconciliation records are extracted from the target system. However, the optimized reconciliation feature identifies and ignores records that have already been reconciled in Oracle Identity Manager. This helps reduce the space occupied by reconciliation data. If this feature were not present, the amount of data stored in the Oracle Identity Manager database would increase rapidly with each reconciliation run.

Note: The outcome of both full and incremental reconciliation is the same:

- All the target system records are reconciled during the first reconciliation run.
 - From the second reconciliation run onward, target system records that are created or updated after the last reconciliation run are reconciled into Oracle Identity Manager.
-

18.3.1.4 Batched Reconciliation

You can specify a batch size for reconciliation. By doing this, you can break into batches the total number of records that the reconciliation engine fetches from the target system during each reconciliation run. This feature provides more control over the reconciliation process.

18.3.1.5 Reconciliation of Multivalued Attribute Data (Child Data) Deletion

You can specify whether or not you want to reconcile into Oracle Identity Manager the deletion of multivalued attribute data on the target system.

Note: Generic technology connectors do not support the reconciliation of parent data deletion. For example, if the account of user `John Doe` is deleted from the target system, you cannot use a generic technology connector to reconcile this user account deletion into Oracle Identity Manager.

18.3.1.6 Failure Threshold for Stopping Reconciliation

During reconciliation, validation providers can be used to run checks on target system data before it is stored in Oracle Identity Manager. You can set a failure threshold to automatically stop a reconciliation run if the percentage of records that fail the validation checks to the total number of records processed exceeds the specified threshold percentage.

18.3.2 Other Features

The following features are not specific to the reconciliation or provisioning module:

- [Custom Data Fields and Field Mappings](#)
- [Custom Providers](#)
- [Multilanguage Support](#)
- [Custom Date Formats](#)
- [Propagation of Changes in Oracle Identity Manager User Attributes to Target Systems](#)

18.3.2.1 Custom Data Fields and Field Mappings

While creating a generic technology connector, you can specify the identity fields and field mappings (data flow paths) that must be used during reconciliation and provisioning.

18.3.2.2 Custom Providers

You can create custom providers if the predefined providers shipped with Oracle Identity Manager do not address the transport, format change, validation, or transformation requirements of your operating environment.

18.3.2.3 Multilanguage Support

Generic technology connectors can handle both ASCII and non-ASCII data (multibyte characters), which represent a user, an account, or some other type of provisioned resource object.

18.3.2.4 Custom Date Formats

While creating a generic technology connector, you can specify:

- The format of date values in target system records that are extracted during reconciliation
- The format in which date values must be sent to the target system during provisioning

18.3.2.5 Propagation of Changes in Oracle Identity Manager User Attributes to Target Systems

While creating a generic technology connector, you can enable the automatic propagation of changes in Oracle Identity Manager User attributes to the target system.

18.4 Connector Objects Created by the Generic Technology Connector Framework

The list of connector objects created by the generic technology connector framework depends on the combination of the reconciliation and provisioning options that you select on the Step 1: Basic Information page:

- [Both Reconciliation and Provisioning Are Selected](#)
- [Only Reconciliation Is Selected](#)
- [Only Provisioning Is Selected](#)

Note: Except for the form names, the names of the generic technology connector objects are in the *GTC_NAME_GTC* format, where *GTC_NAME* is the name that you assign to the connector.

For example, if you specify `DBTables_conn` as the name of a generic technology connector that you create, all the connector objects (except the forms) are named `DBTables_conn_GTC`.

18.4.1 Both Reconciliation and Provisioning Are Selected

The following objects are created when you select both the provisioning and reconciliation options on the Step 1: Basic Information page:

- IT resource type

The parameters of the IT resource type are the run-time parameters of the format and transport providers (for both reconciliation and provisioning) that you select on the first page.
- IT resource

The IT resource is an instance of the IT resource type. It contains the run-time parameter values of the providers.
- Resource object

The resource object holds the values of the fields that constitute the reconciliation staging parent data set. For each reconciliation staging child data set, multilevel reconciliation fields (with corresponding child fields as their attributes) are created automatically.

Note: When you select the trusted source reconciliation option, a trusted resource object is one of the objects created automatically at the end of the connector creation process.

- Parent and child forms

Parent and child forms are based on the OIM - Account data set and its child data sets, respectively. By default, the names of the forms are the same as the names of their corresponding data sets. On the Step 3: Verify Form Names page, you can change the form names as required.

- Process definition

The process definition contains the reconciliation field mappings and the system-defined and provisioning-specific process tasks. See [Section 21.2.6, "Configuring Provisioning"](#) for information about the process tasks that are included in the process definition.
- Generic adapter

The generic adapter contains the code for all the provisioning functions that a generic technology connector performs.
- Scheduled task

During a reconciliation run, the scheduled task triggers the reconciliation processes in the predefined sequence. [Section 21.2.5, "Configuring Reconciliation"](#) provides information about setting up the scheduled task.
- Reconciliation rule

The reconciliation rule consists of rule elements. A single rule element represents a mapping created between a field of the reconciliation staging data set and a field of the OIM - User data set.
- Action rules

Any one of the following default action rules are created for target resource reconciliation:

Rule Condition	Action
One Entity Match Found	Establish Link
One Process Match Found	Establish Link

Any one of the following default action rules are created for trusted source reconciliation:

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

The user group to which the creator of the generic technology connector belongs is made the administrator of the following connector objects that are created automatically during the generic technology connector creation process:

- IT resource
- Resource object (Administrator and Object Authorizer)
- All forms
- Process definition
- Reconciliation fields

- Reconciliation field mappings

18.4.2 Only Reconciliation Is Selected

See ["Both Reconciliation and Provisioning Are Selected"](#) on page 18-9 for the list of objects that are created when you select both the Reconciliation and Provisioning options. From that list, the following objects are *not* created when you select only the Reconciliation option on the Step 1: Basic Information page:

- Generic adapter.
- Provisioning-specific process tasks.

However, the process definition itself and its constituent system-defined process tasks are created.

18.4.3 Only Provisioning Is Selected

See ["Both Reconciliation and Provisioning Are Selected"](#) on page 18-9 for the list of objects that are created when you select both the Reconciliation and Provisioning options. From that list, the following objects are *not* created when you select only the Provisioning option on the Step 1: Basic Information page:

- Scheduled task
- Reconciliation rule
- Reconciliation fields
- Reconciliation field mappings

18.5 Roadmap for Information on Generic Technology Connectors in This Guide

The following is an overview of the remaining chapters and appendixes on generic technology connectors:

- [Chapter 19, "Predefined Providers for Generic Technology Connectors"](#) provides a survey of available providers, which include the shared drive reconciliation transport provider, CSV reconciliation format provider, SPML provisioning format provider, Web Services provisioning transport provider, transformation provider, and validation provider.
- [Chapter 20, "Creating Custom Providers for Generic Technology Connectors"](#) explains the role of providers during provisioning and reconciliation, and describes how to create custom providers.
- [Chapter 21, "Creating and Managing Generic Technology Connectors"](#) describes how to create and maintain Generic Technology Connectors, and how to use the generic Connection Pool Framework in custom connectors.
- [Chapter 22, "Troubleshooting Generic Technology Connectors"](#) describes general and configuration issues related to Generic Technology Connectors and how to troubleshoot the issues.

Predefined Providers for Generic Technology Connectors

The following predefined providers are shipped with the current release of Oracle Identity Manager:

See Also: "Integration Solutions" in the *Oracle Fusion Middleware User's Guide for Oracle Identity Manager*

Note: You must determine the values of parameters for providers that you decide to use. You would need to use these values while creating the generic technology connector by using the Administrative and User Console.

- [Shared Drive Reconciliation Transport Provider](#)
- [CSV Reconciliation Format Provider](#)
- [SPML Provisioning Format Provider](#)
- [Web Services Provisioning Transport Provider](#)
- [Transformation Providers](#)
- [Validation Providers](#)

19.1 Shared Drive Reconciliation Transport Provider

The shared drive reconciliation transport provider reads data from flat files stored in staging directories and moves the files to an archiving directory. The staging and archiving directories must be shared for access from the Oracle Identity Manager server.

The following are parameters of this provider:

- **Staging Directory (Parent identity data)**
Use this parameter to specify the path of the directory in which files containing parent data are stored. It is mandatory to specify a value for this parameter. This is a run-time parameter.

In this guide, **parent data** means the user account information that is stored in the target system.

Sample value for this parameter:

T:/TargetSystemDirectory/ParentData

Note: If the staging directory is not on the server on which Oracle Identity Manager is installed, it must be shared and mapped as a network drive on the Oracle Identity Manager server.

Data stored in the parent data files must conform to the following conventions:

- First line of the file

The first line of the parent data file must be the file header that describes the contents of the file.

The file header can be preceded by any number of lines that begin with the hash-mark or pound-sign (#). These are ignored while the file is read. However, you must ensure that there are no spaces at the start of the header. If you are using a language other than English, you must not enter non-ASCII characters on this line.

Note: There are no checks to stop you from entering non-ASCII characters on the first line. In addition, the generic technology connector framework can parse such characters. However, the use of non-ASCII characters would result in problems at the time when the connector objects are automatically created for the generic technology connector that you create.

- Second line of the file

The second line of the parent data file must contain the field names (metadata) for the data in the file.

Note: In the generic technology connector context, the term **metadata** refers to the set of identity fields that constitute the user account information.

If you are using a language other than English, you must not enter non-ASCII characters on this line. See the Note in the preceding point for more information about this limitation.

- Third line of the file onward

From the third line onward, the parent data file can contain data in the language that you have selected for Oracle Identity Manager. This language can have an ASCII or non-ASCII character set. See "[Multilanguage Support](#)" on page 18-8 for more information about this limitation.

Even if there is no data from the third line onward, reconciliation will take place and the files are archived.

The following are contents of a sample parent data file:

```
##Active Directory user
Name TD,Address TD,User ID TD
John Doe,Park Street,jodoe
Jane Doe,Mark Street,jadoe
```

See Also: ["Permissions to Be Set on the Staging and Archiving Directories"](#)

- **Staging Directory (Multivalued identity data)**

Use this parameter to specify the path of the directory in which files containing multivalued (or child) account or identity data (for example, role membership data) are stored. It is *not* mandatory to specify a value for this parameter. This is a run-time parameter.

Note: In this guide, the terms multivalued account or identity data and child data have been used interchangeably.

Sample value for this parameter:

T:/TargetSystemDirectory/ChildData

Note:

- The staging directory for parent data files cannot be the same as the staging directory for multivalued user data files. In addition, if the staging directory is not on the same server on which Oracle Identity Manager is installed, it must be shared and mapped as a network drive on the Oracle Identity Manager server.
 - If you select the Trusted Source Reconciliation option on the Step 1: Provide Basic Information page, you must not specify a value for the Staging Directory (Multivalued Identity Data) parameter. This is because the reconciliation of multivalued (child) data is not supported in trusted source reconciliation.
-

For each type of multivalued account or identity data, there must be a different file in the shared directory. For example, if the multivalued user data for a particular target system is group membership data and role data, there must be one file for group membership data and a different file for role data.

Data stored in the child data files must conform to the conventions (first line, second line, and remaining lines) that are specified for the parent data files.

In addition, the same unique field must be present in the parent data file and each child data file. This field is used to uniquely link each record in the child data files with a single record in the parent data file. This structure is similar to the concept of integrity constraints (primary key-foreign key) in RDBMSs.

Note: The unique field must be the first field in the child data files.

The following are contents of a sample child data file holding role information that is linked to the sample parent data file listed earlier:

```
###Role
User ID TD,Role Name TD,Role Type TD
jodoe,admin1,admin
jadoe,admin2,admin
```

The following are contents of a sample child data file holding group membership information that is linked to the sample parent data file listed earlier:

```
###Group Membership
User ID TD,Group Name TD,Group Type TD
jodoe,OracleDev1,OracleDev
jadoe,OracleDev2,OracleDev
jadoe,OracleDev3,OracleDev
jadoe,OracleDev4,OracleDev
jadoe,OracleDev5,ConnectorDev
```

Note that the name of the unique field, `User ID TD`, is the same in the child data files and the parent data file.

On the Step 3: Modify Connector Configuration page as described in "[Step 3: Modify Connector Configuration Page](#)" on page 21-15, the name of a child data set is the same as the header that you provide in the child data file. For these sample child data files, the child data sets would be labeled `Role` and `Group Membership`. In addition, on the Step 4: Verify Connector Form Names page, the default names displayed for forms corresponding to the child data sets would be `Role` and `Group Membership`. As mentioned in "[Step 4: Verify Connector Form Names Page](#)" on page 21-29, you can either accept the default form names or change them.

See Also: "[Permissions to Be Set on the Staging and Archiving Directories](#)"

■ Archiving Directory

Use this parameter to specify the path of the directory in which parent and child data files that have already been reconciled are to be stored. This is a run-time parameter.

It is mandatory to specify a value for this parameter.

At the end of the reconciliation run, the data files are copied into the archiving directory and deleted from the staging directory.

The files moved to the archiving directory are not time stamped or marked in any way. Therefore, while specifying the path of the archiving directory, bear in mind the following guidelines:

- The archiving directory path that you specify must not be the same as the staging directory path. If you specify the same path, the existing files in the archiving directory are deleted at the end of the reconciliation run.
- If data files with the same names as the files used in the last reconciliation run are placed in the staging directory, the existing files in the archiving directory are overwritten by the new files from the staging directory at the end of the current reconciliation run.

These points are also mentioned in "[Step 2: Specify Parameter Values Page](#)" on page 21-6.

See Also: "[Permissions to Be Set on the Staging and Archiving Directories](#)"

■ File Prefix

Use this parameter to specify the prefix used to filter the names of files in the staging directories for both parent and child data files. During reconciliation, all

files (in the staging directories) with names that start with the specified prefix are processed, regardless of the file extension. This is a run-time parameter.

For example:

If you specify `usrdata` as the value of the File Prefix parameter, data is parsed from the following files placed in the staging directory for multivalued (child) user data files:

```
usrdataRoleData.csv
usrdataGroupMembershipData.txt
```

Data is not extracted from the following files in the same directory, because the file names do not begin with `usrdata`:

```
RoleData.csv
GroupMembershipData.txt
```

- **Specified Delimiter**

Use this parameter to specify the character that is used as the delimiter character in the parent and child data files. You can specify only a single character as the value of this parameter. This is a run-time parameter. This parameter overrides the Tab Delimiter parameter.

Note: You cannot use the space character () as a delimiter.

In addition, you must ensure that the character you specify is used only as the delimiter in the data files. If this character is also used inside the data itself, the data row (or record) is not parsed correctly. For example, you must not use the comma (,) as the delimiter if any data value contains a comma.

- **Tab Delimiter**

Use this parameter to specify whether or not the file is delimited by tabs. This is a run-time parameter. This parameter is ignored if you specify a value for the Specified Delimiter parameter.

- **Fixed Column Width**

If the input file contains fixed-width data, use this parameter to specify the width in characters of the data columns. This is a run-time parameter.

Note: In this context, the term "fixed-width" refers to the number of characters in the data field, not the byte length of the field. This means that, for example, four characters of single-byte data and four characters of multibyte data are the same in terms of width.

This parameter is ignored if you specify a value for the Specified Delimiter or Tab Delimiter parameter.

- **Unique Attribute (Parent Data)**

For multivalued user data, use this parameter to specify the field that is common to both the parent data and child data files. In the examples described earlier, the requirement for a unique attribute is fulfilled by the `USER ID TD` field, which is present in both the parent and child data files. This is a run-time parameter.

Note: If you select the Trusted Source Reconciliation option on the Step 1: Provide Basic Information page, you must not specify a value for the Unique Attribute (Parent Data) parameter. This is because the reconciliation of multivalued (child) data is not supported in trusted source reconciliation.

- **File Encoding**

Use this parameter to specify the character set encoding used in the parent and data files. This is a design parameter.

Specify Cp1251 for data files stored on a computer running an operating system with the English-language setting. This is the canonical name for the `java.io` API that is supported by the generic technology connector framework. For any other language that you select from the list given in the "Multilanguage Support" section, you must specify the canonical name for the corresponding `java.io` API listed on the following Web page:

<http://java.sun.com/j2se/1.4.2/docs/guide/intl/encoding.doc.html>

Note: The canonical name that you specify for the API must be entered exactly the way it is displayed on this Web page. You must not change the case (uppercase or lowercase) of the canonical name.

For example, if you want to specify the encoding set for the Traditional Chinese language on a Microsoft Windows computer, you specify MS950 as the value of the File Encoding parameter.

Permissions to Be Set on the Staging and Archiving Directories

You must ensure that the required permissions are set on the staging and archiving directories. The following table describes the effect of the various permissions on the shared directories that are used to hold staging and archiving data files.

Storage Entity	Access Permission	Reason for Access Permission Requirement
Staging directory for parent data files	Read	This permission is required for reconciliation to take place. An error message is logged if this permission is not applied.
Staging directory for parent data files	Write	This permission is required for the deletion of data files from the parent staging directory at the end of the archive process.
Staging directory for parent data files	Execute	Not applicable
Staging directory for child data files	Read	This permission is required for the reconciliation of child data. An error message is logged if this permission is not applied.
Staging directory for child data files	Write	This permission is required for the deletion of data files from the child staging directory at the end of the archive process.
Staging directory for child data files	Execute	Not applicable

Storage Entity	Access Permission	Reason for Access Permission Requirement
Archiving directory	Write	This permission is required for the copying of parent and child data files to the archiving directory during the archive process. Even if this permission is not applied: <ul style="list-style-type: none"> ■ Parent and child data reconciliation takes place. ■ Files are deleted from the parent and child staging directories if the required permissions have been set on those directories.
Archiving directory	Execute	Not applicable
Parent or child data file in staging directory	Read	This permission is required for the reconciliation of the data in the file. An error message is logged if this permission is not applied.
Parent or child data file in staging directory	Write	This permission is required for the deletion of the data file at the end of the archive process. An error message is logged if this permission is not applied. However, data in this file is reconciled.
Parent or child data file in staging directory	Execute	Not applicable

Note: Data files in the staging directory cannot be deleted if they are open in any editor or are open for writing by any other program.

19.2 CSV Reconciliation Format Provider

The CSV reconciliation format provider converts reconciliation data that is in character-delimited, tab-delimited, or fixed-length format into a format that is supported by Oracle Identity Manager.

Although the CSV reconciliation format provider is packaged as a standalone provider, all of its parameters are bundled with the shared drive transport provider. If you select the shared drive transport provider on the Step 1: Provide Basic Information page, you must select the CSV format provider. When you select this provider, its parameters are displayed along with the shared drive transport provider parameters.

19.3 SPML Provisioning Format Provider

The SPML provisioning format provider converts the provisioning data generated during a provisioning operation on Oracle Identity Manager into an SPML request that can be processed by an SPML-compatible target system.

Note: Each SPML request is sent in a SOAP message. The SOAP header carries authentication information for the request. The actual SPML request data is the SOAP message body.

See [Chapter 32, "Using SPML Services"](#) for information about the structure of the SPML-SOAP message.

You can access sample SOAP messages in the following directory:

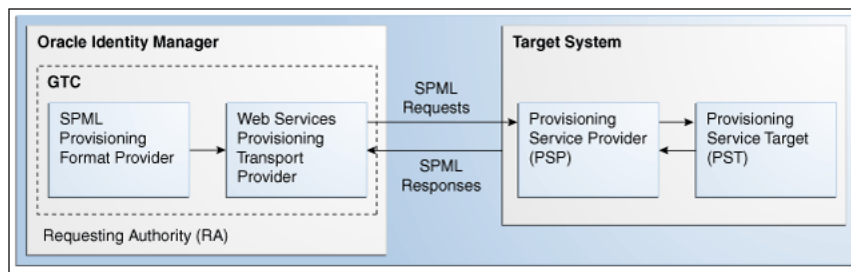
`OIM_HOME/GTC/Samples/spml`

For information about the SPML specification, see the following Web page on the OASIS Web site at

<http://www.oasis-open.org/specs/index.php#spmlv2.0>

Figure 19–1 shows the setup of the system in which the SPML provisioning format provider acts as the requesting authority (RA), and the target system provides the provisioning service provider (PSP) and the provisioning service target (PST).

Figure 19–1 Communication Between the SPML Provisioning Format Provider and the Target System



During actual provisioning, a Velocity template engine is used to create the SOAP-SPML requests. For the following processes, the provider generates SOAP requests based on the SPML 2.0 DSML profile:

- Add request
- Modify request for the following Oracle Identity Manager process tasks:
 - Field updated
 - Add child data
 - Modify child data
 - Delete child data
- Suspend request (for Disable Oracle Identity Manager process tasks)
- Resume request (for Enable Oracle Identity Manager process tasks)
- Delete request

The Create Organization, Update Organization, and Delete Organization are not supported. This is because the resource object created for a generic technology connector does not support provisioning operations for organizations. The Create Group, Update Group, and Delete Group operations are not supported. This is because Oracle Identity Manager does not support operations to provision groups.

When you select this provider, the following identity fields are displayed by default on the Step 3: Modify Connector Configuration page as described in "[Step 3: Modify Connector Configuration Page](#)" on page 21-15, along with the ID field:

- objectClass
- containerID

For each provisioning task (for example, Create User and Modify User), the provider generates a request in a predefined format.

The following sections discuss the parameters of this provider:

- [Run-Time Parameters](#)
- [Design Parameters](#)

Depending on the application server that you use, some of the run-time and design parameters are mandatory and some have fixed values. The following sections discuss these parameters:

- [Nonmandatory Parameters](#)
- [Parameters with Predetermined Values](#)

19.3.1 Run-Time Parameters

The following are run-time parameters of the SPML provisioning format provider:

- **Target ID**
This value uniquely identifies the target system for provisioning operations.
- **User Name (authentication)**
This is the user name of the account required to connect to the target system (PST) through the Web service interface (PSP).
- **User Password (authentication)**
This is the password of the user account required to connect to the target system (PST) through the Web service interface (PSP).

19.3.2 Design Parameters

The following are design parameters of the SPML provisioning format provider:

See Also: For more information about the SOAP elements and attributes mentioned in this section, visit the following Web site

<http://www.w3.org/TR/wsd120/>

- **Web Service SOAP Action**
In the WSDL file, this is the value of the soapAction attribute of the operation element.
- **WSSE Configured for SPML Web Service?**
Select this check box if the Web service is configured to authenticate incoming requests by using WS-Security credentials.
- **Custom Authentication Credentials Namespace**

Note: You need not specify a value for this parameter if you select the SPML Web Service WSSE Configured? check box.

This is the name of the credentials namespace that you have defined for the Web service. In most cases, this namespace is the same as the target namespace.

- **Custom Authentication Header Element**

Note: You need not specify a value for this parameter if you select the SPML Web Service WSSE Configured? check box.

This is the name of the element that will contain the credentials of the user account used to connect to the target system. In other words, this is the parent element in the custom authentication section of the SOAP message header.

- **Custom Element to Store User Name**

Note: You need not specify a value for this parameter if you select the SPML Web Service WSSE Configured? check box.

This is the name of the element in the custom authentication section that will contain the user name you specify as the value of the User Name (authentication) parameter.

- **Custom Element to Store Password**

Note: You need not specify a value for this parameter if you select the SPML Web Service WSSE Configured? check box.

This is the name of the element in the custom authentication section that will contain the user name you specify as the value of the User Password (authentication) parameter.

- **SPML Web Service Binding Style (DOCUMENT or RPC)**

In the WSDL file, this is the value of the `style` attribute of the binding element. You must enter either `DOCUMENT` or `RPC`.

Note: You must enter the value `DOCUMENT` or `RPC`. Do not use lowercase letters in the value that you specify.

- **SPML Web Service Complex Data Type**

In the WSDL file, this is the value of the `name` attribute of the `complexType` element. This parameter is applicable only if the binding style is `DOCUMENT`. You must specify a value for this parameter if the target Web service is running on Oracle WebLogic Server.

- **SPML Web Service Operation Name**

In the WSDL file, this is the value of the `name` attribute of the `operation` element. This parameter is applicable only if the binding style is `RPC`.

- **SPML Web Service Target Namespace**

In the WSDL file, this is the value of the `targetNamespace` attribute of the `definition` element.

- **SPML Web Service Soap Message Body Prefix**

This is the name of the custom prefix element that contains the SOAP message body. If the target Web service is running on Oracle WebLogic Server, IBM WebSphere Application Server, JBoss Application Server, or Oracle Application Server, then you need not specify a value for this parameter. However, if you are using a different application server, you must enter the name of the custom prefix element. The following is the prefix element if the Web service is running on Oracle WebLogic Server:

```
<SPMLv2Document xmlns="http://xmlns.oracle.com/OIM/provisioning">
```

- **ID Attribute for Child Dataset Holding Group Membership Information**

This is the name of the unique identifier field for a provisioning staging child data set that holds group membership information. For provisioning operations on the child data set that contains this field, the SOAP packet will contain SPML code for group operations. The following is an SPML code block for this type of group operation:

```
<modification modificationMode="add">
  <capabilityData capabilityURI="urn:oasis:names:tc:SPML:2:0:reference"
    mustUnderstand="true">
    <reference typeOfReference="memberOf"
      xmlns="urn:oasis:names:tc:SPML:2:0:reference">
      <toPsoID ID="Groups:1" targeted="120"/>
    </reference>
  </capabilityData>
</modification>
```

For provisioning operations on the child data sets that do not contain this field, the SOAP packet will contain ordinary SPML code. The following is an SPML code block for this type of group operation:

```
<modification>
  <dsml:modification name="Group Membership" operation="add">
    <dsml:value>AdminOra, System Admins, USA</dsml:value>
  </dsml:modification>
</modification>
```

19.3.3 Nonmandatory Parameters

For Oracle WebLogic Server, you need not specify values for the following parameters:

- SPML Web Service Complex Data Type
- SPML Web Service Soap Message Body Prefix
- ID Attribute for Child Dataset Holding Group Membership Information

19.3.4 Parameters with Predetermined Values

For Oracle WebLogic Server, you can specify predetermined values for the following parameters:

- Web Service URL:
`http://IP_address:port_number/spmlws/OIMProvisioning`
- SPML Web Service Binding style (DOCUMENT or RPC): RPC
- SPML Web Service Operation Name: `processRequest`

19.4 Web Services Provisioning Transport Provider

The Web Services provisioning transport provider acts as a Web service client and carries provisioning request data from Oracle Identity Manager to the target system Web service.

The following types of target system Web services are supported:

- RPC-literal
- RPC-encoded
- DOCUMENT-literal

The following is the parameter of the Web Services provisioning transport provider:

Web Service URL

Use this parameter to specify the URL of the Web service that you want to use for sending a provisioning request to the target system. This is a run-time parameter. In the WSDL file, the Web service URL is the value of the `location` attribute of the `wsdl:soap:address` element.

If you include the Web Services provisioning transport provider in the generic technology connector that you create, you may want to configure Secure Sockets Layer (SSL) communication between the target system and Oracle Identity Manager. The following section provides information about this procedure.

19.4.1 Configuring SSL Communication Between Oracle Identity Manager and the Target System Web Service

This section describes the procedure to configure the application server on which Oracle Identity Manager is installed for SSL communication.

You can perform this procedure only if all the following conditions are true:

- You want to include the Web Services provisioning transport provider in the generic technology connector that you plan to create.
- The target Web service is running on an SSL-enabled application server.

To configure SSL communication between Oracle Identity Manager and the target system Web service:

Note: You can perform this procedure prior to creating the generic technology connector.

1. Export the target application server certificate as follows:
 - For a target system Web service deployed on JBoss Application Server, Oracle WebLogic Server, or Oracle WebLogic Server, run the following command:

```
JAVA_HOME/jre/bin/keytool -export -alias default -file
exported-certificate-file -keystore app-server-specific-keystore
-storetype jks -storepass keystore-password -provider
```

```
sun.security.provider.Sun
```

In this command:

- * Replace *JAVA_HOME* with the full path to the SUN JDK directory.
 - * Replace *exported-certificate-file* with the name of the file in which you want the exported certificate to be stored.
 - * Replace *app-server-specific-keystore* with the path to the keystore on the application server.
 - * Replace *keystore-password* with the password for the keystore.
- For a target system Web service deployed on IBM WebSphere Application Server or Oracle WebLogic Server on AIX, run the following command:

```
JAVA_HOME/jre/bin/keytool -export -alias default -file
exported-certificate-file -keystore app-server-specific-keystore -storetype
jks -storepass keystore-password -provider com.ibm.crypto.provider.IBMJCE
```

In this command:

- * Replace *JAVA_HOME* with the full path to the IBM JDK directory.
- * Replace *exported-certificate-file* with the name of the file in which you want the exported certificate to be stored.
- * Replace *app-server-specific-keystore* with path to the keystore on the application server.
- * Replace *keystore-password* with the password for the keystore.

When the command is run, the exported certificate file is stored in the file that you specify as the value of *exported-certificate-file*.

2. Import the certificate file exported in the preceding step into the Oracle Identity Manager truststore as follows:
 - a. Copy the certificate file exported in the preceding step into a temporary directory on the Oracle Identity Manager server.
 - b. Run the following command:

```
JAVA_HOME/jre/bin/keytool -import -trustcacerts -alias servercert -noprompt
-keystore OIM_HOME\config\xlkeystore -file certificate_file
```

In this command:

- Replace *JAVA_HOME* with full path to the JDK directory. For Oracle Identity Management Server deployed on IBM WebSphere Application Server, the path must be that of the IBM JDK directory. For Oracle Identity Manager deployed on JBoss Application Server, Oracle WebLogic Server, or Oracle WebLogic Server, the path must be that of the SUN JDK directory.
- Replace *OIM_HOME* with the full path of the Oracle Identity Manager home directory
- Replace *certificate_file* with the path of the temporary directory into which you copy the certificate file.

Note: If the application server is enabled for one-way SSL communication, you need not perform the rest of this procedure.

3. Import the Oracle Identity Manager certificate into the target system application server truststore as follows:

Note: Perform the following steps only if the application server is enabled for two-way SSL communication.

- a. Export the Oracle Identity Manager certificate file.

For Oracle Identity Manager deployed on Oracle WebLogic Server, run the following command:

```
JAVA_HOME/jre/bin/keytool -export -alias xell -file
OIM_HOME\config\xell.cert -keystore OIM_HOME\config\xlkeystore -storetype
jks -provider sun.security.provider.Sun
```

In this command:

- Replace *JAVA_HOME* with the full path to the SUN JDK directory.
- Replace *OIM_HOME* with the full path of the Oracle Identity Manager home directory.

- b. Import the certificate file that you export in Step 3a into the truststore of the application server as follows:

Copy the exported Oracle Identity Manager certificate file to a temporary directory on the target application server.

Next, run the following command on the target application server:

- If the target application server is JBoss Application Server, Oracle WebLogic Server, or Oracle WebLogic Server, run the following command:

```
JAVA_HOME/jre/bin/keytool -import -alias alias -trustcacerts -file
OIM-certificate-file -keystore app-server-specific-truststore
-storetype jks -storepass truststore-password -provider
sun.security.provider.Sun
```

In this command:

- * Replace *JAVA_HOME* with the full path to the SUN JDK directory.
 - * Replace *alias* with an alias for the certificate in the truststore of the target application server.
 - * Replace *OIM-certificate-file* with the name of the exported Oracle Identity Manager certificate file.
 - * Replace *app-server-specific-truststore* with path to the truststore on the target application server.
 - * Replace *truststore-password* with the password for the truststore on the target application server.
- If the target application server is IBM WebSphere Application Server, run the following command:

```
JAVA_HOME/jre/bin/keytool -import -alias alias -trustcacerts -file
OIM-certificate-file -keystore app-server-specific-truststore
-storetype pkcs12 -storepass truststore-password -provider
com.ibm.crypto.provider.IBMJCE
```

In this command:

- * Replace *JAVA_HOME* with the full path to the SUN JDK directory.
- * Replace *alias* with an alias for the certificate in the target truststore.
- * Replace *OIM-certificate-file* with the name of the exported Oracle Identity Manager certificate file.
- * Replace *app-server-specific-truststore* with the path to the truststore on the target application server.
- * Replace *truststore-password* with the password for the truststore on the target application server.

See Also: SSL configuration documentation for the target application server

19.5 Transformation Providers

Note: Use the information provided in this section while performing the instructions given in [Section 21.2.4.3, "Step 3: Modify Connector Configuration Page"](#).

A transformation provider is used to transform user data while it is in transit between the source and destination data sets listed in the following table.

Source Data Set	Destination Data Set	Purpose of the Transformation
Source	Reconciliation Staging	Data is transformed before it is used to create reconciliation events.
Oracle Identity Manager	Provisioning Staging	Data is transformed before it is used to create the provisioning request to be sent to the target system.

The following predefined transformation providers are included in the current release of Oracle Identity Manager:

- [Concatenation Transformation Provider](#)
- [Translation Transformation Provider](#)

19.5.1 Concatenation Transformation Provider

You use the concatenation transformation provider to concatenate the values of two fields of data sets to create the input for a single field of another data set.

The following example explains the output format of this provider:

Suppose the input values are the following fields of the source data set:

- First Name: John
- Last Name: Doe

When the concatenation transformation provider is applied to these two fields, the output value is as follows:

John Doe

Note: As shown in the preceding example, the concatenation transformation provider adds a space between the values of the two input fields.

The following procedure describes how to add a concatenation transformation provider while creating a generic technology connector:

Note: This procedure explains in detail the instruction given in Step 5 of [Section 21.2.4.3.1, "Adding or Editing Fields in Data Sets"](#). It is assumed that you have already selected the **Concatenation** option from the **Mapping Action** list on the Step 1: Field Information page and that you have performed Steps 2 and 3 given in that section.

On the Step 2: Mapping page in the pop-up window, perform the following steps:

1. From the **Dataset** list in the Input 1 region, select the data set containing the first field that you want to concatenate. From the **Field Name** list, select the first field. Alternatively, you can use the **Literal** option to specify a literal (or fixed) value as the first concatenation input.

For the example described earlier, from the **Dataset** list in the Input 1 region, select the data set containing the **First Name** field. Then, from the **Field Name** list, select **First Name**.

2. From the **Dataset** list in the Input 2 region, select the data set containing the second field that you want to concatenate. Then, from the **Field Name** list, select the second field. Alternatively, you can use the **Literal** option to specify a literal (or fixed) value as the second concatenation input.

For the example described earlier, from the **Dataset** list in the Input 2 region, select the data set containing the **Last Name** field. Then, from the **Field Name** list, select **Last Name**.

19.5.2 Translation Transformation Provider

A translation operation involves accepting a certain (literal) value as input and converting it into another value.

The following example illustrates a translation operation:

Suppose the Source data set contains the Country field and data values stored in this field can take one of the following values:

- Austria
- France
- Germany
- India
- Japan

When these values are propagated to the reconciliation staging data set, you want to convert these values to the following:

- AT
- FR

- DE
- IN
- JP

To automate this translation, you can use the translation transformation provider.

To use the translation transformation provider:

1. Use the Design Console to create a lookup definition that stores the input and decoded values.

See Also: [Section 15.2.1, "Creating a Lookup Definition"](#)

Note: While creating a lookup definition in the Lookup Definition form, you must select the Lookup Type option, and not the Field Type option.

For the Country field example described earlier, the Code Key and Decode values are as shown in the following table.

Code Key	Decode
Austria	AT
France	FR
Germany	DE
India	IN
Japan	JP

2. Define a transformation (translation) mapping between the input field and output field for the translation. As mentioned earlier, a transformation can be set up between the following pairs of data sets:
 - Source and Reconciliation Staging
 - Oracle Identity Manager and Provisioning Staging

Note: This procedure explains in detail the instruction given in Step 5 of [Section 21.2.4.3.1, "Adding or Editing Fields in Data Sets"](#). It is assumed that you have already selected the **Concatenation** option from the **Mapping Action** list on the Step 1: Field Information page and that you have performed Steps 2 and 3 given in that section.

- a. On the Step 3: Mapping page, from the **Dataset** list in the Input region, select the data set containing the field that will provide the input value for the translation operation. Then, from the **Field Name** list, select the field itself.

For the Country field example described earlier, select the data set containing the Country field and select the Country field.

- b. In the Lookup Code Name region, select **Literal** and enter the name of the lookup definition that you create in the preceding step.

Note: You must not specify a data set name and field in the Lookup Code Name region. Although there is no validation to stop you from selecting a data set name and field, the translation operation would fail during actual reconciliation or provisioning operations.

This point is also mentioned in the Mappings section .

For the Country field example described earlier, select **Literal** and select the lookup definition you create in Step 1.

19.5.2.1 Configuring Account Status Reconciliation

User account status information is used to track whether or not the owner of a target system account is to be allowed to access and use the account. If required, you can use the translation transformation provider to reconcile account status information.

Note: The Design Console offers an alternative method to configure account status reconciliation. This method does not involve the use of a generic technology connector. [Section 12.3.2.2.1, "User Account Status Reconciliation"](#) describes this method.

You need to use the translation transformation provider only if account status values used in the target system are not the same as the values used in Oracle Identity Manager. For a target resource, Oracle Identity Manager uses the following values:

- Enabled state: `Enabled`
- Disabled state: `Disabled`

For a trusted source, Oracle Identity Manager uses the following values:

- Enabled state: `Active`
- Disabled state: `Disabled`

The procedure to configure account status reconciliation can be summarized as follows:

Note: Detailed instructions to perform these steps are provided later in this section.

1. Create a lookup definition that maps the status values used in the target system with the values used in Oracle Identity Manager.
2. While creating the generic technology connector, use the translation transformation provider to create a transformation mapping between the fields that hold account status values in the Source data set and the reconciliation staging data set.

The following example describes the action that you must perform:

Suppose the following fields are used to hold account status values:

- The User Status field of the Source data set holds the values `True` (for a user in the Enabled state) and `False` (for a user in the Disabled state).
- The User Status field of the reconciliation staging data set must hold one of the following pairs of values:

- For target resource reconciliation, the field must hold `Enabled` or `Disabled`.
- For trusted source reconciliation, the field must hold `Active` or `Disabled`.

You must create a transformation mapping that converts the `True/False` values in the `User Status` field of the Source data set into corresponding `Enabled/Disabled` or `Active/Disabled` values. During reconciliation, these converted values are sent to the `User Status` field of the reconciliation staging data set.

3. Create a mapping between the field that holds account status values in the reconciliation staging data set and one of the following fields:
 - The `OIM Object Status` field of the `OIM – Account` data set, for target resource reconciliation
 - The `Status` field of the `OIM – User` data set, for trusted source reconciliation

During reconciliation, this mapping is used to propagate status values from the reconciliation staging data set to the `OIM – Account` or `OIM – User` data set.

Detailed steps to configure account status reconciliation are as follows:

1. Create a lookup definition that maps the status values used in the target system with the values used in Oracle Identity Manager.

See Also: [Section 15.2, "Lookup Definition Form"](#)

The Code Key values in the lookup definition must be the same as the values used to represent the account status in the target system. The Code Key and Decode values for both trusted and target resource reconciliation are as shown in the following table:

Code Key	Decode (for Trusted Source Reconciliation)	Decode (for Target Resource Reconciliation)
<i>Target system status value for a user account that is in the Enabled state</i>	Active	Enabled
<i>Target system status value for a user account that is in the Disabled state</i>	Disabled	Disabled

Examples of Code Key values are `True/False`, `Yes/No`, and `1/0`. The Decode values must be set to the exact value, including the case (uppercase and lowercase), shown in the table.

Note: While creating the lookup definition in the Lookup Definition form, you must select the Lookup Type option, and not the Field Type option.

2. The procedure to create the generic technology connector is described in [Chapter 21, "Creating and Managing Generic Technology Connectors"](#). While creating the generic technology connector, perform the following steps on the Step 3: Modify Connector Configuration page:
 3. Modify Connector Configuration page:

Note: These steps are a condensed version of the procedure described in [Section 21.2.4.3.1, "Adding or Editing Fields in Data Sets"](#). Refer to that section for a description of the terms and GUI elements mentioned in the following steps.

- a. If the target system status field is displayed on the Step 3: Modify Connector Configuration page, click the Edit icon for the field in the reconciliation staging data set.

If the field is not displayed, click the Add icon of the reconciliation staging data set.
 - b. On the Step 1: Field Information page, specify values for the following GUI elements:
 - **Field Name:** If you are adding the field, specify a name for it. The field name that you specify must contain only ASCII characters, because non-ASCII characters are not allowed.
 - **Mapping Action:** Select **Create Mapping With Translation** from this list.
 - **Matching Only:** Ensure that this check box is deselected.
 - **Create End-to-End Mapping:** If you are adding the field, select this check box.
 - **Multi-Valued Field:** Ensure that this check box is deselected.
 - **Data Type:** Select the data type of the field.
 - **Length:** Specify the character length of the field.
 - **Required:** Select this check box if you want to ensure that the field always contains a value.
 - **Encrypted:** Ensure that this check box is deselected.
 - **Password Field:** Ensure that this check box is deselected.
 - c. Click **Continue**.
 - d. On the Step 3: Provide Mapping Information page, perform the following steps:

In the Input region:
 - From the **Dataset** list, select **Source**.
 - From the **Field Name** list, select the field that stores status values.In the Lookup Code Name region, select **Literal** and enter the name of the lookup definition that you create in Step 1.
 - e. If required, select a validation check for the field and click **Add**. In other words, select the validation provider that you want to use.
 - f. Click **Continue**, and click **Close**.
3. Create a mapping between the status field of the reconciliation staging data set and either the OIM Object Status field of the OIM - Account data set or the Status field of the OIM - User data set as follows:

Note: These steps are a condensed version of the procedure described in [Section 21.2.4.3.1, "Adding or Editing Fields in Data Sets"](#).

- a. For target resource reconciliation, click the edit icon for the OIM Object Status field of the OIM - Account data set.

For target resource reconciliation, click the edit icon for the Status field of the OIM - User data set.

Note: If a mapping already exists between the status field of the reconciliation staging data set and the OIM Object Status field or Status field, apply the instructions given in this step only where required.

- b. On the Step 1: Field Information page, specify values for the following GUI elements:
- Mapping Action: Select **Create Mapping Without Transformation** from this list.
 - **Matching Only**: Ensure that this check box is deselected.
- c. Click **Continue**.
- d. In the Input region on the Step 3: Mapping page, select the status field of the reconciliation staging data set.
- e. Click **Continue**, **Continue**, and click **Close**.
- f. To add or edit other fields displayed on the Step 3: Modify Connector Configuration page, continue with the procedure described in [Section 21.2.4.3.1, "Adding or Editing Fields in Data Sets"](#).

19.6 Validation Providers

[Table 19–1](#) describes the validation providers that are shipped with this release of Oracle Identity Manager.

Note: Except for the Validate Date Format provider, all the providers in this table are implementations of methods of the `GenericValidator` class in the Apache Jakarta Commons API.

Table 19–1 Validation Providers

Validation Provider	Description
IsBlankOrNull	Returns true if the field value is null and is not blank
IsInRange	Returns true if the field value is within a range specified by a minimum and maximum value pair
IsByte	Checks if the field value can be converted to a byte primitive
IsDouble	Checks if the field value can be converted to a double primitive
IsFloat	Checks if the field value can be converted to a float primitive
IsInteger	Checks if the field value can be converted to an integer primitive

Table 19–1 (Cont.) Validation Providers

Validation Provider	Description
IsLong	Checks if the field value can be converted to a long primitive
IsShort	Checks if the field value can be converted to a short primitive
MatchRegexp	Checks if the field value matches the specified regular expression Note: A regular expression is a string that is used to describe or match a set of strings according to specific syntax rules.
MaxLength	Checks if the length of the field value is less than or equal to the specified value
MinLength	Checks if the length of the field value is greater than or equal to the specified value
Validate Date Format	Validates date values in target system records before these records are reconciled into Oracle Identity Manager The value of the Source Date Format parameter is used as the basis for validation. This validation provider is applied if you specify a value for the Source Date Format parameter on the Step 2: Specify Parameter Values page, regardless of whether or not you select this provider on the Step 3: Modify Connector Configuration page. Note: Unlike the other providers in this table, the Validate Date Format is not an implementation of a method of the <code>GenericValidator</code> class in the Apache Jakarta Commons API.

Creating Custom Providers for Generic Technology Connectors

You will need to create custom providers to address provider requirements that cannot be addressed by the predefined providers. This chapter discusses the steps to create custom providers.

Topics in this chapter include:

- [Role of Providers](#)
- [Creating Custom Providers](#)
- [Reusing Providers](#)
- [Deploying the Custom Providers](#)

20.1 Role of Providers

The following sections discuss the role of providers during generic technology connector creation and use:

- [Role of Providers During Generic Technology Connector Creation](#)
- [Role of Providers During Reconciliation](#)
- [Role of Providers During Provisioning](#)

20.1.1 Role of Providers During Generic Technology Connector Creation

You create a generic technology connector by using the Administrative and User Console. Defining data sets and the flow of data between these data sets is one of the tasks of the connector creation procedure. The metadata detection process facilitates this task.

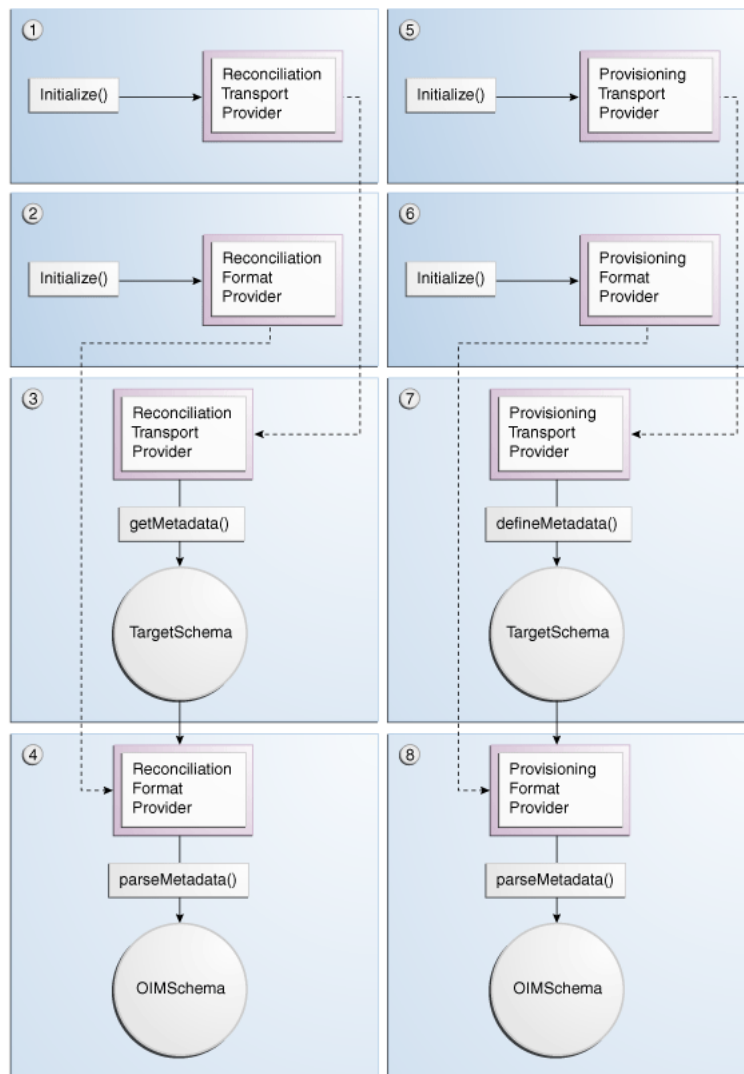
In the generic technology connector context, the term **metadata** refers to the set of identity fields that constitute the user account information. The term **metadata detection** refers to the reading and parsing of target system metadata by the providers.

The metadata detection feature is supported for all the provider types. In other words, when you create a custom provider, you can incorporate into the provider the capability to read metadata.

Note: The Javadocs use the term **metadata definition** instead of **metadata detection**.

Figure 20–1 shows the metadata detection process.

Figure 20–1 Metadata Detection Process



The following sequence of steps describe the process of metadata detection. This sequence of steps is based on the assumption that you select both the Reconciliation and Provisioning options while creating the generic technology connector. If you do not select either option, the corresponding steps are not performed.

See Also: *Oracle Fusion Middleware Java API Reference for Oracle Identity Manager* for detailed information about the SPI methods and value objects mentioned in the following steps.

In the Javadocs, the terms **metadata detection** and **metadata definition** have been used interchangeably.

1. The `initialize` method of the reconciliation transport provider is called to create an instance of that provider.
2. The `initialize` method of the reconciliation format provider is called to create an instance of that provider.

3. The `getMetadata` method of the reconciliation transport provider is called to fetch metadata from the target system. The output of this method is the `TargetSchema` value object containing metadata fetched from the target system.
4. The `parseMetadata` method of the reconciliation format provider is called to parse metadata fetched from the target system. The output of this method is the `OIMSchema` value object containing metadata fetched from the target system.

Note: The `OIMSchema` value object corresponds to the Source data sets discussed in [Section 18.2.1, "Providers and Data Sets of the Reconciliation Module"](#).

5. The `initialize` method of the provisioning transport provider is called to create an instance of that provider.
6. The `initialize` method of the provisioning format provider is called to create an instance of that provider.
7. If the reconciliation transport provider and reconciliation format provider are not able to detect metadata, Steps 1 through 4 are repeated for the provisioning transport provider and provisioning format provider.

Note: After a provider is initialized, it is stored in the Oracle Identity Manager cache until any one of the following events occurs:

- Cache is purged.
- Oracle Identity Manager is restarted.
- The generic technology connector is modified after it is created.

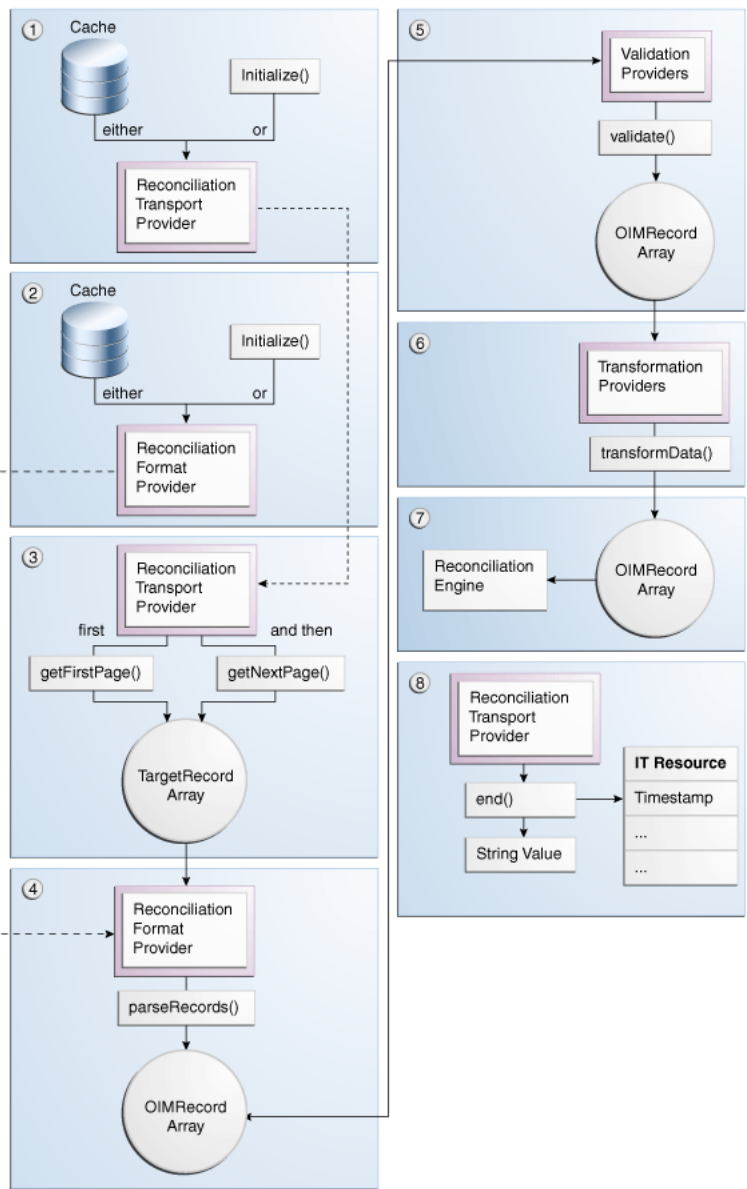
The validation providers and transformation providers are instantiated only when they are needed. They are not stored in cache.

The shared drive reconciliation transport provider and CSV reconciliation format provider can detect metadata from the target system. However, this function is not supported for the Web Services provisioning transport provider and SPML provisioning format provider.

20.1.2 Role of Providers During Reconciliation

[Figure 20-2](#) shows the role of providers during reconciliation.

Figure 20–2 Role of Providers During Reconciliation



The following steps describe the role of providers during reconciliation:

1. If an instance of the reconciliation transport provider is not available in cache, the `initialize` method is called to create an instance of that provider.
2. If an instance of the reconciliation format provider is not available in cache, the `initialize` method is called to create an instance of that provider.
3. While using the Administrative and User Console to create a generic technology connector, you can specify a batch size for the reconciliation run. By using this parameter, you break the total number of records that the reconciliation engine fetches from the target system, during each reconciliation run, into batches. The default value of this parameter is `All`.

If you do not specify a batch size, at this stage of reconciliation, the `getFirstPage` method of the reconciliation transport provider is called to fetch the entire set of target system records that are ready for reconciliation.

If you specify a batch size, the `getFirstPage` method of the reconciliation transport provider is called to fetch the first batch of target system records for reconciliation. The `getNextPage` method of the same provider is called (multiple times, if required) if there are more batches of target system records for reconciliation.

4. The `parseRecords` method of the reconciliation format provider is called to process each record of the `TargetRecord` value objects array. The output of this method is an array of `OIMRecord` value objects.
5. While creating the generic technology connector, if you select validation providers to validate data while it is in transit from the Source data sets to the reconciliation staging data sets:
 - a. An instance of the validation provider is created.
 - b. The `validate` method of each validation provider is run on the specified attribute of each record of the `OIMRecord` value objects array.

If you do not select validation providers while creating the generic technology connector, Step 5 is not performed and each element of the `OIMRecord` value objects array is passed on to Step 6.

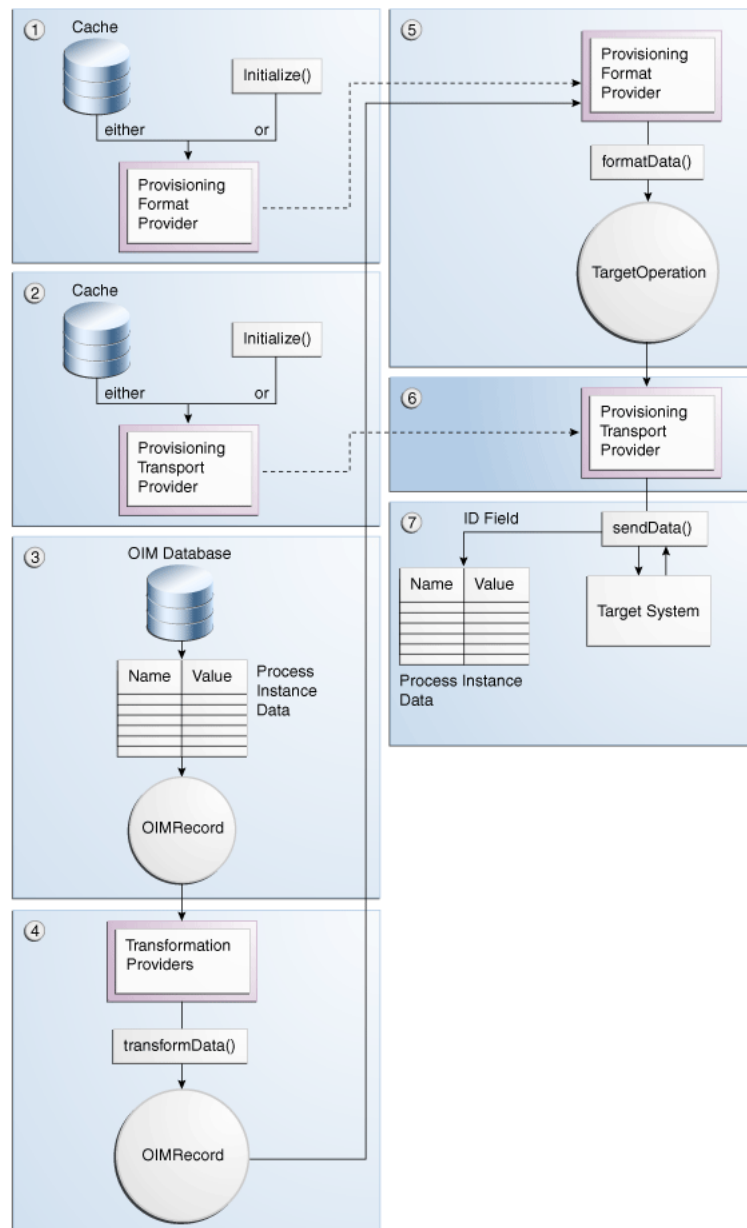
6. While creating the generic technology connector, if you selected transformation providers to modify data that is in transit from the Source data sets to the reconciliation staging data sets:
 - a. An instance of the transformation provider is created.
 - b. The `transformData` method of the transformation providers processes the `OIMRecord` value objects array that was generated as the output of one of the following:
 - The `validate` method of each validation provider (if you selected validation providers)
 - The `parseRecords` method of the reconciliation format provider (if you did not select validation providers)

If you do not select transformation providers while creating the generic technology connector, Step 6 is not performed and each element of the `OIMRecord` value objects array from the previous step (Step 4 or 5) is passed on to Step 7.

7. At this stage, the `OIMRecord` value objects array corresponds to the reconciliation staging data sets discussed in [Section 18.2.1, "Providers and Data Sets of the Reconciliation Module"](#). Each element of the `OIMRecord` value objects array is passed on to the reconciliation engine.
8. At the end of the reconciliation procedure, the `end` method of the reconciliation transport provider is called. This method returns a string value, which the generic technology connector framework stores in the `Timestamp` parameter of the IT resource. The framework uses the `Timestamp` parameter to track the stage at which the reconciliation run was completed.

20.1.3 Role of Providers During Provisioning

Figure 20–3 shows the role of providers during provisioning.

Figure 20–3 Role of Providers During Provisioning

The following steps describe the role of providers during provisioning:

1. If an instance of the provisioning transport provider is not available in cache, the `initialize` method is called to create an instance of that provider.
2. If an instance of the provisioning format provider is not available in cache, the `initialize` method is called to create an instance of that provider.
3. The generic technology connector adapter is one of the connector objects created when you create the generic technology connector. This adapter converts the provisioning data record into a hashmap of name-value pairs. This hashmap contains process instance data. Each hashmap is converted into an element of the `OIMRecord` value object. At this stage, the `OIMRecord` value object corresponds to the Oracle Identity Manager data sets discussed in [Section 18.2.3, "Oracle Identity Manager Data Sets"](#).

4. While creating the generic technology connector, if you select transformation providers to modify data that is in transit from the Oracle Identity Manager data sets to the provisioning staging data sets:
 - a. An instance of the transformation provider is created.
 - b. The `transformData` method of the transformation providers processes the specified attributes of the input `OIMRecord` value object and converts these records into an output `OIMRecord` value object. At this stage, the `OIMRecord` value object corresponds to the provisioning staging data sets discussed in [Section 18.2.2, "Providers and Data Sets of the Provisioning Module"](#).
5. The `formatData` method of the provisioning format provider is called to process the `OIMRecord` value object. The output of this process is the `TargetOperation` value object.
6. The `sendData` method of the provisioning transport provider is called to send the `TargetOperation` value object to the target system.
7. If the provisioning operation is a Create request, the outcome is one of the following events:
 - On successful completion of the Create operation on the target system, the ID field value assigned to the newly created user account is returned by the `sendData` method. This value is passed on to the generic technology connector framework, which posts this value to the database.
 - If the ID field value is not returned, it is assumed that the Create operation has failed. An error message is displayed on the Administrative and User Console.
 - If the operation fails at any stage after the name-value pairs are created, an error message is displayed on the Administrative and User Console.

If the provisioning operation is an Update or Delete request, the ID field is one of the name-value pairs. When this type of provisioning request is sent, the outcome is one of the following events:

- If the operation fails at any stage after the name-value pairs are created, an error message is displayed on the Administrative and User Console.
- On successful completion of the Update or Delete operation, the ID field value may or may not be returned depending on the implementation of the provisioning transport provider.

In either case, the generic technology connector framework does not need the ID field value.

20.2 Creating Custom Providers

The procedure to create custom providers consists of the following steps:

1. [Determining Provider Requirements](#)
2. [Identifying the Provider Parameters](#)
3. [Developing Java Code Implementations of the Value Objects](#)
4. [Developing Java Code Implementations of the Provider SPI Methods](#)
5. [Developing Java Code for Logging and Exception Handling](#)
6. [Creating the Provider XML File](#)
7. [Creating Resource Bundle Entries for the Provider](#)

8. [Deploying the Provider](#)

20.2.1 Determining Provider Requirements

Guidelines for determining provider requirements are as follows:

- [Determining the Reconciliation Provider Requirements](#)
- [Determining the Provisioning Provider Requirements](#)

20.2.1.1 Determining the Reconciliation Provider Requirements

Apply the following guidelines to determine the reconciliation provider requirements:

- If you want to use the target system only as a source of user account information for Oracle Identity Manager, you need only the reconciliation transport provider and reconciliation format provider. You do not need the provisioning transport provider and provisioning format provider.

If you are going to include the reconciliation module in the generic technology connector, you must include both the reconciliation transport provider and the reconciliation format provider, even if you do not need any one of these providers.

The function of the reconciliation format provider is to convert target system data into a format that is supported by Oracle Identity Manager. Even if the target system generates data in a format supported by Oracle Identity Manager, you must include the reconciliation format provider when you create the generic technology connector.

- You must create custom providers only to address provider requirements that are not addressed by the predefined providers.

The types of providers you must include in the generic technology connector depend on the data formats and data transport mechanisms that your target system supports. If any combination of the data formats and data transport mechanisms are compatible with any combination of the predefined providers, you need not create custom providers.

For example, if your target system can generate reconciliation data files in comma-delimited format, you can use the shared drive reconciliation transport provider and CSV reconciliation format provider. You need not create custom reconciliation providers.

See Also: [Section 20.3, "Reusing Providers"](#)

20.2.1.2 Determining the Provisioning Provider Requirements

Apply the following guidelines to determine the provisioning provider requirements for the provisioning module:

- If you want to use the target system only as a target for provisioning operations initiated on Oracle Identity Manager, you need only the provisioning transport provider and provisioning format provider. You do not need the reconciliation transport provider and reconciliation format provider.

If you are going to include the provisioning module in the generic technology connector, you must include both the provisioning transport provider and the provisioning format provider, even if you do not need any one of these providers. This guideline is illustrated by the following example:

The function of the provisioning format provider is to convert Oracle Identity Manager data into a format that is supported by the target system. Even if the

target system supports the output data format of Oracle Identity Manager, you must include the provisioning format provider when you create the generic technology connector.

- You must create custom providers only to address provider requirements that are not addressed by the predefined providers.

The types of providers you must include in the generic technology connector depend on the data formats and data transport mechanisms that your target system supports. If any combination of the data formats and data transport mechanisms are compatible with any combination of the predefined providers, you need not create custom providers.

For example, if your target system is a Web service that can accept and parse SPML-based provisioning requests packaged in SOAP messages, you can use the SPML provisioning format provider and Web Services provisioning transport provider. You need not create custom provisioning providers.

See Also: [Reusing Providers](#) on page 20-15

20.2.2 Identifying the Provider Parameters

Provider parameters are the values that a provider must have in order to perform its intended function. For example, a provisioning transport provider that copies provisioning request files to a target system server will need the connection parameters required to connect to the target system.

While creating a generic technology connector, you specify values for the parameters of the providers that you select for the generic technology connector.

For the custom provider that you are creating, you must identify all the parameters required for the provider to function. You must also categorize these parameters as run-time and design parameters.

A run-time parameter represents a value that is not constrained by the design of the provider. For example, the location of the directories containing data files that you want to reconcile is a run-time parameter. A design parameter represents a value or set of values that is defined as part of the provider design. For example, the character set encoding formats that can be parsed by a reconciliation format provider are a design parameter for that provider.

20.2.3 Developing Java Code Implementations of the Value Objects

Develop the Java code implementations of the value objects listed in [Table 20-1](#). As described earlier, these value objects are used at various stages of provider operations.

Note: You need not develop Java code implementations of the value objects that you are not going to include in the generic technology connector.

Table 20-1 Value Objects Used During Provider Operations

Area of Use	Value Object	Javadocs Package
Metadata Detection	TargetSchema	com.thortech.xl.gc.vo.designtime
	OIMSchema	com.thortech.xl.gc.vo.designtime
Provisioning	TargetOperation	com.thortech.xl.gc.vo.runtime

Table 20–1 (Cont.) Value Objects Used During Provider Operations

Area of Use	Value Object	Javadocs Package
Reconciliation	TargetRecord	com.thortech.xl.gc.vo.runtime
	OIMRecord	com.thortech.xl.gc.vo.runtime

20.2.4 Developing Java Code Implementations of the Provider SPI Methods

Develop the Java code implementations of the SPI methods that are used during provider operations. As described earlier, these SPI methods are called at various stages of provider operations. See the Package `com.thortech.xl.gc.spi` page of the Javadocs for links to information about the SPI methods of each provider.

Note: You need not develop Java code implementations of SPI methods for the providers that you are not going to include in the generic technology connector.

20.2.5 Developing Java Code for Logging and Exception Handling

Oracle recommends that you to incorporate logging in the Java code implementations of the value objects and SPI methods. By doing this, you can simplify troubleshooting errors that may occur when you use the providers.

The logging modules for the generic technology connector framework are an extension of the logging functionality of Oracle Identity Manager. [Table 20–2](#) lists the modules that are specific to the supported provider types.

Table 20–2 Logging Modules Specific to the Supported Provider Types

Logging Module	Functional Module of the Generic Technology Connector Framework
XELLERATE.GC.PROVIDER.PROVISIONINGFORMAT	Provisioning format provider
XELLERATE.GC.PROVIDER.PROVISIONINGTRANSPORT	Provisioning transport provider
XELLERATE.GC.PROVIDER.RECONCILIATIONTRANSPORT	Reconciliation transport provider
XELLERATE.GC.PROVIDER.RECONCILIATIONFORMAT	Reconciliation format provider
XELLERATE.GC.PROVIDER.TRANSFORMATION	Transformation Provider
XELLERATE.GC.PROVIDER.VALIDATION	Validation Provider

See *Oracle Fusion Middleware Java API Reference for Oracle Identity Manager* for information about incorporating exception handling in the custom provider.

20.2.6 Creating the Provider XML File

The provider XML file contains the data required to register the provider with the generic technology connector framework. You must create the provider XML file. [Table 20–3](#) describes the elements that you can include in the provider XML files for custom providers.

Note: You can use a single provider XML file to define any number of providers. Alternatively, you can create a provider XML file for each provider that you create.

You must ensure that the provider XML file adheres to the schema definition provided in MDS. The locations for schema and provider definition XML files are as follows:

```
PROVIDER_DEF_XSD_LOCATION = "/db/GTC/Schema";
```

```
PROVIDER_DEF_XML_LOCATION = "/db/GTC/ProviderDefinitions";
```

Table 20-3 Elements of the Provider XML File

Element	Description
Provider	Root element of the provider XML file
Reconciliation	Parent element of the configuration elements that are used to describe reconciliation providers
Provisioning	Parent element of the configuration elements that are used to describe provisioning providers
Transformation	Parent element of the configuration elements that are used to describe transformation providers
Validation	Parent element of the configuration elements that are used to describe validation providers
ReconTransportProvider	Parent element of the configuration elements that are used to describe a reconciliation transport provider This element has the following attributes: name: Name of the provider class: Name of the Java class of the provider implementation
ReconFormatProvider	Parent element of the configuration elements that are used to describe a reconciliation format provider This element has the following attributes: name: Name of the provider class: Name of the Java class of the provider implementation
ProvFormatProvider	Parent element of the configuration elements that are used to describe a provisioning format provider This element has the following attributes: <ul style="list-style-type: none"> ■ name: Name of the provider ■ class: Name of the Java class of the provider implementation
ProvTransportProvider	Parent element of the configuration elements that are used to describe a provisioning transport provider This element has the following attributes: <ul style="list-style-type: none"> ■ name: Name of the provider ■ class: Name of the Java class of the provider implementation
TransformationProvider	Parent element of the configuration elements that are used to describe a transformation provider This element has the following attributes: <ul style="list-style-type: none"> ■ name: Name of the provider ■ class: Name of the Java class of the provider implementation
ValidationProvider	Parent element of the configuration elements that are used to describe a validation provider This element has the following attributes: <ul style="list-style-type: none"> ■ name: Name of the provider ■ class: Name of the Java class of the provider implementation

Table 20–3 (Cont.) Elements of the Provider XML File

Element	Description
Configuration	Parent element of the configuration elements of any type of provider
Parameter	<p>Element that provides information about a parameter of a provider</p> <p>The <code>Parameter</code> element has the following attributes:</p> <ul style="list-style-type: none"> ■ <code>type</code>: Type of parameter, either <code>Runtime</code> or <code>DesignTime</code> ■ <code>datatype</code>: Data type of parameter, either <code>String</code> or <code>Boolean</code>. Any parameter whose data type is not <code>Boolean</code> must be represented as a string. ■ <code>required</code>: Specifies whether or not the parameter is mandatory ■ <code>encrypted</code>: Specifies whether or not the parameter display must be encrypted ■ <code>name</code>: Name of the parameter ■ <code>dataLength</code>: Character length of the parameter value
DefaultAttribute	<p>Child element of the <code>Configuration</code> element</p> <p>This element must be included only in the <code>ProvFormatProvider</code> element. It has the following attributes:</p> <ul style="list-style-type: none"> ■ <code>datatype</code>: Data type of parameter, either <code>String</code> or <code>Boolean</code>. Any parameter whose data type is not <code>Boolean</code> must be represented as a string. ■ <code>name</code>: Name of the parameter ■ <code>encrypted</code>: Specifies whether or not the parameter display must be encrypted ■ <code>size</code>: Size of the default attribute <p>Some data attributes included in the provisioning request are essential for the provisioning operation to be successfully completed. Because the provisioning format provider generates the final provisioning input data format, the definition of this provider must include these mandatory data attributes. Therefore, if such attributes are required by a target system, they must be defined by using the <code>DefaultAttribute</code> element in the provisioning format provider XML file.</p>
Response	<p>Child element of the <code>Configuration</code> element</p> <p>This element must be included only in the <code>ProvFormatProvider</code>, <code>ProvTransportProvider</code>, <code>TransformationProvider</code>, and <code>ValidationProvider</code> elements. It represents the response returned from the providers that are called by the provisioning engine. This response is displayed on the Oracle Identity Manager Administrative and User Console. This element has the following attributes:</p> <ul style="list-style-type: none"> ■ <code>code</code>: Corresponds to the Oracle Identity Manager process task response code to be generated ■ <code>description</code>: Corresponds to the description of the Oracle Identity Manager process task response code to be generated <p>Note:</p> <p>For a provisioning format provider or provisioning transport provider, you must ensure that the sum of the number of characters of the <code>name</code> attribute of the <code>ProvFormatProvider</code> or <code>ProvTransportProvider</code> element and the number of characters of the <code>Response</code> element is less than or equal to 70. If the sum of the number of characters exceeds 70, the response code cannot be stored in the database and an error is thrown.</p>

See Also:

- ["Step 2: Uploading Request Datasets into MDS"](#) on page 23-24 for information about uploading data to MDS
- [Chapter 33, "MDS Utilities and User Modifiable Metadata Files"](#) for detailed information about the MDS utilities used for importing and exporting files and modifying Oracle Identity Manager metadata

20.2.7 Creating Resource Bundle Entries for the Provider

A resource bundle is a file containing locale-specific text strings. At run time, Oracle Identity Manager reads these text strings and displays them as GUI element labels and messages on the Administrative and User Console. The file extension of a resource bundle is `.properties`.

During installation of Oracle Identity Manager, resource bundles for each of the supported languages are copied to the Oracle Identity Manager server. These include the resource bundles for the predefined providers.

For a custom provider, you must create resource bundle entries for each locale that you plan to use. The following is a summary of the steps involved in creating a resource bundle:

1. Open a new file in a text editor.
2. In this file, create entries for the following text strings:

Note: ■ In the resource bundle, you must provide localized text for the part of each line that follows the equal sign (=).

- The `Provider_Type`, `Parameter_Name`, and `RESPONSE_CODE` values mentioned in this section must be the same as the values you specify in the provider XML file while performing the procedure described in [Section 20.2.6, "Creating the Provider XML File"](#).
-
-

- **Provider names**

The format for provider names is as follows:

```
gc.provider.Provider_Type.Provider_Name=Label_string_in_the_required_language
```

The following is an English-language example of the provider name entry for a provisioning format provider:

```
gc.provider.ProvFormatProvider.SPML=SPML
```

- **Provider parameter labels and description**

The format for provider parameter labels and parameter descriptions is as follows:

```
gc.Provider_Type.Provider_Name.Parameter_Name.label=Parameter_label_in_the_required_language
gc.Provider_Type.Provider_Name.Parameter_Name.description=Parameter_description_in_the_required_language
```

The following is an English-language example of the provider parameter label and parameter description entries for a provisioning format provider:

```
gc.ProvFormatProvider.SPML.targetID.label=Target ID
gc.ProvFormatProvider.SPML.targetID.description=Target ID of the
provisioning target
```

- Response codes and descriptions

The format for response codes and descriptions is as follows:

```
GC.GCPROV.PROVIDER_TYPE.PROVIDER_NAME.RESPONSE_CODE=Response_code_in_the_re
quired_language
GC.GCPROV.PROVIDER_TYPE.PROVIDER_NAME.RESPONSE_CODE.description=Description
_in_the_required_language
```

The following is an English-language example of the response code and description entries for a provisioning format provider:

```
GC.GCPROV.ProvFormatProvider.SPML.SPML_VELOCITY_PROPERTIES_NOT_READ=SPML
Velocity Properties Not Read
GC.GCPROV.ProvFormatProvider.SPML.SPML_VELOCITY_PROPERTIES_NOT_READ.descrip
tion=Necessary SPML template properties could not be read.
```

- Metadata detection error messages

The format for metadata detection error messages is as follows:

```
gc.error.Provider_Type.Provider_Name.ERROR_CODE=Error_Description
```

Here, *ERROR_CODE* must be the same as the value of the *errorCode* string passed as an argument to the constructor of the exception class. For example, the following is one of the constructors of the *ReconciliationTransportException* class:

```
ReconciliationTransportException(java.lang.String errorCode,
java.lang.String isMessage)
```

You must add lines in the resource bundle for all possible values of the *errorCode* string.

The following is an English-language example of the metadata detection error message for a reconciliation transport provider:

```
gc.error.ReconTransportProvider.SharedDrive.NO_READ_FILE=There are no
readable files to detect metadata.
```

3. Save and close the resource bundle.

20.2.8 Deploying the Provider

To deploy the provider:

1. Deploy the JAR files as follows:
 - a. Compile and package all the Java code files in a JAR file.
 - b. Copy the JAR file into the following directory:

```
OIM_HOME/JavaTasks
```

In addition, upload the JAR files to Oracle Identity Manager database. See ["Deploying the Custom Providers"](#) on page 20-17 for details.

Note: In a clustered environment, you must copy each file that you create to the corresponding directory on each node of the cluster.

2. Deploy the provider XML files as follows:
 - a. Upload the provider XML file to MDS at the following location:
PROVIDER_DEF_XML_LOCATION = "/db/GTC/ProviderDefinitions":
 - b. Restart Oracle Identity Manager.
 - c. To check if the provider XML file has been correctly registered:
 - i. Log in to the Administrative and User Console.
 - ii. Expand **Generic Technology Connector**, and click **Create**. If there are any errors in the provider XML file, an error message is displayed.

If an error message is displayed, fix the problem in the XML file, restart Oracle Identity Manager, and repeat Steps i and ii.

Repeat this procedure until no error messages are displayed when you click Create.
3. Deploy the provider resource bundles as follows:
 - a. Upload the resource bundles to the MDS by using the UploadResourceBundles.sh utility present in the OIM_HOME/bin/ directory. See "[Upload Resource Bundle Utility](#)" on page 35-4 for information about running the utility.
 - b. For the new resource bundle entries to take effect, either run the PurgeCache script or restart the application server. See "Purging the Cache" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager* for information about running the PurgeCache utility.

20.3 Reusing Providers

Format providers and transport providers work in pairs. During reconciliation, the output of the reconciliation transport provider is passed on to the reconciliation format provider. During provisioning, the output of the provisioning format provider is passed on to the provisioning transport provider. This means that the implementation of the transport providers and format providers is linked through the implementation of the value objects that are passed between them. This dependency forms the basis of guidelines on reusing format providers and transport providers.

In contrast, a validation provider or transformation provider does not have any such dependency on other providers. Therefore, there are no guidelines on reusing validation Providers and transformation Providers. You can reuse both predefined and custom transformation and validation providers, because their action is not specific to the data format or data transport mechanism of the target system.

Guidelines on reusing format providers and transport providers are dividing into the following sections:

- [Reusing Reconciliation Providers](#)
- [Reusing Provisioning Providers](#)

20.3.1 Reusing Reconciliation Providers

As described in [Section 20.1.2, "Role of Providers During Reconciliation"](#), the `TargetRecord` value object is used to exchange data between the reconciliation transport provider and the reconciliation format provider. The reconciliation transport provider creates an array of `TargetRecord` value objects for the target system records fetched during reconciliation. The reconciliation format provider processes the data in the value objects array and passes it on to the reconciliation engine.

Suppose the operating environment of your organization contains two target systems, TS1 and TS2. TS1 offers a Web-based interface for extracting data during reconciliation. TS2 provides APIs for enabling other applications to read data from its identity store. Both target systems support the same data format. If you want to reconcile user data from both target systems, you must create one generic technology connector for each target system. For each generic technology connector, you must create a reconciliation transport provider. However, instead of creating a reconciliation format provider for each generic technology connector, you can create and reuse a single reconciliation format provider. Similarly, if TS1 and TS2 supported the same data transport mechanism (even if they do not support the same data format), you can reuse the reconciliation transport provider and create different reconciliation format providers.

If you want to reuse a reconciliation transport provider, you must ensure that the implementation of the `TargetRecord` value object does not contain code that is specific to the function performed by the reconciliation format provider. If you want to reuse a reconciliation format provider, you must ensure that the implementation of the `TargetRecord` value object does not contain code that is specific to the function performed by the reconciliation transport provider.

Reusing the Predefined Reconciliation Providers

The implementation of the shared drive reconciliation transport provider and CSV reconciliation format provider is such that these predefined providers are built for a fixed combination of data formats and a single data transport mechanism. The shared drive reconciliation transport provider reads data from flat files and passes an array of `TargetRecord` value objects to the CSV reconciliation format provider.

Implementation of the shared drive reconciliation transport provider is based on two factors: paged reconciliation and multivalued (child) data reconciliation. These factors require the provider to be able to parse target system data before it can create an array of `TargetRecord` value objects. In other words, the ability of the shared drive reconciliation transport provider to parse certain types of target system data and the ability of the CSV reconciliation format provider to use only the output provided by the shared drive reconciliation transport provider makes them interdependent. Therefore, the parameters of the CSV reconciliation format provider are bundled along with those of the shared drive reconciliation transport provider.

For this reason, you cannot use the shared drive reconciliation transport provider with a custom reconciliation format provider and you cannot use the CSV format provider with a custom reconciliation transport provider.

20.3.2 Reusing Provisioning Providers

As described in [Section 20.1.3, "Role of Providers During Provisioning"](#), the `TargetOperation` value object is used to exchange data between the provisioning transport provider and the provisioning format provider. The provisioning format provider creates a `TargetOperation` value object out of the provisioning data to be

sent to the target system. The provisioning transport provider passes this value object to the target system.

Suppose the operating environment of your organization contains two target systems, TS1 and TS2. TS1 offers a Web-based interface for accepting provisioning request data. TS2 provides APIs for enabling provisioning data to be written to the identity store. Both target systems support the same data format. If you want to perform provisioning operations on both target systems, you must create one generic technology connector for each target system. For each generic technology connector, you must create a provisioning transport provider. However, instead of creating a provisioning format provider for each generic technology connector, you can create and reuse a single provider.

If TS1 and TS2 supported the same data transport mechanism but different data formats, you can reuse the provisioning transport provider and create different provisioning format providers.

If you want to reuse the provisioning transport provider, you must ensure that the implementation of the `TargetOperation` value object does not contain code that is specific to the function performed by the provisioning format provider. If you want to reuse the provisioning format provider, you must ensure that the implementation of the `TargetOperation` value object does not contain code that is specific to the function performed by the provisioning transport provider.

Reusing the Predefined Provisioning Providers

If the target system is a Web service, you can use the Web Services provisioning transport provider along with any custom provisioning format provider that you create. This is illustrated by the following example:

As mentioned earlier in this guide, the SPML provisioning format provider supports only a subset of the provisioning operations that are described in the SPML specification. You can develop a custom provisioning format provider that supports all the SPML provisioning operations. If the target system is a Web service, you can use the Web Services provisioning transport provider to carry SPML requests from your custom provisioning format provider to the target system.

Similarly, you can use the SPML provisioning format provider along with a custom provisioning transport provider to send SPML requests to an SPML-based target system.

The following is the implementation of the `TargetOperation` value object that is created by the SPML provisioning format provider and used as an input for the Web Services provisioning transport provider:

```
com.thortech.xl.gc.impl.prov.WSTransportTargetOperation
```

See *Oracle Fusion Middleware Java API Reference for Oracle Identity Manager* for information about this class.

If you want to reuse the SPML provisioning format provider, you must create a custom transport provider that can accept an instance of this class as input and call the relevant `set` method. Similarly, if you want to reuse the Web Services provisioning transport provider, you must create a custom provisioning format provider that can create an instance of this class and call the relevant `get` method.

20.4 Deploying the Custom Providers

To deploy the custom providers:

1. Upload the Provider definition XML file to the MDS location /db/GTC/ProviderDefinitions. Oracle Identity Manager provides utilities to export/import data to and from MDS repository.
See [Chapter 33, "MDS Utilities and User Modifiable Metadata Files"](#) for information about the MDS utilities.
2. The provider resource bundles and JAR files need to be uploaded to Oracle Identity Manager database. Utilities are available in *OIM_HOME/bin/* directory for uploading resource bundles and JAR files to Oracle Identity Manager database.
See [Chapter 35, "Upload JAR and Resource Bundle Utilities"](#) for information about the upload resource bundles and JAR utilities.

Creating and Managing Generic Technology Connectors

This chapter explains how to create and maintain Generic Technology Connectors. It contains these sections:

- [Overview](#)
- [Creating Generic Technology Connectors](#)
- [Managing Generic Technology Connectors](#)
- [Using the Generic Connection Pool Framework in Custom Connectors](#)
- [Best Practices](#)

21.1 Overview

Providers are the starting point for developing generic technology connectors. Oracle Identity Manager provides a standard set of providers that you can use as building blocks of your generic technology connectors. For details about these providers, see [Chapter 19, "Predefined Providers for Generic Technology Connectors"](#).

If no suitable provider is available, you can develop a provider to fit your requirements. For details, see [Chapter 20, "Creating Custom Providers for Generic Technology Connectors"](#).

Finally, if generic technology connectors do not meet your integration requirements, you can make use of the programmatic options available with adapters. For details, see [Part V, "Requests and Approval Processes"](#).

21.2 Creating Generic Technology Connectors

This section explains how to create generic technology connectors.

The procedure to create a generic technology connector is composed of the following steps:

- [Determining Provider Requirements](#)
- [Selecting the Providers to Include](#)
- [Addressing the Prerequisites](#)
- [Using the Administrative and User Console to Create the Connector](#)
- [Configuring Reconciliation](#)
- [Configuring Provisioning](#)

- [Enabling Logging](#)

21.2.1 Determining Provider Requirements

As mentioned in [Chapter 19, "Predefined Providers for Generic Technology Connectors"](#), the following providers can be used as the building blocks of the generic technology connectors you create:

- Reconciliation Transport Provider
- Reconciliation Format Provider
- Provisioning Transport Provider
- Provisioning Format Provider
- Transformation Provider
- Validation Provider

See [Section 18.2, "Functional Architecture of Generic Technology Connectors"](#) for the definitions of these providers. Then, based on your knowledge of the data formats and data transport mechanisms supported by the target system, identify the providers that must be included in the generic technology connector that you create. If the target system supports multiple data formats and data transport mechanisms, you must select a single combination of the transport and format providers discussed in the first chapter. You cannot include, for example, multiple reconciliation format providers in a single generic technology connector.

See Also: [Section 21.2.1, "Determining Provider Requirements"](#)

21.2.2 Selecting the Providers to Include

Identify the predefined providers that can be used to meet your provider requirements. See [Chapter 19, "Predefined Providers for Generic Technology Connectors"](#) for information about the predefined providers.

If all your provider requirements are addressed by the predefined providers, you need not create custom providers. You must create custom providers to address only the requirements that are not addressed by the predefined providers. See [Chapter 20, "Creating Custom Providers for Generic Technology Connectors"](#) for information about creating custom providers.

21.2.3 Addressing the Prerequisites

You must address the following prerequisites:

- If you are creating the generic technology connector on a production server, enable the cache for the following cache categories:
 - `GenericConnector`
 - `GenericConnectorProviders`

See "Tuning and Managing Application Cache" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager* for more information.

- Testing connectivity between the target system server and the Oracle Identity Manager server

You must take steps to ensure that connectivity can be established between the target system server and the Oracle Identity Manager server. For example, in a

UNIX environment, you must enter the fully qualified host name of the Oracle Identity Manager server in the `/etc/hosts` file on the target system server.

- Creating the user account to be used for creating the generic technology connector

All users belonging to the `SYSTEM ADMINISTRATORS` group of Oracle Identity Manager can create generic technology connectors. Alternatively, members of a group to which you assign the required menu items and permissions can create generic technology connectors.

The required menu items are as follows:

- Create Generic Technology Connector menu item
- Manage Generic Technology Connector menu item

The required permissions are as follows:

- Form Designer (Allow Insert, Write Access, Delete Access)
- Structure Utility.Additional Column (Allow Insert, Write Access, Delete Access)
- Meta-Table Hierarchy (Allow Insert, Write Access, Delete Access)

If these permissions are not correctly assigned to the group, an error is thrown when the user clicks the Create button on the final Administrative and User Console page for creating generic technology connectors.

21.2.4 Using the Administrative and User Console to Create the Connector

To navigate to the first Administrative and User Console page for creating a generic technology connector, login to the Administrative and User Console, click Advanced, and under Configuration, click **Create Generic Connector**.

From this point onward, page-wise instructions are provided in the following sections:

- [Step 1: Provide Basic Information Page](#)
- [Step 2: Specify Parameter Values Page](#)
- [Step 3: Modify Connector Configuration Page](#)
- [Step 4: Verify Connector Form Names Page](#)
- [Step 5: Verify Connector Information Page](#)

21.2.4.1 Step 1: Provide Basic Information Page

To provide basic information about the generic technology connector that you want to create, use this page as follows

1. In the **Name** field, specify a name for the generic technology connector.

The following are guidelines related to selecting a name for the generic technology connector:

- The name must not be the same as that of any other connector (predefined connector or generic technology connector) on this Oracle Identity Manager installation.
- The name must not be the same as that of any other connector object (such as resource objects, IT resources, and process forms) on this Oracle Identity Manager installation.

Note: An error message is displayed if you specify a name that is the same as the name of an existing connector. However, an error message is *not* displayed if you specify a name that is the same as the name of an existing connector object. Therefore, you must ensure that the name you want to specify is not the same as the name of any existing connector object.

See [Section 18.4, "Connector Objects Created by the Generic Technology Connector Framework"](#) for more information about connector objects that are automatically created as part of the generic technology connector creation process.

- The name must not contain non-ASCII characters, because Oracle Identity Manager does not support non-ASCII characters in connector names. However, you can include the underscore character (`_`) in the name.

See Also: [Section 22.2.1, "Names of Generic Technology Connectors and Connector Objects"](#) for information about limitations related to the names of generic technology connectors.

2. If you want to use the generic technology connector for reconciliation, select **Reconciliation** and perform the following steps:

- From the **Transport Provider** list, select the reconciliation transport provider that you want to use for this connector. This list displays the predefined reconciliation transport providers and the reconciliation transport providers that you create.
- From the **Format Provider** list, select the reconciliation format provider that you want to use for this connector. This list displays the predefined reconciliation format providers and the reconciliation format providers that you create.

Note: If you select the shared drive reconciliation transport provider, you must also select the CSV reconciliation format provider because all the parameters of this provider are bundled with the parameters of the shared drive reconciliation transport provider.

- If you want to use the connector to perform trusted source reconciliation with the target system, select **Trusted Source Reconciliation**.

Note: If you select the Trusted Source Reconciliation check box, the Provisioning region of the page is disabled. This is because you cannot provision to a target system that you designate as a trusted source. You can only reconcile data from a trusted source.

3. If you want to use the generic technology connector for provisioning, select **Provisioning** and perform the following steps:

Note: You can select only Reconciliation, only Provisioning, or both Reconciliation and Provisioning.

- From the **Transport Provider** list, select the provisioning transport provider that you want to use for this connector. This list displays the predefined provisioning transport providers and the provisioning transport providers that you create.

If you select the Web Services provisioning transport provider and if Secure Sockets Layer (SSL) is enabled for the target Web service, you must perform the procedure described in [Section 19.4.1, "Configuring SSL Communication Between Oracle Identity Manager and the Target System Web Service"](#).

- From the **Format Provider** list, select the provisioning format provider that you want to use for this connector. This list displays the predefined provisioning format providers and the provisioning format providers that you create.

If you select the SPML provisioning format provider, you must also select the Web Services provisioning transport provider because the parameters of this provider are related to the parameters of the Web Services provisioning transport provider.

4. Click Continue.

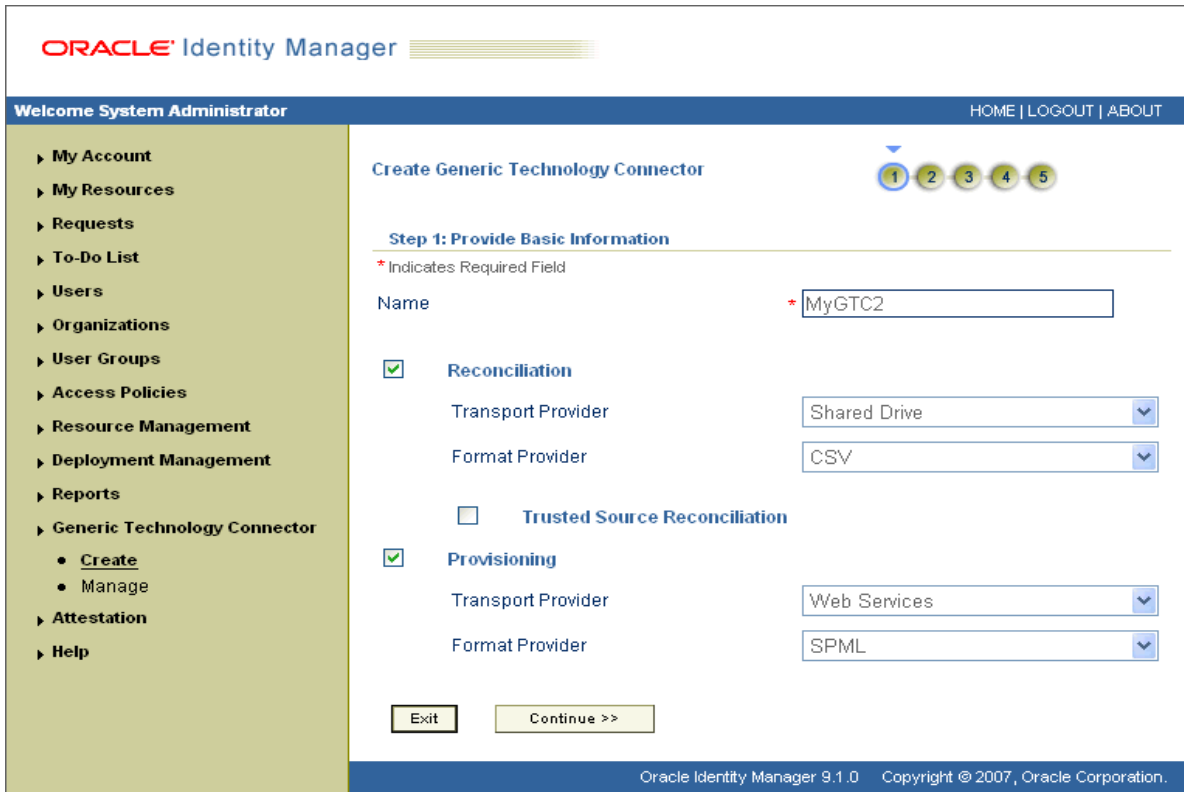
[Table 21-1](#) lists sample entries for the GUI elements on the Step 1: Provide Basic Information page.

Table 21-1 Sample Entries for the Step 1: Provide Basic Information Page

Label on the Step 1: Provide Basic Information Page		
Label on the Step 1: Provide Basic Information Page	Sample Value or Action	Reference Information
Name field	MyGTC2	NA
Reconciliation check box	Check box selected	NA
Transport Provider list	Shared Drive	shared drive reconciliation transport provider
Format Provider list	CSV	CSV Reconciliation format provider
Provisioning check box	Check box selected	NA
Transport Provider list	Web Services	Web Services provisioning transport provider
Format Provider list	SPML	SPML provisioning format provider

[Figure 21-1](#) shows the Step 1: Provide Basic Information page on which the entries described in [Table 21-1](#) have been made.

Figure 21–1 Step 1: Provide Basic Information Page



21.2.4.2 Step 2: Specify Parameter Values Page

Use this page to specify values for the parameters of the providers that you select on the Step 1: Provide Basic Information page.

On this page, the provider parameters are divided into two categories:

- Run-time parameters

See Also: [Chapter 19, "Predefined Providers for Generic Technology Connectors"](#) for detailed information about the run-time parameters of predefined providers that you select on the Step 1: Provide Basic Information page

Run-time parameters are input variables of the providers that you select on the previous page. A run-time parameter represents a value that is not constrained by the design of the provider. For example, the location of the directories containing the data files that you want to reconcile is a run-time parameter.

- Design parameters

The parameters listed in this section are either design parameters of providers or reconciliation-specific parameters that are common to all generic technology connectors. A design parameter represents a value or set of values that is defined as part of the provider design.

See Also: [Chapter 19, "Predefined Providers for Generic Technology Connectors"](#) for detailed information about the design parameters of predefined providers that you select on the Step 1: Provide Basic Information page

For example:

The format of data files that can be parsed by a format provider is a design parameter for that provider. While designing the provider, you define the set of formats the provider can parse. On the Step 2: Specify Parameter Values page, you specify the particular format (from the set of supported formats) that an instance of the format provider must parse.

The following are reconciliation-specific design parameters:

Note: If you do not select the Reconciliation option on the previous page, these reconciliation-specific design parameters are not displayed on this page.

– **Batch Size**

Use this parameter to specify a batch size for the reconciliation run. By using this parameter, you can break into batches the total number of records that the reconciliation engine fetches from the target system during each reconciliation run.

The default value of this parameter is All.

– **Stop Reconciliation Threshold**

During reconciliation, data from the reconciliation format provider is accepted as input by the validation provider. Some of the reconciliation data records may not clear the validation checks. You can use the Stop Reconciliation Threshold parameter to automatically stop reconciliation if the percentage of records that fail the validation checks to the total number of reconciliation records processed exceeds the specified value.

The following example illustrates how this parameter works:

Suppose you specify 20 as the value of the Stop Reconciliation Threshold parameter. This means that you want reconciliation to stop if the percentage of failed records to the total number of records processed becomes equal to or greater than 20. Suppose the second and eighth records fail the validation checks. At this stage, the number of failed records is 2 and the total number of records processed is 8. The percentage of failed records is 25, which is greater than the specified threshold of 20. Therefore, reconciliation is stopped after the eighth record is processed.

Note:

- The Stop Reconciliation Threshold parameter is used during reconciliation only if you select validation Providers on the Step 3: Modify Connector Configuration page.
 - If reconciliation is stopped because the actual percentage of failed records exceeds the specified percentage, the records that have already been reconciled into Oracle Identity Manager are not removed.
-
-

The default value of this parameter is None. This default value specifies that during a reconciliation run, you want all the target system records to be processed, regardless of the number of records that fail the checks.

– **Stop Threshold Minimum Records**

If you use the Stop Reconciliation Threshold parameter, there may be a problem if invalid records are encountered right at the beginning of the reconciliation run. For example, suppose you specify 40 as the value of the Stop Reconciliation Threshold parameter. When reconciliation starts, suppose the first record fails the validation checks. At this stage, the percentage of failed records to total records processed is 100. Therefore, reconciliation would stop immediately after the first record is processed.

To avoid such situations, you can use the Stop Threshold Minimum Records parameter in conjunction with the Stop Reconciliation Threshold parameter. The Stop Threshold Minimum Records parameter specifies the number of records that must be processed by the validation provider before the Stop Reconciliation Threshold validation is enabled.

The following example illustrates how this parameter works:

Suppose you specify the following values:

Stop Reconciliation Threshold: 20

Stop Threshold Minimum Records: 80

With these values, from the eighty-first record onward, the Stop Reconciliation Threshold validation is enabled. In other words, after the eightieth record is processed, if any record fails the validation check, the reconciliation engine calculates the percentage of failed records to total records processed.

The default value of this parameter is `None`.

Note:

- The Stop Threshold Minimum Records parameter is used during reconciliation only if you select validation Providers on the Step 3: Modify Connector Configuration page.
 - You must specify a value for the Stop Threshold Minimum Records parameter if you specify a value for the Stop Reconciliation Threshold parameter.
-
-

– **Reconciliation Type**

Use this parameter to specify whether you want the reconciliation engine to perform incremental or full reconciliation.

Note: The outcome of both full and incremental reconciliation is the same: target system records that are created or updated after the last reconciliation run are reconciled into Oracle Identity Manager.

In incremental reconciliation, only target system records that are newly added or modified after the last reconciliation run are brought to Oracle Identity Manager. Reconciliation events are created for each of these records.

In full reconciliation, all target system records are brought to Oracle Identity Manager. The optimized reconciliation feature identifies and ignores records that have already been reconciled. Reconciliation events are created for the remaining records.

You must select incremental reconciliation if either one of the following conditions is true:

- * The target system time stamps or uniquely marks (in some way) files or individual data records that it generates, and the reconciliation transport provider can recognize records that have been time stamped or marked by the target system.

For example:

Suppose the target system can time stamp the creation of or modifications to user data records. If you can create a custom reconciliation transport provider that can read this time-stamp information, only new or modified data records will be transported to Oracle Identity Manager during reconciliation.

- * The target system provides only data records that are newly added or modified after the last reconciliation run.

If *neither* of these conditions is true, you must select full reconciliation.

– **Reconcile Deletion of Multivalued Attribute Data**

Use this parameter to specify whether or not you want to reconcile into Oracle Identity Manager the deletion of multivalued attribute data (child data) on the target system.

The following example explains how this design parameter works:

There is an account for user John Doe on the target system. This user is a member of two user groups, `CREATE_USERS` and `REVIEW_PERMISSIONS`, on the target system. This user account (along with the group membership information) also exists on Oracle Identity Manager.

On the target system, suppose this user is removed from the `REVIEW_PERMISSIONS` group. During the next reconciliation run, the action that will be taken in Oracle Identity Manager depends on whether or not you select the **Reconcile Deletion of Multivalued Attribute Data** check box:

- * If you select the check box, information about this user being a member of the `REVIEW_PERMISSIONS` group on the target system is removed from the Oracle Identity Manager database. All other changes made to this user account on the target system are also reconciled.
- * If you do not select the check box, information about this user being a member of the `REVIEW_PERMISSIONS` group on the target system is *not* removed from the Oracle Identity Manager database. However, all other changes made to this user account on the target system are reconciled.

– **Source Date Format**

Use this parameter to specify the format in which date values are stored in the target system.

The format that you specify is used to validate date values fetched during reconciliation and to convert the date values to the format used internally by Oracle Identity Manager.

The Validate Date Format provider is one of the predefined validation providers. During a reconciliation run, the Validate Date Format provider uses the source date format to validate date values fetched from the target system. Only date values that match the source date format are converted to the date format used by Oracle Identity Manager and reconciled. This format

validation and conversion applies to all date fields (for example, Date of Birth and Hire Date) of the target system.

See Also: "[Validation Providers](#)" on page 19-21 for more information about validation providers

For information about the date formats that you can specify, see the following page on the Sun Java Web site:

<http://java.sun.com/docs/books/tutorial/i18n/format/simpleDateFormat.html>

Note: If you want the source date format to be used in date validation, while performing the procedure described in [Section 21.2.4.3.1, "Adding or Editing Fields in Data Sets"](#), you must:

- Map date fields of the Source data sets to date fields of the reconciliation staging data sets.
 - Edit each date field of the reconciliation staging data sets and set its data type to the Date data type.
-
-

The default value of the Source Date Format parameter is the date format specified as the value of the `XL.DefaultDateFormat` system property. If you do not specify a value for the Source Date Format parameter, the default date format is used for date validation during reconciliation.

See Also: "System Properties in Oracle Identity Manager" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager* for information about the system properties of Oracle Identity Manager

The following example illustrates how the Source Date Format parameter is used:

Suppose the following are date values in the target system:

- Date 1: 05/04/2007 06:25:44 PM
- Date 2: 05/06/2007 07:31:44 PM
- Date 3: Thu, Apr 9, '98
- Date 4: 07/03/2008 02:15:55 PM

Scenario 1:

While creating the connector, you had entered the following as the value of the Source Date Format parameter:

`MM/dd/yyyy hh:mm:ss a`

During a reconciliation run, the record containing the Date 3 value is not reconciled because it does not conform to the specified source date format.

Scenario 2:

While creating the connector, you had not entered a value for the Source Date Format parameter. Therefore, during a reconciliation run, all four records are validated against the date format specified as the value of the `XL.DefaultDateFormat` system property.

The following is a provisioning-specific design parameter:

Note: If you do not select the Provisioning option on the previous page, this provisioning-specific design parameter is not displayed.

- **Target Date Format**

Use this parameter to specify the format in which you want to send date values to the target system during provisioning operations.

During a provisioning operation, date values are converted to the format that you specify as the value of the Target Date Format parameter. This format conversion applies to all date fields (for example, Date of Birth and Hire Date) that are used in the provisioning operation.

For information about the date formats that you can specify, see the following page on the Sun Java Web site:

<http://java.sun.com/docs/books/tutorial/i18n/format/simpleDateFormat.html>

If you do not specify a date format, the following date format is used as the default value of this parameter:

```
yyyy/MM/dd hh:mm:ss z
```

The following example illustrates how the Target Date Format parameter is used:

During a provisioning operation, any date value that you enter will be in the `yyyy/MM/dd hh:mm:ss z` format.

Scenario 1:

While creating the connector, you had entered the following as the value of the Target Date Format parameter:

```
yyyy.MM.dd G 'at' hh:mm:ss z
```

During a provisioning operation, an Oracle Identity Manager date value (for example, 2007/05/04 06:25:44 IST) will be converted into the target date format (for example, 2007.05.04 AD at 06:25:44 IST) and sent to the target system.

Scenario 2:

While creating the connector, you had not entered a value for the Target Date Format parameter. During a provisioning operation, date values are sent to the target system in the (default) `yyyy/MM/dd hh:mm:ss z` format.

After you specify values for the run-time and design parameters, click **Continue**.

Note: If any value that you provide on this page is not correct, an error message is displayed at the top of the page after you click **Continue**. If this happens, fix the parameter value and click **Continue** again.

Table 21–2 lists sample entries for the Step 2: Specify Parameter Values page. The GUI elements displayed on this page are based on the entries made on the Step 1: Provide Basic Information page.

Table 21–2 Sample Entries for the Step 2: Specify Parameter Values Page

Label on the Step 2: Specify Parameter Values Page	Sample Value or Action	Reference Information
Run-Time Parameters of the Shared Drive Reconciliation Transport Provider		
Staging Directory (Parent Identity Data) field	D:\gctestdata\commaDelimited\parent	NA
Staging Directory (Multivalued Identity Data) field	D:\gctestdata\commaDelimited\child	NA
Archiving Directory field	D:\gctestdata\commaDelimited\archive	NA
File Prefix field	file	NA
Specified Delimiter field	,	NA
Tab Delimiter check box	Check box not selected	NA
Fixed Column Width field		NA
Unique Attribute (Parent Data) field	UserIDTD	NA
Run-Time Parameter of the Web Services Provisioning Transport Provider		
Web Service URL field	http://acme123:8080/spmlws/services/HttpSoap11	NA
Run-Time Parameters of the SPML Provisioning Format Provider		
Target ID field	target	NA
User Name (authentication) field	xelsysadm	NA
User Password (authentication) field		NA
Design Parameters of the Shared Drive Reconciliation Transport Provider		
File Encoding field	Cp1251	NA
Design Parameters of the Web Services Provisioning Transport Provider		
Web Service SOAP Action field	http://xmlns.oracle.com/OIM/provisioning//processRequest	NA
Design Parameters of the SPML Provisioning Format Provider		
WSSE Configured for SPML Web Service? check box	Check box not selected	NA
Custom Authentication Credentials Namespace field	http://xmlns.oracle.com/OIM/provisioning	NA
Custom Authentication Header Element field	OIMUser	NA
Custom Element to Store User Name field	OIMUserId	NA
Custom Element to Store Password field	OIMUserPassword	NA

Table 21–2 (Cont.) Sample Entries for the Step 2: Specify Parameter Values Page

Label on the Step 2: Specify Parameter Values Page	Sample Value or Action	Reference Information
SPML Web Service Binding Style (DOCUMENT or RPC) field	RPC	NA
SPML Web Service Complex Data Type field		NA
SPML Web Service Operation Name field	processRequest	NA
SPML Web Service Target Namespace field	http://xmlns.oracle.com/OIM/provisioning	NA
SPML Web Service Soap Message Body Prefix field		NA
ID Attribute for Child Dataset Holding Group Membership Information field		NA
Generic Design Parameters		NA
Target Date Format field	yyyy-MM-dd hh:mm:ss.ffffff	NA
Batch Size field	All	NA
Stop Reconciliation Threshold field	None	NA
Stop Threshold Minimum Records field	None	NA
Source Date Format field	yyyy/MM/dd hh:mm:ss z	NA
Reconcile Deletion of Multivalued Attribute Data check box	Check box selected	NA
Reconciliation Type list	Incremental	NA

Figure 21–2 shows the first section of the Step 2: Specify Parameter Values page on which the entries listed in Table 21–2 have been made.

Figure 21–2 First Section of the Step 2: Specify Parameter Values Page

ORACLE Identity Manager

Welcome System Administrator HOME | LOGOUT | ABOUT

Create Generic Technology Connector 1 2 3 4 5

Step 2: Specify Parameter Values

* Indicates Required Field

Run-Time Parameters

Shared Drive

Staging Directory (Parent identity data) This is the directory location in which parent identity data files are stored.

Staging Directory (Multivalued identity data) This is the directory location in which multivalued identity data files are stored.

Archiving Directory This is the directory location in which identity data files are archived after reconciliation.

File Prefix This is the prefix given to the names of the identity data files.

Specified Delimiter This is the delimiter for the file. If it is set, then it overrides all other delimiter settings. The value is a comma (,) for CSV format files.

Tab Delimiter This specifies whether or not tab delimiters are used. If it is set, then it overrides the setting of the Fixed Column Width field.

Fixed Column Width This is the common column width of the identity fields. This value is used if neither delimiter is set.

Unique Attribute (Parent Data) This is the name of the CSV data column that uniquely identifies each parent identity data record.

Web Services

Web Service URL This is the URL for the Web service receptor.

SPML

Target ID ID of the target system for provisioning operations.

User Name (authentication) User name required for authentication by the Web service.

User Password (authentication) Password required for authentication by the target Web service.

Figure 21–3 shows the second section of the Step 2: Specify Parameter Values page on which the entries listed in Table 21–2 have been made.

Figure 21–3 Second Section of the Step 2: Specify Parameter Values Page

Design Parameters

Shared Drive

File Encoding This is the character set encoding used for the data files. Cp1251 is the English language default.

Web Services

Web Service SOAP Action In the WSDL file, this is the value of the "soapAction" attribute of the "operation" element.

SPML

WSSE Configured for SPML Web Service? Specify whether or not the target SPML Web Service is configured to receive WS-Security credentials.

Custom Authentication Credentials Namespace Namespace that defines custom authentication credentials. Specify a value only if the Web service is not configured for WSSE.

Custom Authentication Header Element Name of the header element for the custom authentication section that is to be included in the SOAP header. Specify a value only if the Web service is not configured for WSSE.

Custom Element to Store User Name Name of the element in the custom authentication section that will store the user name required for authentication by the Web service. Specify a value only if the Web service is not configured for WSSE.

Custom Element to Store Password Name of the element in the custom authentication section that will store the password required for authentication by the Web service. Specify a value only if the Web service is not configured for WSSE.

SPML Web Service Binding Style (Document or RPC) In the WSDL file, this is the value of the style attribute of the binding element.

SPML Web Service Complex Data Type In the WSDL file, this is the value of the "name" attribute of the "complexType" element. This parameter is applicable only if the binding style is "Document".

SPML Web Service Operation Name In the WSDL file, this is the value of the "name" attribute of the "operation" element. This parameter is applicable only if the binding style is "RPC".

SPML Web Service Target Namespace In the WSDL file, this is the value of the "targetNamespace" attribute of the "definition" element.

SPML Web Service Soap Message Body Prefix Name of the custom prefix element that contains the soap message body. If the target Web service is running on BEA WebLogic, IBM WebSphere, jBoss Application Server, or OC4J, then you need not specify a value for this parameter.

ID Attribute for Child Dataset Holding Group Membership Information Name of the ID attribute for a Provisioning Staging child dataset holding group membership information.

Target Date Format Date Format supported by the Date attributes of Provisioning Staging Dataset. Default value is "yyyy-MM-dd hh:mm:ss.ffffff".

Figure 21–4 shows the third section of the Step 2: Specify Parameter Values page on which the entries listed in Table 21–2 have been made.

Figure 21–4 Third Section of the Step 2: Specify Parameter Values Page

The screenshot displays the 'Third Section of the Step 2: Specify Parameter Values Page' in Oracle Identity Manager 9.1.0. The page is divided into two columns. The left column contains configuration fields: 'Batch Size' (text box with 'All'), 'Stop Reconciliation Threshold' (text box with 'None'), 'Stop Threshold Minimum Records' (text box with 'None'), 'Source Date Format' (text box with 'yyyy/MM/dd hh:mm:ss z'), 'Reconcile Deletion of Multivalued Attribute Data' (checkbox checked), and 'Reconciliation Type' (dropdown menu with 'Incremental'). The right column contains explanatory text for each field. At the bottom, there are three buttons: 'Exit', '<< Back', and 'Continue >>'. The footer reads 'Oracle Identity Manager 9.1.0 Copyright © 2007, Oracle Corporation.'

21.2.4.3 Step 3: Modify Connector Configuration Page

Use this page to define data sets and mappings between the fields of the data sets. In other words, you use this page to specify the user data fields that you want to:

- Propagate from the target system to Oracle Identity Manager during reconciliation
- Propagate from Oracle Identity Manager to the target system during provisioning

In the generic technology connector context, the term **metadata** refers to the set of identity fields that constitute the user account information on the target system.

First Name, Last Name, Hire Date, and Department ID are examples of user data fields that constitute metadata. The values assigned to these fields constitute the user data on the target system. For example, the identity information of user John Doe on the target system can be composed of the following fields:

- First Name: John
- Last Name: Doe
- Hire Date: 04-December-2007
- Department ID: Sales
- . . .

After you click the **Continue** button on the Step 2: Specify Parameter Values page, the metadata displayed on the Step 3: Modify Connector Configuration page depends on the following factors:

- Input provided on the Step 1: Provide Basic Information and Step 2: Specify Parameter Values pages
- Availability of sample target system data

Note: In the generic technology connector context, the term **metadata detection** refers to the process in which sample user data is read from the target system and the corresponding metadata (identity field names) is displayed on the Step 3: Modify Connector Configuration page.

Oracle Identity Manager performs the following steps while attempting to detect metadata:

1. The reconciliation transport provider and reconciliation format provider try to fetch and parse metadata from the target system.

Together, the shared drive reconciliation transport provider and CSV reconciliation format provider can detect metadata from the target system. If you want custom providers to perform the same function, you must ensure that:

- The Java code for the reconciliation transport provider contains an implementation of the `getMetadata()` method of the `ReconTransportProvider` interface.
- The Java code for the reconciliation format provider contains an implementation of the `parseMetadata()` method of the `ReconFormatProvider` interface.

See Also: [Chapter 20, "Creating Custom Providers for Generic Technology Connectors"](#)

If these providers successfully fetch and parse metadata from the target system, Oracle Identity Manager uses information returned by them to display metadata and the following step is not performed.

2. If the reconciliation transport provider and reconciliation format provider cannot fetch and parse metadata from the target system, the provisioning transport provider and provisioning format provider try to perform this function.

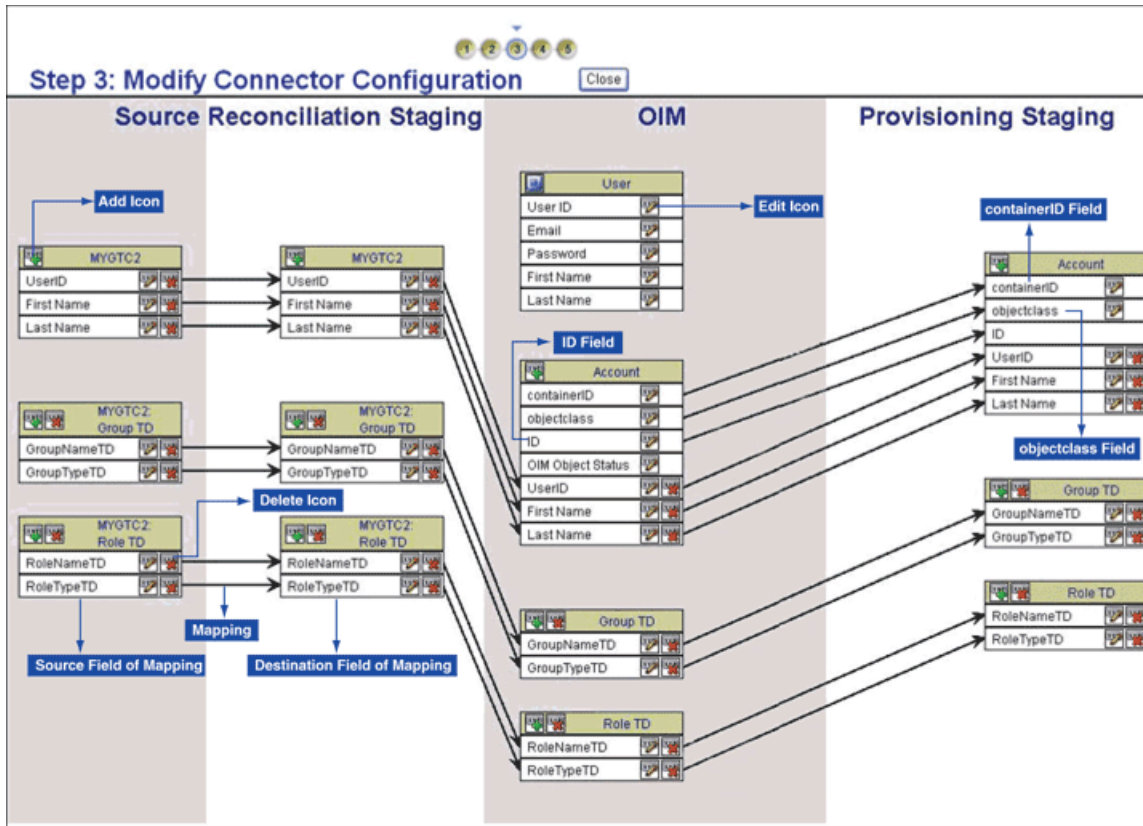
The Web Services provisioning transport provider and SPML provisioning format provider cannot detect metadata from the target system. If you want custom providers to be able to detect metadata, you must ensure that:

- The Java code for the provisioning transport provider contains an implementation of the `defineMetadata()` method of the `ProvisioningTransportProvider` interface.
- The Java code for the provisioning format provider contains an implementation of the `parseMetadata()` method of the `ProvisioningFormatProvider` interface.

If the provisioning transport provider and provisioning format provider successfully fetch and parse metadata from the target system, Oracle Identity Manager uses information returned by these providers to display metadata. If these providers are not successful, only the default fields defined for any of the provisioning-specific providers that you select are displayed. For example, the `ID` field of the OIM - Account data set and the `objectClass` and `containerID` fields of the provisioning staging data set are displayed by default. These data sets and fields are discussed later in this guide.

[Figure 21–5](#) shows the Step 3: Modify Connector Configuration page for the sample entries listed at the end of the "Step 1: Provide Basic Information Page" and "Step 2: Specify Parameter Values Page" sections.

Figure 21–5 Step 3: Modify Connector Configuration Page



- [Data Sets](#)
- [Mappings](#)

Data Sets

The data sets displayed on the Step 3: Modify Connector Configuration page are categorized as follows:

- **Source**
The Source data sets are displayed only if you select the Reconciliation option on the first page, regardless of whether or not you select the Provisioning option.
- **Reconciliation Staging**
The reconciliation staging data sets are displayed only if you select the Reconciliation option on the Step 1: Provide Basic Information page, regardless of whether or not you select the Provisioning option.
- **Oracle Identity Manager**
The Oracle Identity Manager data sets are always displayed, regardless of the options you select on the Step 1: Provide Basic Information page. However, the OIM - Account data set and its child data sets are not displayed if you select the Trusted Source Reconciliation option on the Step 1: Provide Basic Information page.

The fields displayed in the OIM - User data set are predefined for the Oracle Identity Manager User. You can show or minimize the full list of OIM - User data

set fields by clicking the arrow icon at the top of the data set. The following fields are displayed in the minimized state of the data set:

- User ID
- Email
- Password
- First Name
- Last Name

Note: If you select the Trusted Source Reconciliation option on the Step 1: Provide Basic Information Page, all the fields of the OIM - User data set are displayed and you cannot use the arrow icon to minimize the display.

These fields constitute the minimum set of Oracle Identity Manager User fields for which values must be defined. You can designate some or all of the remaining OIM - User data set fields as mandatory Oracle Identity Manager User fields for your Oracle Identity Manager installation. You do this by ensuring that these fields always hold values when the Oracle Identity Manager User is created.

Note: Data set and field names that take up more than a certain amount of space are truncated and dots are displayed after the truncated part of the names. For example, the Deprovisioning Date field of the OIM - User data set is displayed as follows:

Deprovisioning Da..

To view the full name of a field, you can click the edit icon for that field or the field to which that field is mapped. In the pop-up window, the field name that you want to view is on either the first page or the second page, depending on the data set to which the field belongs.

You can add user-defined fields (UDFs) to the list of predefined Oracle Identity Manager User fields by using the Design Console. These UDFs are displayed in the OIM - User data set on the Step 3: Modify Connector Configuration page.

Depending on the options that you select on the Step 1: Provide Basic Information page, some fields are displayed by default on the Step 3: Modify Connector Configuration page:

- ID field

The ID field is displayed by default in the OIM - Account data set, regardless of whether or not you select the Reconciliation option or Provisioning option on the Step 1: Provide Basic Information page. When an account is created, this field is used to store the value that uniquely identifies the account in Oracle Identity Manager and in the target system. For a particular user, this unique field is used to direct other operations, such as modify, delete, enable, disable, and child data operations.

Every target system would have a unique field for tracking the creation of and updates made to a user account. While creating a custom provisioning transport provider, you must ensure that the provider retrieves this unique

field value from the target system at the end of a Create User operation. This value must be used to populate the ID field of the OIM - Account data set.

During reconciliation, the value of the ID field must come from the corresponding unique field of the reconciliation staging data set. To set this up, you must create a mapping between the two fields. The procedure to create a mapping is discussed later in this section.

Caution: If you select both the Provisioning and Reconciliation options while creating a generic technology connector and if you do not create a mapping between the ID field and the unique field of the target system, records that are linked through reconciliation cannot be used for provisioning operations (such as modify, delete, enable, disable, and child data operations). This is because the ID field is not populated in the linked records.

- objectClass field

The objectClass field is displayed by default in the OIM - Account data set and provisioning staging data set only if you select the SPML provisioning format provider on the Step 1: Provide Basic Information page.

- containerID field

The containerID field is displayed by default in the OIM - Account data set and provisioning staging data set only if you select the SPML provisioning format provider on the Step 1: Provide Basic Information page.

- Provisioning Staging

The provisioning staging data sets are displayed only if you select the Provisioning option on the first page, regardless of whether or not you select the Reconciliation option.

The display of data sets on the Step 3: Modify Connector Configuration page depends on the input that you provide on the Step 1: Provide Basic Information page and Step 2: Specify Parameter Values page. The display of fields within the data sets depends on whether or not metadata detection has taken place.

Note: Metadata detection does not take place if any of the following conditions are true:

- Sample target system data (including metadata) is not available.
 - The Transport and format providers that you select are not capable of detecting metadata from sample target system data.
-

This is illustrated by the following example:

Suppose you select only the Reconciliation option on the Step 1: Provide Basic Information page. In addition, metadata detection has not taken place. Under these conditions, the display of data sets and fields on the Step 3: Modify Connector Configuration page can be summarized as follows:

The following data sets are displayed:

- Source
- Reconciliation Staging

- Oracle Identity Manager

The fields that constitute the data sets are *not* displayed.

In addition, if you had selected the Trusted Source Reconciliation option on the Step 1: Provide Basic Information page, the OIM - Account data set and its child data sets are not displayed.

In [Table 21-3](#), Scenario 1 shows the outcome of this set of input conditions. The rest of the scenarios in this table describe the display of data sets and fields under the combination of input conditions listed in the first row and first column of the table.

Table 21-3 Display of Data Sets and Fields Under Various Input Conditions

	Only Reconciliation Option Selected	Both Reconciliation and Provisioning Options Selected	Only Provisioning Option Selected
Metadata detection has <i>not</i> taken place	<p>Scenario 1</p> <p>The following data sets are displayed:</p> <ul style="list-style-type: none"> ■ Source ■ Reconciliation Staging ■ Oracle Identity Manager <p>The fields that constitute the data sets are <i>not</i> displayed.</p> <p>If you select the Trusted Source Reconciliation option on the Step 1: Provide Basic Information page, the OIM - Account data set and its child data sets are not displayed.</p>	<p>Scenario 2</p> <p>The following data sets are displayed:</p> <ul style="list-style-type: none"> ■ Source ■ Reconciliation Staging ■ Oracle Identity Manager ■ Provisioning Staging <p>The fields that constitute the data sets are <i>not</i> displayed.</p>	<p>Scenario 3</p> <p>The following data sets are displayed:</p> <ul style="list-style-type: none"> ■ Oracle Identity Manager ■ Provisioning Staging <p>The fields that constitute the data sets are <i>not</i> displayed.</p>
Metadata detection has taken place	<p>Scenario 4</p> <p>The following data sets are displayed:</p> <ul style="list-style-type: none"> ■ Source ■ Reconciliation Staging ■ Oracle Identity Manager <p>The fields that constitute the data sets are displayed.</p> <p>If you select the Trusted Source Reconciliation option on the Step 1: Provide Basic Information page, the OIM - Account data set and its child data sets are not displayed.</p>	<p>Scenario 5</p> <p>The following data sets are displayed:</p> <ul style="list-style-type: none"> ■ Source ■ Reconciliation Staging ■ Oracle Identity Manager ■ Provisioning Staging <p>The fields that constitute the data sets are displayed.</p>	<p>Scenario 6</p> <p>The following data sets are displayed:</p> <ul style="list-style-type: none"> ■ Oracle Identity Manager ■ Provisioning Staging <p>The fields that constitute the data sets are displayed.</p>

See Also: [Section 22.1.2, "Multi-language Support"](#) for information about limitations related to the display of non-ASCII characters on this page

Mappings

Each flow line displayed on the Step 3: Modify Connector Configuration page represents a mapping (link) between two fields of different data sets. A mapping serves one of the following purposes:

- Establishes a data flow path between fields of two data sets, for either provisioning or reconciliation
A mapping of this type forms the basis for validations or transformations to be performed on data.
- Creates a basis for comparing (matching) field values of two data sets
The following are examples of matching-only mappings:
 - Mappings created between fields of the reconciliation staging data set and the OIM - User data set form the basis of a reconciliation rule.
 - A mapping between the unique field of the reconciliation staging data set and the ID field of the OIM - Account data set helps identify the key field for reconciliation matching. Along with the ID field, other fields of the OIM - Account data set can be (matching-only) mapped to corresponding fields of the reconciliation staging data set to create a composite key field for reconciliation matching.

You can perform the following actions on the Step 3: Modify Connector Configuration page:

- [Adding or Editing Fields in Data Sets](#)
- [Removing Fields from Data Sets](#)
- [Removing Mappings Between Fields](#)
- [Removing Child Data Sets](#)

21.2.4.3.1 Adding or Editing Fields in Data Sets Identity fields detected through metadata detection are displayed on the Step 3: Modify Connector Configuration page. You can modify these fields and the mappings between them. If required, you can also add new fields on this page and create mappings between them.

The following is a summary of the actions that you can perform while adding or editing fields on the Step 3: Modify Connector Configuration page:

Note: These actions are described in detail in the procedure that follows this list. The procedure also describes the conditions that must be fulfilled before you can perform some of these actions.

- Default attributes (such as the data type and length) are assigned to the fields displayed through metadata detection. You must edit these fields to set the required attributes for them.

Note: Oracle Identity Manager can recognize date values fetched during reconciliation only if you set the Date data type for fields of the reconciliation staging data sets. In addition, if you have specified a value for the Source Date Format parameter on the Step 2: Specify Parameter Values page, you must map date fields of the Source data sets to the corresponding date fields of the reconciliation staging data sets.

The Source Date Format parameter is described in [Section 21.2.4.2, "Step 2: Specify Parameter Values Page"](#).

- You can create transformation mappings between fields by using a transformation provider. While performing this action, you can use the predefined concatenation transformation provider or translation transformation provider, or a custom transformation provider that you have created.
- You can create matching-only mappings between fields of the reconciliation staging data set and Oracle Identity Manager data sets. Matching-only mappings that you create between the reconciliation staging data set and the OIM - User data set forms the reconciliation rule. Matching-only mappings that you create between the reconciliation staging data set and the OIM - Account data set identifies the key field for reconciliation matching.
- You can add a child data set to an existing data set.
- You can encrypt the value of a field, both in the process form and in the database.
- You can designate a field as a lookup field and select an input source for the field. The input source can be a lookup definition or a combination of columns from Oracle Identity Manager database tables.
- You can configure user account status reconciliation.

If you want to configure user account status reconciliation, refer to the "Configuring Account Status Reconciliation" section.

To add or edit a field in a data set:

Note: The display of the GUI elements and pages described in the following steps depends on the data set in which you are adding or editing a field. For example, the Required and Encrypted check boxes are not displayed if you are adding or editing a field in a Source data set.

1. Depending on whether you want to add or edit a field, click the Add icon for the data set or the edit icon for the field.
2. On the Step 1: Field Information page, specify values for the following GUI elements:

See Also: [Section 21.2.4.3, "Step 3: Modify Connector Configuration Page"](#) for information about validations applied to the names of fields

- **Field Name:** If you are adding a field, specify a name for the field. The field name that you specify must contain only ASCII characters, because non-ASCII characters are not allowed.
- **Mapping Action:** Select the type of mapping that you want to create with this field as the destination field of the mapping. You can select one of the following mapping actions:
 - Select **Create Mapping Without Transformation** if you only want to create a one-to-one mapping between a source (input) field and the field that you are adding or editing, and you do not want to use a transformation provider.
 - Select the **Remove Mapping** option if you are editing the field and you want to remove the mapping for which this field is the destination field. The procedure to remove a mapping is covered in detail in the Removing Mapping Between Fields section.

- The transformation mapping options displayed in the Mapping Action list are based on the predefined transformation providers and the custom transformation providers that you create. The following menu options correspond to the predefined transformation providers:

- * **Create Mapping With Concatenation**

- * **Create Mapping With Translation**

See Also: [Section 19.5, "Transformation Providers"](#) for information about these predefined transformation providers

Apply the following guidelines while selecting a transformation mapping:

- * You can create transformation mappings only between fields of the following data sets:

- Source and Reconciliation Staging

- Oracle Identity Manager and Provisioning Staging

This means that, for example, you cannot create transformation mappings between a field in a reconciliation staging data set and a field in an Oracle Identity Manager data set.

You cannot create a 1-to-2 mapping with the following source and destination fields:

Source field: Unique field of the reconciliation staging data

Destination fields: `User ID` field of the OIM - User data set and `ID` field of the OIM - Account data set

This mapping is not supported. Instead, you must create a one-to-one mapping between the unique field of the reconciliation staging data and either the `User ID` field (of the OIM - User data set) or the `ID` field (of the OIM - Account data set).

- * Ensure that all the fields of provisioning staging data sets are mapped to corresponding fields of OIM - User and OIM - Account data sets.
- * When you create a mapping that has any field of the OIM - User data set as the source or destination field, the display of the OIM - User data set fields list is frozen in the position it was in (expanded or minimized) when the mapping was created. To unfreeze the display of the OIM - User data set so that you are able to use the arrow icon, you must remove all mappings that have any OIM - User data set field as the source or destination field.
- * A literal field can be used as one of the input fields of a transformation field. If you select the Literal option, you must enter a value in the field. You must not leave the field blank after selecting it.

See [Section 21.2.4.3, "Step 3: Modify Connector Configuration Page"](#) for information about limitations related to creating transformation mappings.

- **Matching Only:** Select this check box if the field is to be used as the destination field of a matching-only mapping. As mentioned earlier in this document, you can create the following types of matching-only mappings:

Note: You must create matching-only mappings for both parent and child data sets.

- To create the reconciliation rule, you create matching-only mappings between fields of the reconciliation staging data set and the OIM - User data set. Each mapping represents a reconciliation rule element. If there are child data sets, you must ensure that the names of fields of the reconciliation staging data set that are input fields for the matching-only mappings are not used in any of the reconciliation staging child data sets.
- To specify the key field for reconciliation matching, you create a matching-only mapping between the unique field of the reconciliation staging data set and the ID field of the OIM - Account data set. Along with the ID field, other fields of the OIM - Account data set can be (matching-only) mapped to corresponding fields of the reconciliation staging data set to create a composite key field for reconciliation matching.

Caution: If the name of a reconciliation staging field used in a matching-only mapping were to be reused as the name of a field in a reconciliation staging child data set, matching would not take place during a reconciliation run.

This known issue is explained in the Modify Connector Config Page section .

- **Create End-to-End Mapping:** If you are adding a field, select this check box if you want the same field to be added in all the data sets that are displayed to the right of the data set in which you are adding the field.
- **Multi-Valued Field:** Select this check box if you want to add a child data set. If you select this check box, the name that you specify in the Field Name field is used as the name of the child data set.

Note: If you select the Trusted Source Reconciliation check box on the Step 1: Provide Basic Information page, this check box (in selected or deselected state) is ignored. This is because the reconciliation of multivalued (child) data is not supported in trusted source reconciliation.

- **Data Type:** Select the data type of the field.
After metadata detection, the String data type is applied by default to all the fields of the reconciliation staging and OIM - Account data sets. Where required, you must use the Data Type list to specify the actual data type of each field.
- **Length:** Specify the character length of the field.
- **Required:** Select this check box if you want to ensure that the field always contains a value.
- **Encrypted:** Select this check box if the value of the field must be stored in encrypted form in the Oracle Identity Manager database.
- **Password Field:** Select this check box if the value of the field must be encrypted on the process form. Values of fields for which this check box is selected are displayed as asterisks (*) on the process forms.

Note: If you select the Encrypted and Password Field check boxes, see [Section 21.5.3.3, "Password-Like Fields"](#) for information about guidelines that you must follow.

- **Lookup Field:** Select this check box if you want to make the field a lookup field.
- 3. Click **Continue**.
- 4. If you select the **Lookup Field** check box on the Step 1: Field Information page, the Step 2: Lookup Properties page is displayed. On this page, you can select and specify values for any combination of the lookup properties described in [Table 21-4](#).

Table 21-4 Lookup Properties

Lookup Property	Value
Column Names	<p>In the Property Value field, enter the name of the database column containing the values that must be displayed in the lookup window. If required, you can enter multiple database column names separated by commas.</p> <p>Note: If you select the Lookup Column Name property, you must also select the Column Names property, which is described later in this table.</p> <p>After you enter a value in the Property Value field, click Submit.</p> <p>The following SQL query can be used to illustrate how the Column Names and Lookup Column Name properties are used:</p> <pre>SELECT USR_FIRST_NAME, USR_LOGIN, USR_LAST_NAME FROM USR</pre> <p>Suppose you set the following as the values of the two properties:</p> <ul style="list-style-type: none"> - Column Names: USR_FIRST_NAME, USR_LAST_NAME - Lookup Column Name: USR_LOGIN <p>When the user selects a particular USR_FIRST_NAME, USR_LAST_NAME combination from the lookup window, the corresponding USR_LOGIN value is stored in the database.</p>
Column Captions	<p>In the Property Value field, enter the name of the column heading that must be displayed in the lookup window. If multiple columns are going to be displayed in the lookup window, enter multiple column captions separated by commas, for example, Organization Name, Organization Status.</p> <p>After you enter a value in the Property Value field, click Submit.</p>
Column Widths	<p>In the Property Value field, enter the character width of the column that must be displayed in the lookup window. This must be the same as the maximum length of the underlying field or column from which data values are drawn to populate the lookup field.</p> <p>If the lookup window is going to display multiple columns, enter multiple column widths separated by commas.</p> <p>After you enter a value in the Property Value field, click Submit.</p>

Table 21–4 (Cont.) Lookup Properties

Lookup Property	Value
Lookup Query	<p>To specify a value for the Lookup Query property:</p> <ol style="list-style-type: none"> 1. In the Property Value field, enter the SQL query (without the <code>WHERE</code> clause) that must be run when a user double-clicks the lookup field to populate the data columns displayed in the lookup window. 2. Click Submit. 3. On the Step 2: Add Validation page, select values from the following lists to create a <code>WHERE</code> clause for the <code>SELECT</code> statement that you specify in Step 1: <ul style="list-style-type: none"> - Filter Column - Source - Field Name <p>From the values that you select, the <code>WHERE</code> clause is created as follows:</p> <pre>WHERE Filter_Column=Source.Field_Name</pre> 4. Click Save. <p>To correctly display the data returned from a query, you must add a <code>lookupfield.header</code> property to the <code>xlWebAdmin_locale.properties</code> file.</p> <p>For example, consider the following SQL query:</p> <pre>SELECT usr_status FROM usr</pre> <p>To view the data returned from the query, you must add the following entry to the <code>xlWebAdmin_locale.properties</code> files:</p> <pre>lookupfield.header.users.status=User Status</pre> <p>If the <code>xlWebAdmin_locale.properties</code> file does not contain a <code>lookupfield.header</code> property for your specified query, the Administrative and User Console displays a lookup window after you click the corresponding lookup icon.</p> <p>The syntax for a <code>lookupfield.header</code> property is as follows:</p> <pre>lookupfield.header.column_code=display value</pre> <p>The <code>column_code</code> portion of the entry must be lowercase and any spaces must be replaced by underscore characters (<code>_</code>).</p> <p>By default, the following entries for lookup field column headers are already available in the <code>xlWebAdmin_locale.properties</code> file:</p> <pre>lookupfield.header.lookup_definition.lookup_code_information .code_key=Value lookupfield.header.lookup_definition.lookup_code_information .decode=Description lookupfield.header.users.manager_login=User ID lookupfield.header.organizations.organization_name=Name lookupfield.header.it_resources.key=Key lookupfield.header.it_resources.name=Instance Name lookupfield.header.users.user_id=User ID lookupfield.header.users.last_name=Last Name lookupfield.header.users.first_name=First Name lookupfield.header.groups.group_name=Group Name lookupfield.header.objects.name=Resource Name lookupfield.header.access_policies.name=Access Policy Name</pre>

Table 21–4 (Cont.) Lookup Properties

Lookup Property	Value
Lookup Code	<p>In the Property Value field, enter the lookup definition code name. This code must generate all information pertaining to the lookup field, including lookup values and the text that is displayed with the lookup field when a lookup value is selected. The classification type of the lookup definition code must be of Lookup Type (that is, the Lookup Type option on the Lookup Definition form must be selected).</p> <p>To enter a lookup code, open the Lookup Definition form, query for the required code, and copy the code into the Property Value field.</p> <p>After you enter a value in the Property Value field, click Submit.</p> <p>Note:</p> <p>The Lookup Code property can be used to replace the combination of the Column Captions, Column Names, Column Widths, Lookup Column Name, and Lookup Query properties. In addition, the information contained in the Lookup Code property supersedes any values set in these five lookup properties.</p> <p>If you want to implement lookup fields reconciliation, create a scheduled task that populates the lookup code.</p>
Lookup Column Name	<p>In the Property Value field, enter the name of the database column containing the value that must be stored corresponding to the Column Names value selected by the user in the lookup window. If required, you can enter multiple database column names separated by commas.</p> <p>Note: If you select the Column Names property, you must also select the Lookup Column Name property. See the "Lookup Column Name" row in this table for more information about how these two properties are used.</p> <p>After you enter a value in the Property Value field, click Submit.</p>
Auto Complete	<p>If you enter <code>True</code> in the Property Value field, users can filter the values displayed in the lookup window by entering the first few characters of the value they want to select and double-clicking the lookup field. The outcome of this action is that only lookup values that begin with the characters entered by the users are displayed in the lookup window. For example, for the State lookup field, a user can enter <code>New</code> in the field. When the user double-clicks the State lookup field, only states that begin with <code>New</code> (for example, New Hampshire, New Jersey, New Mexico, and New York) are displayed in the lookup window.</p> <p>If you do not want to let users filter the display of values in the lookup field, enter <code>False</code> in the Property Value field.</p> <p>The default value of the Auto Complete property is <code>False</code>.</p> <p>After you enter a value in the Property Value field, click Submit.</p>

If you want to edit the value of a property that is displayed in the table on the Step 2: Lookup Properties page, select the edit option for that property and click **Edit**. If you want to remove a property that is displayed in the table, select the delete option for that property and click **Delete**.

After you specify properties for the lookup field, click **Continue**.

5. If you select a transformation option from the Mapping Action list on the Step 1: Field Information page, the Step 3: Mapping page is displayed. Use this page to define the transformation function that you want to perform on the input data to the field that you are adding. The steps to be performed depend on the transformation provider option (concatenation, translation, or custom transformation provider) that you select on the previous page:

If you select a predefined transformation provider (transformation, concatenation or translation), see Transformation Providers for detailed information about the procedure to specify parameter values for the predefined transformation provider.

That section also provides detailed information about configuring user account status reconciliation.

You must use the translation transformation provider if you want to configure the reconciliation of user account status information. This procedure is described in [Section 19.5.2, "Translation Transformation Provider"](#).

After you specify values for the transformation provider, click **Continue**.

6. If required, select a validation check for the field and click **Add**. In other words, select the validation provider that you want to use.

The validation options displayed in this list are based on the predefined validation Providers and any custom validation Providers that you create.

7. Click **Continue**, and click **Close**.
8. If you do not want to perform any other action on the Step 3: Modify Connector Configuration page, click the **Close** button that is displayed at the top of the page. You must perform the previous step before you click this Close button.

21.2.4.3.2 Removing Fields from Data Sets To remove a field from a data set:

1. Click the Delete icon for that field.
2. If you do not want to perform any other action on the Step 3: Modify Connector Configuration page, click the **Close** button that is displayed at the top of the page.

21.2.4.3.3 Removing Mappings Between Fields To remove a mapping:

1. Click the edit icon for the destination field of the mapping that you want to remove.

Note: If the destination field itself is the source field for another mapping, that mapping is not removed.

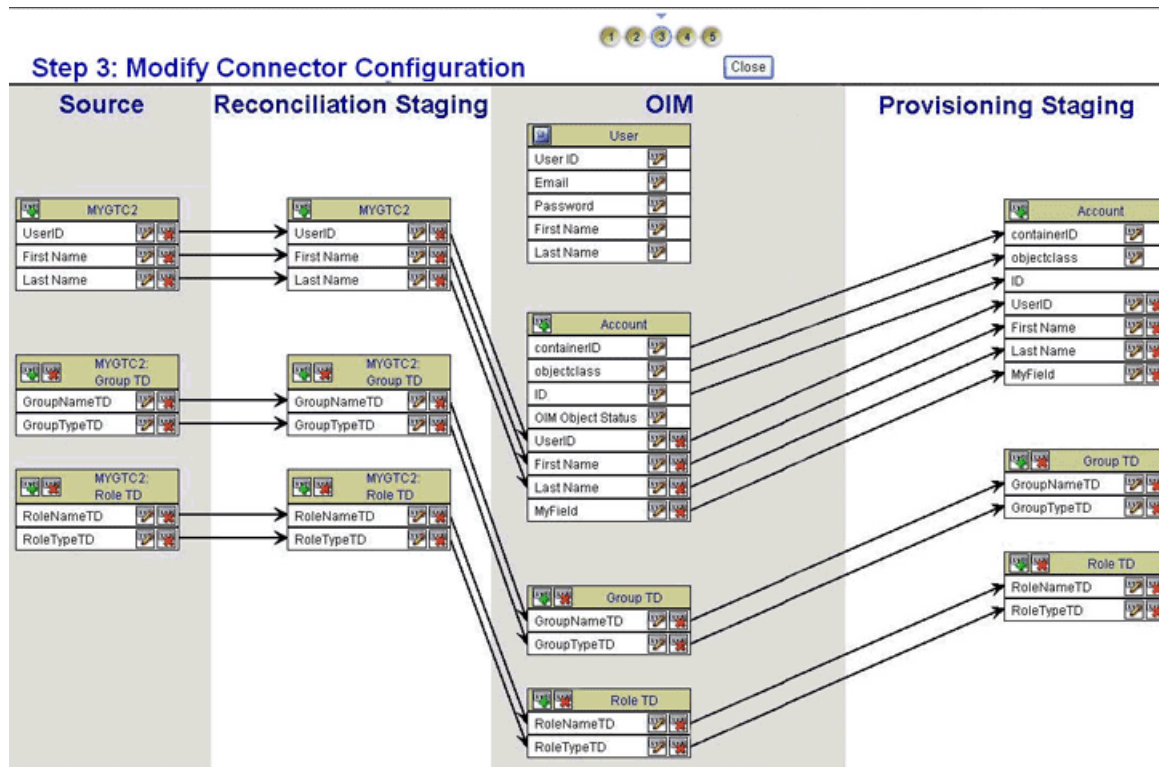
2. On the Step 1: Field Information page, select **Remove Mapping** from the **Transformation** list.
3. Click **Continue**.
4. On the last page, click **Close**.
5. If you do not want to perform any other action on the Step 3: Modify Connector Configuration page, click the **Close** button that is displayed at the top of the page.

21.2.4.3.4 Removing Child Data Sets To remove a child data set:

1. Click the Delete icon for the data set.
2. If you do not want to perform any other action on the Step 3: Modify Connector Configuration page, click the **Close** button that is displayed at the top of the page.

[Figure 21–6](#) shows the Step 3: Specify Connector Configuration page after the MyField field was added to the OIM - Account and provisioning staging data sets.

Figure 21–6 Step 3: Modify Connector Configuration Page After Addition of a Field



21.2.4.4 Step 4: Verify Connector Form Names Page

Use this page to specify form names for the process forms corresponding to the OIM - Account data set and its child data sets.

Note: If you select the Trusted Source Reconciliation option on the Step 1: Provide Basic Information page, the OIM - Account data set and its child data sets are not created. Therefore, this page is not displayed if you select the Trusted Source Reconciliation option.

The generic technology connector framework automatically creates certain objects after you submit all the information required to create a generic technology connector. Parent and child process forms corresponding to the OIM - Account data sets are examples of objects that are automatically created. Each process form on a particular Oracle Identity Manager installation must have a unique name.

See Also: [Section 18.4, "Connector Objects Created by the Generic Technology Connector Framework"](#)

On the Step 4: Verify Connector Form Names page, the generic technology connector framework displays default names for these process forms based on the names of the corresponding data sets. You must verify and, if required, change the names of these forms to ensure that they are unique for this installation of Oracle Identity Manager. While changing the name of a form, you must use only ASCII characters. An error message is displayed if you specify non-unique form names or if any name contains non-ASCII characters.

Note: You cannot revisit this page, so ensure that the form names that you specify meet all the requirements before you click **Continue**.

After you specify the form names, click **Continue**.

Instead of clicking Continue, you can click **Back** to return to the Step 2: Specify Parameter Values page. However, metadata detection does not take place if you make changes on this page and click the Continue button. This is to ensure that any customization in the data set structure and mappings made during the first pass through this page does not get overwritten. You can manually add or edit fields and mappings on the Step 3: Modify Connector Configuration page.

Figure 21–7 shows that Step 4: Verify Connector Form Names page on which the entries listed at the end of Section 21.2.4.1, "Step 1: Provide Basic Information Page" and Section 21.2.4.2, "Step 2: Specify Parameter Values Page" and the changes described at the end of Section 21.2.4.3, "Step 3: Modify Connector Configuration Page" have been made.

Figure 21–7 Step 4: Verify Connector Form Names Page



21.2.4.5 Step 5: Verify Connector Information Page

Use this page to review information that you have provided up to this point for creating generic technology connectors. The following is a page-wise explanation of the changes that are permitted on the earlier pages:

- Step 1: Provide Basic Information page

You can use either the View link or Back button to reopen and view the information provided on the Step 1: Provide Basic Information page. You cannot

change the information displayed on this page, because any change in this information would amount to creating a new generic technology connector.

- **Step 2: Specify Parameter Values page**

You can use either the Change link or Back button to reopen this page. You can change parameter values on this page. However, metadata detection does not take place when you submit the changed values. This is to ensure that any customization in the data set structure and mappings made during the first pass through this page does not get overwritten. You can manually add or edit fields and mappings on the Step 3: Modify Connector Configuration page.

- **Step 3: Modify Connector Configuration page**

You can use the Change link to reopen this page and add or edit fields and mappings.

- **Step 4: Verify Connector Form Names page**

You cannot revisit this page.

After you verify all the information displayed on the Step 5: Verify Connector Information page, click **Create**.

At this stage, the generic technology connector framework creates all the standard connector objects on the basis of the information that you provide. The list of these objects includes the connector XML file, which is created and imported automatically into Oracle Identity Manager. Except for the form names, the names of the connector objects are in the *GTCname_GTC* format.

For example, if you specify `DB_conn` as the name of a generic technology connector that you create, all (except the forms) the connector objects are named `DB_CONN_GTC`.

See Also: [Section 18.4, "Connector Objects Created by the Generic Technology Connector Framework"](#)

At the end of the process, a message stating that the connector has been successfully created is displayed on the page.

Note: If the creation process fails, objects that are created are not automatically deleted. This point is also mentioned in [Section 22.1.1, "Creation Issues"](#).

See [Section 22.2.3, "Errors During Connector Creation"](#) for a listing of error messages related to the creation process.

[Figure 21–8](#) shows the first section of the Step 5: Verify Connector Information page on which the entries listed at the end of [Section 21.2.4.1, "Step 1: Provide Basic Information Page"](#) and [Section 21.2.4.2, "Step 2: Specify Parameter Values Page"](#) and the changes described at the end of [Section 21.2.4.3, "Step 3: Modify Connector Configuration Page"](#) have been made.

Figure 21–8 First Section of the Step 5: Verify Connector Information Page

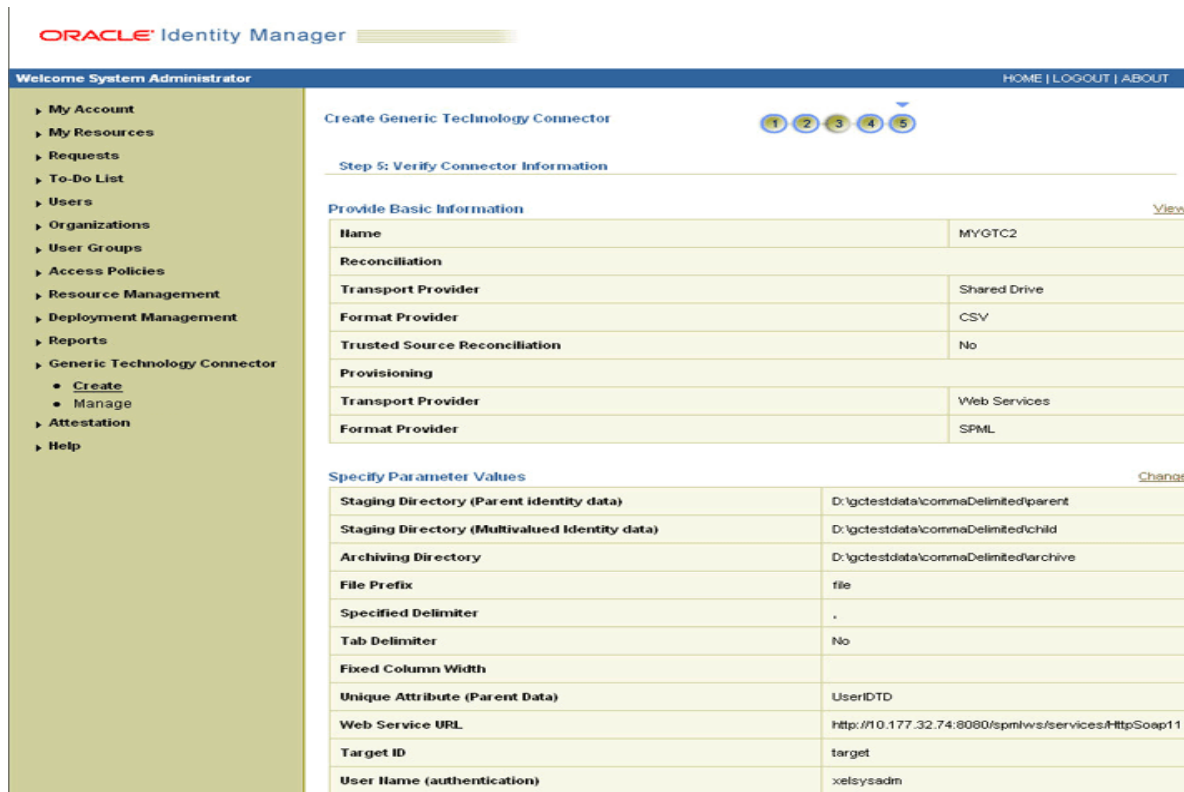


Figure 21–9 shows the second section of the Step 5: Verify Connector Information page on which the entries listed at the end of Section 21.2.4.1, "Step 1: Provide Basic Information Page" and Section 21.2.4.2, "Step 2: Specify Parameter Values Page" and the changes described at the end of Section 21.2.4.3, "Step 3: Modify Connector Configuration Page" have been made.

Figure 21–9 Second Section of the Step 5: Verify Connector Information Page

User Password (authentication)	*****
File Encoding	Cp1251
Web Service SOAP Action	http://xmlns.oracle.com/OIM/provisioning/processRequest
WSSE Configured for SPML Web Service?	No
Custom Authentication Credentials Namespace	http://xmlns.oracle.com/OIM/provisioning
Custom Authentication Header Element	OIMUser
Custom Element to Store User Name	OIMUserId
Custom Element to Store Password	OIMUserPassword
SPML Web Service Binding Style (Document or RPC)	RPC
SPML Web Service Complex Data Type	
SPML Web Service Operation Name	processRequest
SPML Web Service Target Namespace	http://xmlns.oracle.com/OIM/provisioning
SPML Web Service Soap Message Body Prefix	
ID Attribute for Child Dataset Holding Group Membership Information	
Target Date Format	yyyy-MM-dd hh:mm:ss.ffffff
Batch Size	All
Stop Reconciliation Threshold	None
Stop Threshold Minimum Records	None
Source Date Format	yyyy/MM/dd hh:mm:ss z
Reconcile Deletion of Multivalued Attribute Data	Yes
Reconciliation Type	Incremental

Connector Configuration [Change](#)

Exit << Back Save

Oracle Identity Manager 9.1.0 Copyright © 2007, Oracle Corporation.

21.2.5 Configuring Reconciliation

Note: If you select only the Provisioning option on the Step 1: Provide Basic Information page, you can skip this section because you need not configure reconciliation.

A reconciliation scheduled task is created automatically when you create the generic technology connector. To configure and run this scheduled task, follow the instructions in the "Creating and Managing Scheduled Tasks" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager*.

Note: The name of the scheduled task is in the following format:

GTC_Name_GTC

For example, if the name of the generic technology connector is WebConn, the name of the scheduled task is WebConn_GTC.

21.2.6 Configuring Provisioning

Note: If you select only the Reconciliation option on the Step 1: Provide Basic Information page, you can skip this section because you need not configure provisioning.

A process definition is one of the objects that are automatically created when you create a generic technology connector. The name of the process definition is in the following format:

GTC_name_GTC

For example, if the name of the generic technology connector is WebConn, the name of the process definition is WebConn_GTC.

The process tasks that constitute this process definition can be divided into two types:

- System-defined process tasks
System-defined process tasks are included by default in all newly created process definitions.
- Provisioning-specific process tasks
Provisioning-specific process tasks are included in the process definition of a generic technology connector only if you select the Provisioning option on the Step 1: Provide Basic Information page, regardless of whether or not you select the Reconciliation option.

The following are provisioning-specific process tasks:

- Create User
- Delete User
- Enable User
- Disable User
- Updated *Field_Name* (this task is created for each field of the OIM - Account data set, except the ID field)
- For mappings created between fields of the OIM - User data set and the provisioning staging data set, the following process tasks are created:
 - Change *User_data_set_field_name*
 - Edit *Provisioning_Staging_field_name*

For example, suppose you create a mapping between the Last Name field of the OIM - User data set and the LName field of the provisioning staging data set. The following process tasks are automatically created along with the rest of the provisioning-specific process tasks:

- Change Last Name
- Edit LName

In addition, the following provisioning-specific process tasks are created for each child data set of the OIM - Account data set:

- Child Table *Child_Form_Name* row Inserted
- Child Table *Child_Form_Name* row Updated
- Child Table *Child_Form_Name* row Deleted

All provisioning-specific process tasks have the following default assignments:

- Target Type: Group User With Highest Priority
- Group: SYSTEM ADMINISTRATORS
- User: XELSYSADM

If required, you can modify these default assignments by following the instructions given in ["Modifying Process Tasks"](#) on page 12-15.

21.2.7 Enabling Logging

Note: This is an optional step. Perform the procedure discussed in this section only if you want to enable logging for the generic technology connector.

See "Enabling System Logging" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager* for information about enabling logging in Oracle Identity Manager.

21.3 Managing Generic Technology Connectors

The generic technology connector framework offers features that enable you to modify a generic technology connector. In addition, you can export or import a generic technology connector by using the Deployment Manager.

This section contains these topics:

- [Modifying Generic Technology Connectors](#)
- [Exporting Generic Technology Connectors](#)
- [Importing Generic Technology Connectors](#)

21.3.1 Modifying Generic Technology Connectors

Caution: The Design Console can be used to modify connector objects that are automatically created at the end of the generic technology connector creation process. If you use the Manage Generic Technology Connector feature to modify a generic technology connector whose connector objects have been customized by using the Design Console, all the customization work done using the Design Console would get overwritten. Therefore, Oracle recommends that you to follow one of the following guidelines:

- Do not use the Design Console to modify generic technology connector objects.

The exception to this guideline is the IT resource. You can modify the parameters of the IT resource by using the Design Console. However, for the changes to take effect, you must purge the cache either before or after you modify IT resource parameters. See "Purging the Cache" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager* for information about running the `PurgeCache` utility.

- If you use the Design Console to modify generic technology connector objects, do not use the Manage Generic Technology Connector feature to modify the generic technology connector.

See [Section 18.4, "Connector Objects Created by the Generic Technology Connector Framework"](#) for information about connector objects that are created automatically by the framework.

In addition, you can modify only one connector at a time. If you try to use the Modify pages for two different connectors at the same time on the same computer, the Modify features would not work correctly.

[Appendix 22, "Troubleshooting Generic Technology Connectors"](#) discusses both these points.

To modify a generic technology connector:

1. Login to the Administrative and User Console.
2. Click **Advanced**.
3. Under Configuration, click **Manage Generic Connector**.
4. Search for the connector that you want to modify. To simplify your search, you can use a combination of the search criteria provided on this page. Alternatively, to view all the generic technology connectors that have been created on this Oracle Identity Manager installation, click **Search connectors** without specifying any search criteria.
5. In the results that are displayed, click the generic technology connector that you want to modify.
6. Click **Edit Parameters**. The Step 2: Specify Parameter Values page of the connector creation process is displayed. From this point onward, follow the procedure described in the Step 2 section .

Note: The only difference between this procedure and the procedure that you follow to create the generic technology connector procedure is that automatic metadata detection does not take place when you modify an existing generic technology connector.

Caution: If you modify attributes of fields of the OIM - Account data set or its child data sets, corresponding changes are not made in the Oracle Identity Manager database entries for these data sets. At the same time, no error message is displayed.

Therefore, for this release of Oracle Identity Manager, Oracle recommends that you do not modify the fields or child data sets of the OIM - Account data set.

This point has also been discussed in [Section 22.2.2, "Step 3: Modify Connector Configuration Page"](#).

21.3.2 Exporting Generic Technology Connectors

You can export the XML file of a generic technology connector. This XML file contains definitions for all the objects that are part of the connector. If you want to use the same generic technology connector on a new Oracle Identity Manager installation, you must first export the XML file and import it into the new Oracle Identity Manager installation.

To export the connector XML file:

1. In the Oracle Identity Manager Advanced Administration, under System Management, click **Export Deployment Manager File**.
2. On the first page of the Deployment Manager Wizard, select **Generic Connector** from the list and click **Search**.
3. In the search results, select the generic technology connector whose XML file you want to export.
4. Click **Select Children**.
5. For the selected generic technology connector, select the child entities that you want to export and click **Select Dependencies**.
6. Select the dependencies that you want to export, and click **Confirmation**.
7. After you verify that the elements displayed on the page cover your export requirements, click **Add for Export**.
8. Click **Exit wizard and show full selection**, and click **OK**.

21.3.3 Importing Generic Technology Connectors

To copy a generic technology connector to a different Oracle Identity Manager installation:

1. If the connector uses custom providers, you must copy the files created during provider creation to the appropriate directories on the destination Oracle Identity Manager installation.

See Also: ["Chapter 20, "Creating Custom Providers for Generic Technology Connectors"](#) for more information about these provider files and the directories into which you must copy them

2. Export the connector XML file on the source Oracle Identity Manager installation.
3. Import the connector XML file on the destination Oracle Identity Manager installation.

Caution: You must ensure that the names you select for a generic technology connector and its constituent objects on a staging server do not cause naming conflicts with existing connectors and objects on the production server.

The following scenario explains why you must follow this guideline:

Suppose you create a generic technology connector on a staging server, and want to import the connector to a production server. While creating the generic technology connector on the staging server, you would have ensured that the names of the generic technology connector and the connector objects are unique on that server. At the same time, you must also ensure that the names are not the same as the names of connectors and connector objects on the production server.

If any of the names happen to be the same, the old objects would be overwritten by the new objects when you import the connector XML file from the staging server to the production server. No message is displayed during the overwrite process, and the process would lead to eventual failure of the affected connectors.

This is also mentioned in [Section 22.2.1, "Names of Generic Technology Connectors and Connector Objects"](#)

To ensure that you are able to revert to a working state in the event that an object is overwritten, you must create a backup of the destination Oracle Identity Manager database before you import a connector XML file.

To import the connector XML file:

1. In the Oracle Identity Manager Advanced Administration, under System Management, click **Import Deployment Management File**. A dialog box for locating files is displayed.
2. Locate and open the connector XML file from the directory into which you copy it.
3. Click **Add File**.
4. Click **Next**, **Next**, and **Skip**.
5. Click **View Selections**.

The contents of the connector XML file are displayed on the Import page. You *may* see a cross-shaped icon along with some nodes. These nodes represent Oracle Identity Manager entities that are redundant. Before you import the connector XML file, you must remove these entities by right-clicking each node and selecting **Remove**.

6. Click **Import**. The connector file is imported into Oracle Identity Manager.

After you import the connector XML file, you must update the run-time parameters of the generic technology connector.

Note: These values are not copied in the connector XML file when you export it.

To update the values of the run-time parameters, follow the procedure described in [Section 21.5.7, "Modifying Generic Technology Connectors"](#).

21.4 Using the Generic Connection Pool Framework in Custom Connectors

Custom connectors can choose to use the Generic Connection Pool framework (sometimes referred to as the GCP) for any connection pooling needs.

Internally, the Generic Connection Pool framework uses Oracle Universal Connection Pool (UCP) as the default connection pooling mechanism.

Basic steps to use the Generic Connection Pool in a custom connector include:

1. Provide a concrete implementation for the `ResourceConnection` interface.
The implementation should also have a default constructor with no parameters.
2. Define the additional fields in the `ITResource` definition.
3. Invoke the Generic Connection Pool to obtain and release connections from the pool.

Topics in this section include:

- [Providing concrete implementation for ResourceConnection interface](#)
- [Defining Additional ITResource Parameters](#)
- [Getting and Releasing Connections from the Pool](#)
- [Using a Third-party Pool](#)
- [Example: Implementation of ResourceConnection](#)

21.4.1 Providing concrete implementation for ResourceConnection interface

The connection pool makes use of the concrete implementation of `ResourceConnection` to create and close connections, and to validate connections to the target. Thus, you should ensure that this concrete implementation class is available as a jar file under the `JavaTasks` folder.

[Table 21–5](#) describes key methods of `ResourceConnection`:

Table 21–5 Methods of ResourceConnection

Method	Description
Create Connection	<p>This method is called while initializing the pool (to create initial number of connections) and for pool life-cycle events as needed. A hashmap named <code>itResourceInfoMap</code> is available as parameter with <code>ITResource</code> values to this method.</p> <p>The method returns the <code>ResourceConnection</code> which is the actual physical connection to the target.</p>
Close Connection	The pool invokes this method when it needs to close a connection in the course of pool life-cycle events.
Heartbeat	This method is used to maintain the TCP heartbeat (or TCP keepalive) of the connection to the target. The method keeps the TCP connection alive, so that the connection does not time out from the target side.
Validate	<p>This method returns <code>true</code> or <code>false</code> to indicate whether the connection is still valid.</p> <p>The Generic Connection Pool invokes the method if "validate connection on borrow" is set to <code>true</code>. It is invoked for connections that have been in the pool for some time.</p> <p>If the method returns <code>false</code>, the pool will discard that connection, create a new connection, and return to the requester.</p>

21.4.2 Defining Additional ITResource Parameters

[Table 21–6](#) lists other `ITResource` parameters for which you should provide appropriate values:

Table 21–6 ITResource Parameters

Field	Description	Sample Value and Notes
Abandoned connection timeout	Connection timeout for abandoned connections in seconds. After the timeout elapses, the connection is reclaimed.	900
Connection wait timeout	Wait time in seconds for a connection to establish.	60
Inactive connection timeout	Connection timeout, in seconds, for inactive connections in the pool that are idle. <i>Note:</i> These are not borrowed connections.	300
Initial pool size	Initial number of connections in the pool.	3
Max pool size	Maximum number of connections that the pool can create.	30
Min pool size	Minimum number of connections that the pool must maintain.	2
Validate connection on borrow	Indicates if connections should be validated. See Table 21–5 for a detailed explanation.	true or false
Timeout check interval	Frequency, in seconds, at which to check timeout properties.	30
Pool preference	Denotes the preferred pooling mechanism. Default pool implementation is UCP.	"Default" (for UCP). "Native" (for Native implementation)
Connection pooling supported	Denotes whether pooling is supported. If pooling is not supported, returned connections will not be pooled connections. Recommended default is <code>true</code> .	true or false.

Table 21–6 (Cont.) ITResource Parameters

Field	Description	Sample Value and Notes
Target supports only one connection	Denotes whether the target system supports only one connection at a time. When set to true, irrespective of other properties, the following pool parameters are used: <ul style="list-style-type: none"> ■ Min Pool Size = 0 ■ Initial Pool Size = 0 ■ Max Pool Size = 1 Recommended default is false	true if target can handle only one connection, false otherwise.
ResourceConnection class definition	The concrete implementation of the ResourceConnection class	com.oracle.oim.ad.ADResourceConnectionImpl
Native connection pool class definition	The wrapper to the native pool mechanism that implements the GenericPool. Set a value only if the pool preference is set to Native.	com.oracle.oim.ad.ADNativePool
Pool excluded fields	Comma-separated list of fields not needed for creating a connection. When any of the specified fields are updated, the GCP pool is <i>not</i> refreshed. Note: Fields in this list are not available as part of the HashMap parameter to the createConnection method.	Recon TimeStamp,ADSync Enabled

Note the following:

- Updating the ITResource parameters from the Design Console does not refresh the pool. Update values through the Administrative and User Console or through the APIs.
- Avoid updating values when the pool is in use.

21.4.3 Getting and Releasing Connections from the Pool

Consumers of the Generic Connection Pool can invoke the ConnectionService to get pooled connections to the target, and also to return connections back to the pool.

This example code gets a connection from the pool and returns it based on ITResource Name:

```
import com.oracle.oim.gcp.exceptions.ConnectionServiceException;
import com.oracle.oim.gcp.pool.ConnectionService;
import com.oracle.oim.gcp.resourceconnection.ResourceConnection;

public class ConnectionPoolClient {

    public void testConnection(String itResName)
    {
        try{
            //Request for connection from the connection pool
            ConnectionService service = new ConnectionService();
            ResourceConnection myConnection =
            service.getConnection(itResName);

            //"myConnection" is the connection
            //use the connection...

            //Release connection back to the connection pool
        }
    }
}
```

```

        //Connections should always be returned this way.
        service.releaseConnection(myConnection);
    }
    catch(ConnectionServiceException e)
    {
        //handle
    }
}

```

You can also request connections to the target using `ITResource Key`. Here is an example:

```

ConnectionService service = new ConnectionService();
ResourceConnection myConnection = service.getConnection(itResourceKey);

```

21.4.4 Using a Third-party Pool

As mentioned earlier in the section, you can use any third-party pool for your custom connector. However, in addition to the steps described earlier, you must provide a concrete implementation of the `GenericPool` interface as a wrapper to the third-party pool.

Note: If the custom connector does not wish to use the UCP pool, it can choose to use GCP with the Native option, though there are no significant advantages to this. With the Native pool preference, the responsibility of maintaining and implementing the pool rests with the custom connector.

Table 21–7 lists the methods invoked for the `GenericPool` interface:

Table 21–7 *Methods of the GenericPool Interface*

Method	Purpose
<code>initializePool(PoolConfiguration poolConfig)</code>	To initialize the pool. The <code>PoolConfiguration</code> data object contains all pool-related parameters.
<code>borrowConnectionFromPool()</code>	To request a connection.
<code>returnConnectionToPool(ResourceConnection resConn)</code>	To return a connection to the pool.
<code>refreshPool(PoolConfiguration newPoolConfig)</code>	To refresh the pool with updated values.
<code>destroyPool()</code>	To remove the pool (for example when <code>ITResource</code> is deleted).

21.4.5 Example: Implementation of ResourceConnection

This example demonstrates an implementation of the `ResourceConnection` interface. Key methods are highlighted.

Example 21–1 *An Example of ResourceConnection Implementation*

```

/**
 * Sample implementation for Socket Connections:
 */
import java.io.IOException;

```

```

import java.net.InetSocketAddress;
import java.net.Socket;
import java.net.SocketException;
import java.net.UnknownHostException;

import com.oracle.oim.gcp.exceptions.ResourceConnectionCloseException;
import com.oracle.oim.gcp.exceptions.ResourceConnectionCreateException;
import com.oracle.oim.gcp.exceptions.ResourceConnectionValidationException;
import com.oracle.oim.gcp.resourceconnection.ResourceConnection;

public class SocketResourceConnectionImpl extends Socket implements
ResourceConnection {
    public SocketResourceConnectionImpl() {
        super();
    }
    /**
     * Sample: Concrete implementation for closing a socket connection
     */
    public void closeConnection() throws ResourceConnectionCloseException{
        if(!this.isClosed()){
            try {
                this.close();
            } catch (IOException e) {
                throw new
                ResourceConnectionCloseException("[Client
                ResourceConnection implementation]
                Failed to close socket connection! ");
            }
        }
    }
    /**
     * Sample : Concrete implementation for creating a socket connection.
     * The return value is the actual physical socket connection
     *
     */
    public ResourceConnection createConnection(HashMap itResInfoMap)
    throws ResourceConnectionCreateException {
        ResourceConnection r = null;
        SocketResourceConnectionImpl i = new
        SocketResourceConnectionImpl();

        try {
//HashMap has all ITResource related information that is needed
//for connecting to target.
            String serverAddress= ((String) itResInfoMap.get
            ("Server Address")).trim();
//utility method getIntValue returns an int for a String

            int port =
                getIntValue(((String)itResInfoMap.get("Port")).trim());

            System.out.println("Connecting to Socket with IP Address "
                + serverAddress+" at port "+ port);
            InetSocketAddress inet = new
            InetSocketAddress(serverAddress,port);
            i.connect(inet);
            if(!i.isConnected()){
                throw new ResourceConnectionCreateException
                (" Failed to create socket: connection failure");
            }
        }
    }
}

```

```
    }
    r = (ResourceConnection)i;
        } catch (UnknownHostException e) {
            throw new ResourceConnectionCreateException("
            [Client ResourceConnection implementation]
            Failed to create socket connection!", e);
        } catch (IOException e) {
            throw new ResourceConnectionCreateException("
            [Client ResourceConnection implementation]
            Failed to create socket connection! ",e);
        }
    }
    return r;
}
/**
 * Sample : Concrete implementation for heartbeat of a socket connection
 */
public void heartbeat() throws ResourceConnectionValidationxception {
    try {
        this.setKeepAlive(true);

    } catch (SocketException e) {
        throw new
        ResourceConnectionValidationxception
        ("[Client ResourceConnection implementation]
        Failed to set heartbeat ");
    }
}
/**
 * Sample: Concrete implementation for validating connection
 */
public boolean isValid() {
    if(this.isBound()){

        return true;

    }else{
        return false;
    }
}
}
```

21.5 Best Practices

This section contains these topics:

- [Working with the Provide Basic Information Page](#)
- [Working with the Specify Parameter Values Page](#)
- [Working with the Modify Connector Configuration Page](#)
- [Working with Shared Drive Reconciliation Transport Provider](#)
- [Working with Custom Providers](#)
- [Working with Connector Objects](#)
- [Modifying Generic Technology Connectors](#)

21.5.1 Working with the Provide Basic Information Page

Apply the following guidelines while specifying a name for a generic technology connector:

- **Summary:**

Ensure that the name contains only ASCII characters. You can include the underscore (_) character, but do not include any other non-ASCII character in the name.

- **Description:**

For most of the connector objects that are automatically created at the end of the connector creation process, the name of the generic technology connector is part of the name of the object itself. For example, if the name of the generic technology connector is `WebConn`, the name of its scheduled task is `WebConn_GTC`.

In the Oracle Identity Manager database, there is no provision for storing objects with names in non-ASCII characters. Therefore, an error message is displayed if you enter non-ASCII characters while specifying the name of a generic technology connector.

- Ensure that the name is not the same as the name of any connector or connector object on the Oracle Identity Manager installation.
- If you plan to create the generic technology connector on a staging server and move it to a production server, ensure that the name of the generic technology connector does not cause naming conflicts with existing connectors or connector objects on the production server.
- Before you import a generic technology connector created on a staging server to a production server, create a backup of the destination Oracle Identity Manager database to ensure that you are able to revert to a working state in the event that a connector object is overwritten.
- If you select the shared drive transport provider, you must select the CSV format provider.
- If you select the SPML provisioning format provider, you must select the Web Services provisioning transport provider.
- If you select the shared drive reconciliation transport provider, ensure that there is data in the prescribed format on at least the first two lines of the parent and child data files provided by the target system for reconciliation. The prescribed form of data is discussed in [Section 19.1, "Shared Drive Reconciliation Transport Provider"](#).
- If you select the shared drive reconciliation transport provider, ensure that the required permissions are set on the staging and archiving directories before reconciliation begins. The required permissions are discussed in the "Permissions to Be Set on the Staging and Archiving Directories" section.
- Do not try to create more than one generic technology connector at a time, even from different sessions of the Administrative and User Console for the same Oracle Identity Manager installation.

21.5.2 Working with the Specify Parameter Values Page

This section describes the following known issues related to the input that you specify on the Step 2: Specify Parameter Values page:

- **Summary:**

If you use the shared drive reconciliation transport provider, :

- Do not specify the same path for the staging and archiving directories. Existing files in the archiving directory are deleted if you specify the same path for both directories.
- Ensure that the names of files in the staging directory are different from the names of files in the archiving directory. If the file names happen to be the same, existing files in the archiving directory are overwritten at the end of a reconciliation run.

Description:

When you use the shared drive reconciliation transport provider, after each reconciliation run, data files are moved from the staging directory to the archiving directory. The files moved to the archiving directory are not time-stamped or marked in any way. Therefore, when you use the shared drive transport provider, bear in mind the following guidelines:

- The archiving directory path and name that you specify must not be the same as the staging directory path and name. If you specify the same path and name, the existing files in the archiving directory are deleted at the end of the reconciliation run.
- During the current reconciliation run, if data files with the same names as the files used in the last reconciliation run are placed in the staging directory, the existing files in the archiving directory are overwritten by the new files from the staging directory. This can be illustrated by the following example:

Suppose that at the end of the last reconciliation run, the following files were moved automatically from the staging directory to the archiving directory:

```
usrdataParentData.csv
usrdataRoleData.csv
usrdataGroupMembershipData.txt
```

For the current reconciliation run, you place the following files in the staging directory:

```
usrdataParentData.csv
usrdataRoleData_04Feb07.csv
usrdataGroupMembershipData_04Feb07.txt
```

At the end of the current reconciliation run, these files are moved to the archiving directory. When this happens, the old `usrdataParentData.csv` file is overwritten by the new one.

Therefore, if you want to ensure that files in the archiving directory are not overwritten at the end of a reconciliation run, you must ensure that the names of files in the staging directory are not the same as the names of files in the archiving directory.

■ **Summary:**

Metadata detection does not take place a second time if you go back to the Step 2: Specify Parameter Values page. Therefore, if required, you must manually make changes in the fields and field mappings displayed on the Step 3: Modify Connector Configuration page.

Description:

Suppose you want to change a value that you provide on the Step 2: Specify Parameter Values page. You can return to this page from the Step 4: Verify

Connector Form Names or Step 5: Verify Connector Information page. However, metadata detection would not take place a second time when you click the Continue button after changing the provider parameter value. This functionality is aimed at preserving changes that you may have manually made on the Step 3: Modify Connector Configuration page.

As an extension of this functionality, metadata detection does not take place even when you modify an existing generic technology connector.

21.5.3 Working with the Modify Connector Configuration Page

This section discusses best practices related to the following areas:

- [Names of Fields](#)
- [Password Fields](#)
- [Password-Like Fields](#)
- [Mappings](#)
- [Oracle Identity Manager Data Sets](#)

21.5.3.1 Names of Fields

Note that the following validations are applied when you specify a field name while adding or editing fields:

- Two fields that belong to the same data set (parent or child) cannot have the same name.
- Two child data sets of the same parent data set cannot have the same name.
- The name of a field in a parent data set cannot be the same as the name of one of its child data sets.
- Two different child data sets can have fields that have the same name, regardless of whether or not the child data sets belong to the same parent data set. For example, the `GroupMembership` data set and `Role` data set can each have a field with the name `UsrID`.
- Two different parent data sets can have fields that have the same name. Similarly, these data sets can also have child data sets that have the same name.
- The name of a child data set can be the same as that of one of its fields.

21.5.3.2 Password Fields

To ensure the security of passwords, password information must not be reconciled through a generic technology connector. In other words, you must ensure that the Source and reconciliation staging data sets do not contain the Password field. In addition, you must not map any field of the reconciliation staging data sets to the Password field of the OIM - User data set.

21.5.3.3 Password-Like Fields

A password-like field is a field to which you set the Encrypted and Password Field attributes (by selecting the Encrypted and Password Field check boxes). You can create a password-like field by setting these two attributes to a field that you add to the OIM - Account data set.

To secure the contents of password-like fields, bear in mind the following guidelines while adding or editing these fields:

- You can use the Password Field and Encrypted check boxes to secure the display and storage of password information in Oracle Identity Manager. However, when you map password-like fields to fields of the provisioning staging data set, you must take all necessary precautions to secure the data propagated in these fields. For example, you must ensure that this data is not stored in a plain-text file on the target system at the end of a provisioning operation.

Oracle recommends creating only one-to-one mappings between the password field of the OIM - Account data set and the provisioning staging data set. In other words, this password field must not be used as an input field for a transformation mapping with a provisioning staging field. The same precaution must be taken for the Password field of the OIM - User data set.

- As mentioned earlier, the Password field is one of the predefined fields of the OIM - User data set. The Password Field and Encrypted attributes are set for this field. By using the Design Console, you can set the Password Field and Encrypted attributes for a UDF that you create. This would give the newly created UDF the same properties as the existing Password field. However, the generic technology connector framework treats this field the same as any other text field (with the String data type) and the contents are not encrypted in the Administrative and User Console or database.

See also [Section 22.1, "General Issues for Generic Technology Connectors"](#).

21.5.3.4 Mappings

Apply the following best practices while working with fields of the Oracle Identity Manager data sets:

- **Summary:**

If you select the translation transformation provider to create a mapping, specify the name of a lookup definition in the Lookup Code Name region. If you specify a data set name and field in the Lookup Code Name region, translation would fail during actual reconciliation or provisioning operations.

- **Description:**

If you select the translation transformation provider while creating a mapping, the Step 2: Mapping page displays options for selecting a field from a data set and specifying a literal. Because you are using the translation transformation provider, you must select the Literal option and enter the name of the lookup definition that contains the Code Key and Decode values for the translation. You must not select a data set name and field in the Lookup Code Name region. Although there is no validation to stop you from selecting a data set name and field, the translation operation would fail during actual reconciliation or provisioning operations.

- Create a mapping between the ID field of the OIM - Account data set and the field that uniquely identifies records of the reconciliation staging data set.
- Along with the ID field, other fields of the OIM - Account data set can be (matching-only) mapped to corresponding fields of the reconciliation staging data set to create a composite key field for reconciliation matching.
- Create mappings between all fields in provisioning staging data sets and corresponding fields in Oracle Identity Manager data sets.
- To create a reconciliation rule, you create matching-only mappings between fields of the reconciliation staging data set and the OIM - User data set. If there are child data sets, ensure that the names of fields of the reconciliation staging data set that are input fields for the matching-only mappings are not used in any of the

reconciliation staging child data sets. If you do not follow this guideline, reconciliation would fail.

This has also been mentioned in the section "Step 3: Modify Connector Configuration Page".

- A literal field can be used as one of the input fields of a transformation mapping. If you select the Literal option, you must enter a value in the field. You must not leave the field blank after selecting it.

21.5.3.5 Oracle Identity Manager Data Sets

Apply the following best practices while working with fields of the Oracle Identity Manager data sets:

- For trusted source reconciliation, the following fields of the OIM – User data set must always hold values:
 - User ID
 - First Name
 - Last Name
 - Organization Name
 - Xellerate Type
 - Role

In addition, you can select other OIM – User fields that must be populated when a user account is created through reconciliation. For each of these fields, you must create mappings with the corresponding fields of the reconciliation staging data sets. During a reconciliation run, you must ensure that the fields of the target system that serve as the source for these fields always hold values.

For provisioning, you can select fields of the OIM – User and OIM – Account data sets whose values must be propagated to the target system. After you identify these fields, create mappings between them and their corresponding fields in the provisioning staging data sets. During a provisioning operation, you must enter values for each of these fields.

- If required, add user-defined fields (UDFs) to the list of predefined OIM - User data set fields. See "Configuring User Attributes" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager* for details.
- Do not modify or delete attributes of OIM - Account data set fields in an existing generic technology connector.

21.5.4 Working with Shared Drive Reconciliation Transport Provider

Summary

If parent records and child data records are created and linked in both the target system and Oracle Identity Manager, you must ensure that the staging directory contains both parent data and child data files at the start of each reconciliation run.

Description

Suppose there are parent data records with associated child data records in the target system. To reconcile these records into Oracle Identity Manager, you place the parent and child data files containing these records in the staging directory. During the reconciliation run, the child data records are linked to their corresponding parent data records. Before the start of any subsequent reconciliation run, if you remove the child

data files from the staging directory, reconciliation events are not created for this form of child data record deletion. If you want to remove child data records for specific parent data records, you must remove the child data records from the child data file. You must ensure that the child data file is placed in the staging directory for each reconciliation run, even if there are no child data records (from the third line onward) in the files.

21.5.5 Working with Custom Providers

Apply the following guideline while working with custom providers:

When you develop code for a custom provisioning transport provider, ensure that the provider returns the unique field value at the end of a Create User operation. This functionality is implemented by the `sendData` method of the provisioning transport provider. See "Role of Providers During Provisioning" for more information.

21.5.6 Working with Connector Objects

Apply the following guidelines while working with connector objects created automatically during generic technology connector creation:

- **Summary:**

Do not attempt to use for provisioning the resource object created automatically for a reconciliation-only generic technology connector.

Description:

Suppose you select only the Reconciliation option while creating a generic technology connector. At the end of the creation process, a resource object is one of the objects created automatically for this generic technology connector. However, you cannot provision this resource object to any user because a generic adapter is not created for a reconciliation-only generic technology connector.

- **Summary:**

Do not attempt to provision generic technology connector resource objects to organizations defined in Oracle Identity Manager.

Description:

A resource object is one of the connector objects that get created automatically during generic technology connector creation. This resource object can be provisioned only to Oracle Identity Manager Users. You must not attempt to provision it to organizations defined in Oracle Identity Manager.

This has also been mentioned in the Connector Objects section .

- You can use the Design Console to customize connector objects that are automatically created during generic technology connector creation. After you customize connector objects, if you perform a Manage Generic Technology Connector operation, all the customization done on the connector objects would be overwritten. Therefore, Oracle recommends that you to apply one of the following guidelines:

- Do not use the Design Console to modify generic technology connector objects.

The exception to this guideline is the IT resource. You can modify the parameters of the IT resource by using the Design Console. However, if you have enabled the cache for the `GenericConnector` and `GenericConnectorProviders` categories, you must purge the cache either

before or after you modify IT resource parameters. See "Purging the Cache" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager* for more information.

- If you use the Design Console to modify generic technology connector objects, do not use the Manage Generic Technology Connector feature to modify the generic technology connector.

This has also been mentioned in [Section 21.5.6, "Working with Connector Objects"](#).

- Prepopulate adapters are not part of the set of connector objects that are created automatically when you create a generic technology connector. However, while creating a generic technology connector, you can map provisioning input to literals and user data fields. Although this feature cannot be used to prepopulate the User Defined Form, it can be used to prepopulate the provisioning data packet.

21.5.7 Modifying Generic Technology Connectors

Apply the following best practice while modifying generic technology connectors:

Do not try to modify more than one generic technology connector at a time, even from different sessions of the Administrative and User Console for the same Oracle Identity Manager installation.

Troubleshooting Generic Technology Connectors

This chapter describes how to troubleshoot problems that you might encounter during development. It contains these sections:

- [General Issues for Generic Technology Connectors](#)
- [Configuration Issues for Generic Technology Connectors](#)

22.1 General Issues for Generic Technology Connectors

This section describes general issues for generic technology connectors. It contains these topics:

- [Creation Issues](#)
- [Multi-language Support](#)
- [Other General Issues](#)

22.1.1 Creation Issues

This section describes the following known issues related to the connector objects that are automatically created by the generic technology connector framework:

- **Summary:**
 - No warning is displayed if the name that you specify for a generic technology connector is the same as the name of an existing connector object.
 - No warning is displayed if an existing connector object is overwritten by a new connector object when you import a connector XML file.

Description:

This point has also been discussed in the "Names of Generic Technology Connectors and Connector Objects" section.

- **Summary:**

After an error occurs during generic technology connector creation, form names are not displayed on the Step 4: Verify Connector Form Names page when you revisit that page by clicking Back on the Step 5: Verify Connector Information page.

This is intentional and not the result of an issue or limitation of the software.

Description:

As mentioned earlier in this guide, some connector objects are automatically created even if the overall generic technology connector creation process fails. This set of connector objects includes process forms whose names you specify on the Step 4: Verify Connector Form Names page. In the event that the connector creation process fails, you are prompted to enter new form names through the display of blank fields on the Step 4: Verify Connector Form Names page. This is to ensure that the uniqueness checks on the process form names are reapplied when you submit the new form names.

As an alternative to revisiting the previous pages and providing input for creating the generic technology connector, you can start all over again from the Step 1: Provide Basic Information page and re-create the generic technology connector.

- **Summary:**

You cannot provision generic technology connector resource objects to organizations defined in Oracle Identity Manager.

Description:

A resource object is one of the connector objects that get created automatically during generic technology connector creation. This resource object can be provisioned only to Oracle Identity Manager Users. You must not attempt to provision it to organizations defined in Oracle Identity Manager.

- **Summary:**

Customization work done on objects of a generic technology connector would be overwritten if you perform a Manage Generic Technology Connector operation.

Description:

You can use the Design Console to customize connector objects that are automatically created during generic technology connector creation. However, after you customize connector objects, if you perform a Manage Generic Technology Connector operation, then all the customization done on the connector objects would be overwritten. Therefore, Oracle recommends that you to apply one of the following guidelines:

- Do not use the Design Console to modify generic technology connector objects.

The exception to this guideline is the IT resource. You can modify the parameters of the IT resource by using the Design Console. However, if you have enabled the cache for the `GenericConnector` and `GenericConnectorProviders` categories, then you must purge the cache either before or after you modify IT resource parameters. See "Purging the Cache" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager* for information about running the `PurgeCache` utility.

- If you use the Design Console to modify generic technology connector objects, then do not use the Manage Generic Technology Connector feature to modify the generic technology connector.
- Connector objects that are automatically created are not deleted even if the generic technology connector creation process fails.

22.1.2 Multi-language Support

This section describes the following known issues related to the Multilanguage Support feature:

- **Summary:**

No warning is displayed if there are non-ASCII characters in the first or second line of the data files in the staging directory.

Description:

There is no support for non-ASCII data in the metadata of target system user data. If you use the CSV Reconciliation format provider, then this limitation means that you cannot include non-ASCII characters in the metadata line (second line) of the parent and child data files that you store in the staging directory.

The reason for this limitation is as follows:

The generic technology connector framework creates User Defined process forms in Oracle Identity Manager and names the forms and their fields on the basis of the input metadata. In addition, database tables and columns are created for these forms and their fields, respectively. Because non-ASCII characters cannot be used in database object names, these characters are not supported in the target system metadata.

The generic technology connector framework may be able to parse and correctly display non-ASCII characters in the first and second lines of the data files. However, to ensure that the connector objects are created correctly, you must ensure that non-ASCII characters are not used in the first and second lines of the data files.

Note: From the third line onward in the data files, the field data values can contain non-ASCII characters. These data values are reconciled and stored in the Oracle Identity Manager database.

- **Summary:**

For any language that Oracle Identity Manager supports, if the browser language setting does not match the operating system language setting of the Oracle Identity Manager server, then data is not displayed correctly on the Step 3: Modify Connector Configuration page.

Description:

The Step 3: Modify Connector Configuration page displays an image that is dynamically created by the generic technology connector framework. The following are limitations related to the display of localized text items on this page:

The language in which you want field labels to be displayed must match the following language settings:

- Oracle Identity Manager language
- Operating system language
- Browser language

If the browser language setting is the same as the operating system language setting of the Oracle Identity Manager server, then all the text items (field names and GUI element labels) are displayed in the required language.

Note:

- Localized GUI element labels are displayed only if you create and use resource bundles that contain localized labels for these GUI elements.
 - If you are using the Traditional Chinese or Simplified Chinese language, then the browser locale (language and country/region) must be the same as the operating system locale (language and country/region) for all the text items to be displayed in the required language.
-

If the browser language is not the same as the operating system language, then the following static labels would be displayed in English (regardless of the browser language):

- Labels of the OIM - User and OIM - Account data sets: "User" and "Account"
- Labels of the fields that constitute the OIM - User data set:
 - * "User ID"
 - * "Email"
 - * "First Name"
 - * "Last Name"

For non-ASCII languages, labels for the remaining items on the Step 3: Modify Connector Configuration page would not be displayed correctly.

- **Summary:**

Certain text items displayed on the Step 3: Modify Connector Configuration page are always displayed in English.

Description:

For this release, some of the static text displayed on the Step 3: Modify Connector Configuration page has not been localized. For example, suppose you create a generic technology connector named MyGTC. When you provision the resource object of this connector to a user, the following text is displayed on the page:

```
Provisioning form for MyGTC
```

```
Child Form of MyGTC representing child-dataset:
```

```
child_data_set_name
```

In this release of Oracle Identity Manager, the static part of this text is always displayed in English.

If required, you can localize the static text as follows:

1. For the language to which you want to localize the text, open the corresponding `customResources.properties` file by importing it from MDS. The files for all the languages that Oracle Identity Manager supports are in MDS. To import the `customResources.properties` file from MDS, refer to [Chapter 33, "MDS Utilities and User Modifiable Metadata Files"](#).

The following example illustrates this step of the procedure.

Suppose you specify the following values while creating a generic technology connector:

- Connector Name: MyGTC
- Parent Form name: ADUser
- Child data set name: ADUserRole
- Child form name: ADURole1

If you want the static text to be displayed in the Spanish language, then open the `customResources_es.properties` file. This file is in MDS.

2. In the `customResources.properties` file for the required language, add the following lines:

Note: You can access the `customResources.properties` file for the required language from the Oracle Identity Manager page on the Oracle Technology Network (OTN) Web site at the following URL:

<http://www.oracle.com/technetwork/index.html>

```
global.UD_PARENT_FORM_NAME.description=Localized_text_for_"Provisioning
form for" GTC_name
```

```
global.UD_CHILD_FORM_NAME.description=Localized_text_for_"Child Form of"
GTC_name Localized_text_for_"representing the child data set":
child_data_set_name
```

In these two lines, replace:

- * `PARENT_FORM_NAME` with the name of the parent form

The parent form name is always converted to uppercase letters in Oracle Identity Manager. Therefore, the name that you enter must be in uppercase letters.

- * `Localized_text_for_"Provisioning form for"` with localized text for the words "Provisioning form for"

- * `GTC_name` with the name of the generic technology connector

- * `CHILD_FORM_NAME` with the name of the child form

The child form name is always converted to uppercase letters in Oracle Identity Manager. Therefore, the name that you enter must be in uppercase letters.

- * `Localized_text_for_"Child Form of"` with localized text for the words "Child form for"

- * `child_data_set_name` with the name of the child data set

For example:

For the Spanish language, add the following lines in the `customResources_es.properties` file:

```
global.UD_ADUSER.description=Spanish_text_for_"Provisioning form for" MyGTC
```

```
global.UD_ADURole1.description=Spanish_text_for_"Child Form of" MyGTC
Spanish_text_for_"representing the child data set": ADUserRole
```

22.1.3 Other General Issues

This section describes the following known issues that do not fall under any of the preceding categories:

- **Summary:**

Unsafe-Filename exceptions may be thrown during the generic technology connector creation process.

Description:

On Oracle WebLogic Server and Oracle Application Server, the Unsafe-Filename exception may be thrown during the generic technology connector creation process. This exception can be ignored. The generic technology connector creation process is not affected by the occurrence of these exceptions. This issue is not seen on IBM WebSphere Application Server and JBoss Application Server.

- Generic technology connectors do not support the reconciliation of parent data deletion.

You cannot use a generic technology connector to reconcile the deletion of parent data. For example, if the account of user `John Doe` is deleted from the target system, then you cannot use a generic technology connector to reconcile this user deletion in Oracle Identity Manager.

- **Summary:**

The contents of a UDF are not encrypted if the Password Field and Encrypted attributes have been set for the field by using the Design Console.

Description:

As mentioned earlier, the Password field is one of the predefined fields of the OIM - User data set. The Password Field and Encrypted attributes are set for this field. By using the Design Console, you can set the Password Field and Encrypted attributes for a UDF that you create. This would give the newly created UDF the same properties as the existing Password field. However, the generic technology connector framework treats this field the same as any other text field (with the String data type) and the contents are not encrypted in the Administrative and User Console or database.

- In this release of Oracle Identity Manager, the generic technology connector framework does not provide some of the functionality that the Design Console offers for creating reconciliation rules. Only reconciliation rules of the following pattern can be created:

```
A equals B
```

```
"and"
```

```
C equals D
```

```
"and"
```

```
E equals F
```

- While creating a generic technology connector, you cannot specify that the target system requires a remote manager to communicate with the target system. Therefore, a generic technology connector cannot use a remote manager.
- You use the Target Date Format parameter to specify the format in which date values must be sent to the target system during provisioning. Date validation for this parameter does not take place if you enter a date in numeric format. For

information about the date formats that you can specify, see the following Web page:

<http://java.sun.com/docs/books/tutorial/i18n/format/simpleDateFormat.html#datepattern>

- Scheduled tasks that are not currently running have the `INACTIVE` status. These tasks run at the next specified date and time. Under certain conditions, a scheduled task is automatically assigned the `NONE` status. However, this status change does not affect the functionality of the task, which continues to run at the specified date and time.
- When you import a release 9.0.3 generic technology connector into a release 9.1.0.1 Oracle Identity Manager installation, a non-fatal exception is recorded in the application server log file.

This occurs only if the connector supports provisioning, regardless of whether or not it supports reconciliation. You can ignore this exception message. No error message is displayed on the Administrative and User Console.

- During a Manage Generic Technology Connector operation, if you change the data type of a field in the OIM - Account data set, then an error is thrown when you click Create on the Step 5: Verify Connector Information page.

22.2 Configuration Issues for Generic Technology Connectors

This section contains these topics:

- [Names of Generic Technology Connectors and Connector Objects](#)
- [Step 3: Modify Connector Configuration Page](#)
- [Errors During Connector Creation](#)
- [Errors During Reconciliation](#)
- [Errors During Provisioning](#)

22.2.1 Names of Generic Technology Connectors and Connector Objects

This section describes the following known issues related to the names that you specify for generic technology connectors and connector objects:

Summary:

- No warning is displayed if the name that you specify for a generic technology connector is the same as the name of an existing connector object.
- No warning is displayed if an existing connector object is overwritten by a new connector object when you import a connector XML file.

Description:

During the creation or modification of a generic technology connector, various objects are automatically created or modified by the generic technology connector framework. You are prompted to specify names for the generic technology connector and process forms. The framework automatically generates names for the remaining objects. These autogenerated names are based on the name that you specify for the generic technology connector.

When you specify a name for the generic technology connector, you must ensure that the name is unique across all object categories (such as resource objects and IT resources) for that Oracle Identity Manager installation. Similarly, you must also

ensure that the process form names are unique. This guideline must be followed even while importing a generic technology connector XML file to a different Oracle Identity Manager installation. You must ensure that the names of objects defined in the XML file are not the same as the names of objects belonging to the same category on the destination Oracle Identity Manager installation. For example, the name of the scheduled task defined in the XML file must not be the same as the name of any other scheduled task on the destination Oracle Identity Manager installation.

The scope of this guideline covers all connector objects, regardless of whether the object is used by a predefined connector or a generic technology connector on the destination Oracle Identity Manager installation.

If you do not follow this guideline, then existing objects that have the same name as imported objects are overwritten during the XML file import operation. No message is displayed during the overwrite process, and the process leads to eventual failure of the affected connectors.

This point has also been discussed in the "Connection Objects" section .

22.2.2 Step 3: Modify Connector Configuration Page

This section describes the following known issues related to the input that you specify on the Step 3: Modify Connector Configuration page:

- **Summary:**

While modifying an existing generic technology connector, if you modify the fields or child data sets of the OIM - Account data set, then corresponding changes are not made in the Oracle Identity Manager database entries for the forms that are based on these data sets. At the same time, no error message is displayed.

Description:

The Step 3: Modify Connector Configuration page provides features to add, modify, and delete fields and field mappings. You can use these features to modify the length or data type of fields in the OIM - Account data set or its child data sets. However, this action would not translate into corresponding changes in the Oracle Identity Manager database entries for these data sets. At the same time, no error message is displayed.

This issue will be fixed in a future release of Oracle Identity Manager. Until then, you must not make changes in the fields or child data sets of the OIM - Account data set.

- **Summary:**

Suppose you create a generic technology connector, use it for provisioning or reconciliation, and then delete fields or child data sets of the OIM - Account data set. An error occurs the next time you perform provisioning or reconciliation by using the same generic technology connector.

Description:

Suppose you create a generic technology connector and then use it for provisioning or reconciliation. You then delete some fields or child data sets of the OIM - Account data set of this generic technology connector. The next time you perform provisioning or reconciliation by using the same generic technology connector, an exception is thrown.

After you use the generic technology connector for provisioning or reconciliation even once, deleting the fields or child data sets of the OIM - Account data set is an invalid operation. This is because data linked to the fields or child data sets that you delete has already been stored in the Oracle Identity Manager database.

Therefore, you must not delete fields or child data sets of the OIM - Account data set if the generic technology connector has already been used to perform provisioning or reconciliation.

In a future release, an appropriate error message will be displayed instead of the exception that is thrown at present.

- **Summary:**

If the name of a reconciliation staging field used in a matching-only mapping were to be reused as the name of a field in a reconciliation staging child data set, then reconciliation would fail.

Description:

You create a reconciliation rule by creating matching-only mappings between fields of the reconciliation staging data set and OIM - User data set. If there are child data sets, then you must ensure that the names of fields of the reconciliation staging data set that are input fields for the matching-only mappings are not used in any of the reconciliation staging child data sets. If the name of a reconciliation staging field used in a matching-only mapping were to be reused as the name of a field in a reconciliation staging child data set, then reconciliation would fail.

The following example illustrates this scenario:

The `AD_User` data set is the reconciliation staging parent data set. The following are the fields of this data set:

- User ID
- Name
- Designation
- Location

The `Admin_Groups` data set is a child data set of the `AD_User` data set. If you use the `User ID` field of the `AD_User` data set to create a matching-only mapping with the OIM - User data set, then you cannot have a field with the name `User ID` in the `Admin_Groups` data set. If this child data set were to contain a field with the name `User ID`, then reconciliation would fail.

- **Summary:**

The Password field is displayed in the OIM - User data set, even though this field is not reconciled by the reconciliation engine.

Description:

If you select the Trusted Source Reconciliation option on the Step 1: Provide Basic Information page, then the Password field is displayed in the OIM - User data set on the Step 3: Modify Connector Configuration page, even though this field is not reconciled by the reconciliation engine. If you create a mapping between this field and the corresponding target system field in the reconciliation staging data set, then the reconciliation field mapping that is automatically generated would try to map the field to the Password field. This, in turn, would cause the reconciliation event to fail.

- There are limitations related to creating transformation mappings across the following data sets:
 - Source and reconciliation staging
 - Oracle Identity Manager and Provisioning Staging

These limitations are as follows:

- You cannot create a transformation mapping between a child data set of the Source or Oracle Identity Manager data set and a different (that is, not corresponding) child data set of the reconciliation staging or provisioning staging data sets. This also means that you cannot create a many-to-one mapping from multiple child data sets of one parent data set to a single child data set of another parent data set.

The following example illustrates this limitation:

Suppose the Source parent data set has the following child data sets:

MyGTC:Group data set

- * Field 1: Group Name
- * Field 2: Group Type

MyGTC:Role data set

- * Field 1: Role Name
- * Field 2: Role Type

Suppose the reconciliation staging parent data set has the following child data sets:

MyGTC:Group data set

- * Field 1: Group Name
- * Field 2: Group Type

MyGTC:Role data set

- * Field 1: Role Definition

According to this limitation, you cannot create a transformation mapping between, for example, the Group Name field of the Source data set and the Role Definition field of the reconciliation staging data set.

However, you can create a many-to-one transformation mapping between, for example, the Role Name and Role Type fields of the Source data set and the Role Definition field of the reconciliation staging data set.

- You cannot create a transformation mapping between a Source or Oracle Identity Manager parent data set and a reconciliation staging or provisioning staging child data set.

The following example illustrates this limitation:

Suppose the following are Oracle Identity Manager data sets and their fields:

OIM - Account data set

- * Field 1: Name
- * Field 2: Address
- * Field 3: User ID

* ...

Suppose the following are provisioning staging child data sets and their fields:

Group data set

* Field 1: Group Name

* Field 2: Group Type

According to this limitation, you cannot create a transformation mapping between, for example, the Name field of the OIM - Account data set and the Group Name field of the Group data set.

- To create a reconciliation rule, you create matching-only mappings between fields of the reconciliation staging data set and the OIM - User data set. If there are child data sets, then ensure that the names of fields of the reconciliation staging data set that are input fields for the matching-only mappings are not used in any of the reconciliation staging child data sets.

If this guideline is not followed, then reconciliation would fail.

- Suppose you set the Date data type for a field on a child form. A Delete Child Record provisioning operation would fail if there is a date value in this field during the operation.

22.2.3 Errors During Connector Creation

The following are error messages that may be displayed at the end of the generic technology connector creation process. Each message explains the event that causes or during which the error message is displayed.

- An error was encountered while generating the import XML file for generic technology connector *connector_name*.
- An error was encountered while updating the IT resource parameters with the values provided for the run-time provider parameters of generic technology connector *connector_name*.
- An error was encountered while either generating the XML file for generic technology connector *connector_name* or saving it in the Oracle Identity Manager database.
- An error was encountered while importing the XML file for generic technology connector *connector_name*. The required lock on the import operation could not be acquired.
- An error was encountered while saving the information for generic technology connector *connector_name*. Check the application logs for more details.
- An error was encountered while creating a resource object for the generic technology connector *connector_name*. An existing resource object has the same name as the one being assigned to this resource object.

22.2.4 Errors During Reconciliation

Table 22-1 provides solutions to some commonly encountered problems associated with the reconciliation process.

Note: These errors are logged only if you are using the shared drive reconciliation transport provider and the CSV Reconciliation format provider.

If any of these errors occurs, then the error message is written to the application server log file.

Table 22–1 Common Errors Encountered During Reconciliation

Problem Description (Error Message)	Solution
No run time provider parameters available	Use the Manage Generic Technology Connector feature to check the values specified for the run-time parameters. Then, retry reconciliation.
No design time provider parameters available	Use the Manage Generic Technology Connector feature to check the values specified for the design parameters. Then, retry reconciliation.
Staging directory location is not defined	Use the Manage Generic Technology Connector feature to check the value specified for the Staging Directory (Parent Identity Data) parameter. Then, retry reconciliation.
File encoding is not defined	Use the Manage Generic Technology Connector feature to check the value specified for the File Encoding (Parent Data) parameter. Then, retry reconciliation.
Archive directory location is not defined	Use the Manage Generic Technology Connector feature to check the value specified for the Archiving Directory parameter. Then, retry reconciliation.
Cannot process files as not even fixed-width delimiter has been defined	Use the Manage Generic Technology Connector feature to check if a value has been specified for one of the following parameters: <ul style="list-style-type: none"> ■ Specified Delimiter ■ Tab Delimiter ■ Fixed Column Width Then, retry reconciliation.
No Parent files in staging directory No files available for reading	Ensure that data files are present in the directory specified as the value of the Staging Directory (Parent Identity Data) parameter. Then, retry reconciliation.
No child data present in staging directory No files available for reading	Ensure that data files are present in the directory specified as the value of the Staging Directory (Multivalued Identity Data) parameter. Then, retry reconciliation.
The staging directory cannot be accessed. Either the directory path does not exist or necessary access permissions are missing	Ensure that the directories specified as parameter values have the required permissions. See Section 19.1, "Shared Drive Reconciliation Transport Provider" for information about the required permissions. Then, retry reconciliation.
Data files could not be read as its File encoding is not supported.	Use the Manage Generic Technology Connector feature to check the value specified for the File Encoding parameter. Then, retry reconciliation.
Not able to parse metadata	Check the metadata (contents of the second row) present in the parent and child data files. There may be a problem with the delimiter used in the files. Fix the problem, and then retry reconciliation.
Not able to parse header	Check the header (contents of the first row) of the data files. There may be a problem in the format of the header. See Section 19.1, "Shared Drive Reconciliation Transport Provider" for information about the header format. Fix the problem, and then retry reconciliation.

Table 22–1 (Cont.) Common Errors Encountered During Reconciliation

Problem Description (Error Message)	Solution
Current Record is erratic and cannot be parsed	Check the entry that is written to the application server log file. It may contain errors that cannot be parsed. Fix the problem, and then retry reconciliation.

22.2.5 Errors During Provisioning

Table 22–2 provides solutions to some commonly encountered problems associated with the provisioning process.

Note: Most of these errors are logged only if you are using the Web Services provisioning transport provider and the SPML provisioning format provider.

If any of these errors occurs, then the error message is displayed on the UI and written to the application log file.

Table 22–2 Common Errors Encountered During Provisioning

Problem Description	Solution
Response code: SPML Velocity Properties Not Read Response Description: The SPML template properties could not be read.	There is a problem with the Oracle Identity Manager installation. Contact Oracle Support, and send them information about this problem and the response code and description displayed. In addition, send the relevant logs generated after running Oracle Identity Manager with logging set to the DEBUG level.
Response code: SPML Template Not Read Response Description: The SPML template file was not found.	There is a problem with the Oracle Identity Manager installation. Contact Oracle Support, and send them information about this problem and the response code and description displayed. In addition, send the relevant logs generated after running Oracle Identity Manager with logging set to the DEBUG level.
Response code: SPML Unknown Operation Response Description: This provisioning operation is not one of the permitted operations: Create, Delete, Enable, Disable, Modify, and Child Table Operations.	There is a problem with the Oracle Identity Manager installation. Contact Oracle Support, and send them information about this problem and the response code and description displayed. In addition, send the relevant logs generated after running Oracle Identity Manager with logging set to the DEBUG level.
Response code: SPML Provisioning Input Null Response Description: SPML provisioning input data is null.	Check if the provider parameters have been correctly specified. Check if provisioning was initiated by direct provisioning or request provisioning. Retry the procedure by using the direct provisioning option.

Table 22–2 (Cont.) Common Errors Encountered During Provisioning

Problem Description	Solution
<p>Response Code: SPML Template Context Processing Error</p> <p>Response Description: An error was encountered while processing the template context for generation of SPML request.</p>	<p>There is a problem with the Oracle Identity Manager installation. Contact Oracle Support, and send them information about this problem and the response code and description displayed. In addition, send the relevant logs generated after running Oracle Identity Manager with logging set to the DEBUG level.</p>
<p>Response code: SPML Provisioning Operation Name Missing</p> <p>Response Description: The operation name for provisioning is missing.</p>	<p>The generic technology connector may not have been created correctly. Try creating another connector by using the same set of configurations (providers) but with fewer attributes. Try direct provisioning.</p>
<p>Response code: SPML Provisioning Child Name Missing</p> <p>Response Description: The child name is missing.</p>	<p>You may have been trying to perform provisioning for one particular type (for example, role or membership) of multivalued attribute when this error occurred.</p> <p>The connector may not have been created correctly. Try creating another connector by using the same set of configurations (providers) but only one multivalued attribute, which is the one that failed the first time. Try direct provisioning.</p>
<p>Response code: SPML Provisioning Child Meta-Data Null</p> <p>Response Description: The child metadata list is null.</p>	<p>You may have been trying to perform provisioning for one particular type (for example, role or membership) of multivalued attribute when this error occurred.</p> <p>The connector may not have been created correctly. Try creating another connector by using the same set of configurations (providers) but only one multivalued attribute, which is the one that failed the first time. Try direct provisioning.</p>
<p>Response code: SPML Provisioning Child Metadata Problem</p> <p>Response Description: An error was encountered while sorting the child metadata list.</p>	<p>You may have been trying to perform provisioning for one particular type (for example, role or membership) of multivalued attribute when this error occurred.</p> <p>The connector may not have been created correctly. There is a problem in the order that has been set for the provisioning fields. Try creating another connector with fewer attributes for the relevant multivalued field. Try direct provisioning. After each successful round of provisioning, try adding fields one by one by performing the Manage Generic Technology Connector procedure. The point at which you start facing this issue again identifies the field that is not in the correct order.</p>
<p>Response code: SPML Provisioning ID Missing</p> <p>Response Description: The unique ID is missing.</p>	<p>You are trying to run an operation on a created user. However, the Create User operation itself may not have run successfully and the unique ID (psoID) that was expected as the response was not received. Therefore, the provisioned instance data was not updated in Oracle Identity Manager. Check why this operation failed.</p>
<p>Response code: SPML Provisioning Target ID Missing</p> <p>Response Description: The unique Target ID is missing.</p>	<p>Check the provider parameters that have been entered. TargetID may be missing.</p>

Table 22–2 (Cont.) Common Errors Encountered During Provisioning

Problem Description	Solution
<p>Response code: OIM API Error</p> <p>Response Description: An error was encountered in the Oracle Identity Manager API layer.</p>	<p>Check if Oracle Identity Manager is operating correctly for other operations. Check the connectivity between the Oracle Identity Manager front end and the database.</p> <p>Note: This error is not related to the providers that you use.</p>
<p>Response code: OIM Process Form Not Found</p> <p>Response Description: The process form was not found in Oracle Identity Manager.</p>	<p>The generic technology connector may not have been created correctly. Try creating another connector by using the same set of configurations. Try direct provisioning.</p> <p>Note: This error is not related to the providers that you use.</p>
<p>Response code: OIM Process Form Instance Not Found</p> <p>Response Description: The process form instance was not found for the specified form during update.</p>	<p>The provisioned instance information in the Oracle Identity Manager database may have become corrupted. Try direct provisioning.</p> <p>If the problem persists, then there may be an issue with the generic technology connector. Create another generic technology connector by using the same set of configurations.</p> <p>Note: This error is not related to the providers that you use.</p>
<p>Response code: OIM Atomic Process Instance Not Found</p> <p>Response Description: The process instance found is not an atomic process.</p>	<p>The provisioned instance information in the Oracle Identity Manager database may have become corrupted. Try direct provisioning.</p> <p>If the problem persists, then there may be an issue with the generic technology connector. Create another generic technology connector by using the same set of configurations.</p> <p>Note: This error is not related to the providers that you use.</p>
<p>Response code: Column Not Found</p> <p>Response Description: An expected column was not found in the result set.</p>	<p>The generic technology connector may not have been created correctly. Try creating another connector by using the same set of configurations. Try direct provisioning.</p> <p>Note: This error is not related to the providers that you use.</p>
<p>Response code: Invalid Provider</p> <p>Response Description: The provider name specified is invalid.</p>	<p>The Provisioning Format, Transformation, or provisioning transport provider in use may not have been registered correctly. Check if you have correctly followed the steps to register the providers. If this error is displayed when a predefined provider is used, then check the directory on the Oracle Identity Manager server for the XML files of these providers. These XML files are in MDS. The locations for schema and provider definition XML files are as follows:</p> <pre>PROVIDER_DEF_XSD_LOCATION = "/db/GTC/Schema"; PROVIDER_DEF_XML_LOCATION = "/db/GTC/ProviderDefinitions";</pre>
<p>Response code: IT Resource Instance Not Found</p> <p>Response Description: The IT resource instance was not found in Oracle Identity Manager.</p>	<p>The generic technology connector may not have been created correctly. Try creating another generic technology connector by using the same set of configurations. Try direct provisioning.</p> <p>Note: This error is not related to the providers that you use.</p>

Table 22–2 (Cont.) Common Errors Encountered During Provisioning

Problem Description	Solution
<p>Response code: Version Not Found</p> <p>Response Description: The required process form version was not found in Oracle Identity Manager.</p>	<p>The generic technology connector may not have been created correctly. Try creating another connector by using the same set of configurations. If you have edited an existing connector by adding a new field to an existing data set, then that operation may have failed. Try making the same change again in the connector.</p> <p>Note: This error is not related to the providers that you use.</p>
<p>Response code: Version Not Defined</p> <p>Response Description: The required process form version was not defined in Oracle Identity Manager.</p>	<p>The generic technology connector may not have been created correctly. Try creating another connector by using the same set of configurations. If you have edited an existing connector by adding a new field to an existing data set, then that operation may have failed. Try making the same change again in the connector.</p> <p>Note: This error is not related to the providers that you use.</p>
<p>Response code: Web Service Not Found</p> <p>Response Description: The Web service was not found on the target server. Check the service name and IP address.</p>	<p>Check the service name and IP address provided in the Web service URL. If these are correct, then check if the Web service is running.</p>
<p>Response code: Web Service Connection Refused</p> <p>Response Description: The Web service connection could not be established. Check that the server is running and the specified port is correct.</p>	<p>Check if the Web service is running.</p>
<p>Response code: Web Service No Such Method</p> <p>Response Description: The Web service method could not be started. Check the operation name and parameters.</p>	<p>Check the operation name and parameters.</p>
<p>Response code: Web Service Null Parameter Value</p> <p>Response Description: The parameter value passed to the Web service is null.</p>	<p>Check if the provisioning process ran correctly. The provisioning format provider may not have run correctly and, therefore, may have generated NULL output.</p>
<p>Response code: Web Service HTTP Library Missing</p> <p>Response Description: The Web service HTTP library is not included in the classpath.</p>	<p>There is a problem with the Oracle Identity Manager installation. Contact Oracle Support and send them information about this problem and the response code and description displayed. In addition, send the relevant logs generated after running Oracle Identity Manager with logging set to the DEBUG level.</p>
<p>Response code: Web Service Null Result Value</p> <p>Response Description: The Web service result value is null.</p>	<p>Check if the Web service is running correctly. At present, it is generating NULL output as the response to the Oracle Identity Manager provisioning request.</p>

Table 22–2 (Cont.) Common Errors Encountered During Provisioning

Problem Description	Solution
<p>Response code: Web Service Invocation Issue</p> <p>Response Description: An error was encountered while invoking the Web service.</p>	Check the credentials of the Web service.
<p>Response code: Web Service Target URL Missing</p> <p>Response Description: The Web service target URL required to invoke the Web service is missing.</p>	Check the values of the provider parameters. The Web service URL may be missing. Modify the generic technology connector and provide this value again.
<p>Response code: Web Service Target Method Name Missing</p> <p>Response Description: The Web service target method name required to invoke the Web service is missing.</p>	Check the values of the provider parameters. The Web service operation name may be missing. Modify the generic technology connector and provide this value again.
<p>Response code: Web Service Response XML Parsing Error</p> <p>Response Description: An error was encountered during XML parsing of the Web service response.</p>	Check if the Web service is running correctly. It is generating an SPML response that does not conform to the format specified for the Web service provider.
<p>Response code: Web Service Response ID Error</p> <p>Response Description: Either a unique ID is not getting generated from the Web service, or its value could not be parsed because of an incorrect attribute name in the response XML file.</p>	Check if the Web service is running correctly. For the Create User operation, it is generating an SPML response that does not conform to the specified format. In addition, it is not returning the <code>psoid</code> created in the target system. The provider specification for the Web Service provider expects the return of the <code>psoid</code> field.
<p>Response code: Web Service Protocol Connection Error</p> <p>Response Description: An error was encountered in the Oracle-SOAP HTTP connection.</p>	Check the service name and IP address provided in the Web service URL. If these are correct, then check if the Web service is running. Check the operation name and parameters.
<p>Response code: Web Service Protocol Processing Error</p> <p>Response Description: An error was encountered while calling the Oracle-SOAP API.</p>	Check the service name and IP address provided in the Web service URL. If these are correct, then check if the Web service is running. Check the operation name and parameters.

Table 22–2 (Cont.) Common Errors Encountered During Provisioning

Problem Description	Solution
<p>Response Code: Unable to parse the date</p> <p>Response Description: Error encountered while parsing the date.</p>	<p>The value specified for the Target Date Format parameter is not correct. For information about the date formats that you can specify, see the following Web page: http://java.sun.com/docs/books/tutorial/i18n/format/simpleDateFormat.html#datepattern</p>
<p>Response Code: Data Access Error</p> <p>Response Description: A data access error occurred while executing the query or loading the result set.</p>	<p>Check if Oracle Identity Manager is operating correctly for other operations. Check the connectivity between the Oracle Identity Manager front end and the database.</p> <p>Note: This error is not related to the providers that you use.</p>
<p>Response Code: SSL Handshake Did Not Happen</p> <p>Response Description: An SSL handshake did not happen during the secure communication with the target Web service.</p>	<p>Check if the SEcure Sockets Layer (SSL) configuration between Oracle Identity Manager and the target system has been correctly completed. If required, perform the procedure again.</p>
<p>Response Code: Error in Initialization of SSL-Related Properties</p> <p>Response Description: An error was encountered during the initialization of SSL-related properties. The relevant values are read from the "RMSecurity" element in the oim-config.xml file in the MDS.</p>	<p>Check the configuration entries corresponding to the <code>RMSecurity</code> element of the <code>oim-config.xml</code> file.</p>
<p>Response Code: Invalid Web Service Keystore or password</p> <p>Response Description: An invalid keystore name or password was encountered in the oim-config.xml file in the MDS. Check the configuration entries corresponding to the "RMSecurity" element.</p>	<p>Check the configuration entries corresponding to the <code>RMSecurity</code> element of the <code>oim-config.xml</code> file.</p>
<p>Response Code: Error Encountered During Web Service Keystore Initialization</p> <p>Response Description: Keystore initialization failed. Credentials of the keystore are mentioned in the oim-config.xml file under the "RMSecurity" element.</p>	<p>Check the configuration entries corresponding to the <code>RMSecurity</code> element of the <code>oim-config.xml</code> file.</p>

Table 22–2 (Cont.) Common Errors Encountered During Provisioning

Problem Description	Solution
<p>Response Code: Invalid ID</p> <p>Response Description: An invalid ID is present in the input SPML request.</p>	Check the value specified for the <code>Target ID</code> parameter.
<p>Response Code: Object already exists</p> <p>Response Description: This object already exists in the target system.</p>	Check if the object that you are trying to create already exists on the target system.
<p>Response Code: Operation Not Supported</p> <p>Response Description: The requested provisioning operation is not supported.</p>	Check if the target system supports the requested provisioning operation. For information about the types of SPML provisioning operations that can be performed by using the SPML provisioning format provider, see the "SPML provisioning format provider" section.
<p>Response Code: Invalid ID Type in Input SPML Request</p> <p>Response Description: An invalid ID type is present in the input SPML request.</p>	Check the sample SPML request corresponding to the type of request that was sent, and determine if the target system supports all the ID values that were included in the request.
<p>Response Code: ID in Input SPML Request Does Not Exist in the Target System</p> <p>Response Description: The ID in the input SPML request does not exist in the target system.</p>	Ensure that the <code>psoid</code> value that was sent in the request exists in the target system.
<p>Response Code: Requested Execution Mode Not Supported</p> <p>Response Description: The requested execution mode is not supported.</p>	Ensure that the target system supports the execution of requests in synchronous mode.
<p>Response Code: Invalid Container</p> <p>Response Description: The object cannot be added to the specified container. Refer to the log file for more information. Check the value of the "errorMessage" element in the SPML response.</p>	Check if a container corresponding to the container ID specified in the request exists on the target system.

Table 22–2 (Cont.) Common Errors Encountered During Provisioning

Problem Description	Solution
<p>Response Code: Nonstandard SPML Error</p> <p>Response Description: A target-specific error was encountered. Refer to the log file for more information. Check the value of the "errorMessage" element in the SPML response.</p>	<p>Check the value of the <code>errorMessage</code> element in the SPML response. This element contains the target system error message that was generated when the error was encountered.</p>
<p>Response Code: SPML Response Is for Asynchronous Mode</p> <p>Response Description: The SPML response is for asynchronous mode, which is not supported for this release.</p>	<p>Ensure that the target system sends responses corresponding to the synchronous mode of request execution.</p>
<p>Response Code: Error Encountered While Parsing Constituent Elements of Web Service URL</p> <p>Response Description: An error was encountered while parsing the constituent elements of the Web service URL. Check if the specified URL contains the protocol, host name, port, and the endpoint. Oracle recommends copying the URL from the relevant WSDL file while specifying provider parameter values during connector creation.</p>	<p>Check if the specified URL contains the protocol, host name, port, and endpoint. Oracle recommends copying the URL from the relevant WSDL file while specifying provider parameter values during connector creation.</p>
<p>Response Code: SPML Response failed V2 schema validation</p> <p>Response Description: SPML Response received is not compliant with the SPML V2 standard specifications.</p>	<p>Ensure that the SPML response returned by the target system conforms to the SPML V2 standard specification.</p>

Part V

Requests and Approval Processes

This part contains chapters describing how to configure requests and SOA composites.

It contains the following chapters:

- [Chapter 23, "Configuring Requests"](#)
- [Chapter 24, "Understanding Approval Process Development in Oracle SOA Suite"](#)
- [Chapter 25, "Developing SOA Composites"](#)
- [Chapter 26, "Using Oracle Identity Manager APIs in SOA Composites"](#)

Configuring Requests

This chapter describes how to configure a custom request workflow to cater to the specific requirements in your organization. Configuring a custom request workflow involves the following steps:

- [Step 1: Creating a Request Dataset for the Resources](#)
- [Step 2: Uploading Request Datasets into MDS](#)
- [Step 3: Creating SOA Composites Required for Approval](#)
- [Step 4: Registering the SOA Composites in Oracle Identity Manager](#)
- [Step 5: Defining Request Approvals](#)
- [Step 6: Creating Request Templates](#)

The following section describes how to extend the request management operations by using plug-in points:

- [Extending Request Management Operations](#)

23.1 Step 1: Creating a Request Dataset for the Resources

Request dataset is an XML definition file that dictates what data needs to be collected during various phases of the request lifecycle. In the request dataset, you can define what attributes need to be submitted by the requester and approver, whether or not an attribute is mandatory, and how UI should render the attribute to the user. Every attribute defined as a part of the dataset is associated with a set of properties that define the behavior of the attributes. Request dataset also allows you to define additional attributes, which exist only in the context of the request.

Every request is raised by using a request template. Each request template is associated with a request type. Each request type is associated with a request model. There is a one-to-one correspondence between request model and request type. A request model is a specification or configuration that instructs the request engine to work in a specific way for a particular request type. Request models are broadly associated with three types of entities: user, resource, and role. All request models are shipped with Oracle Identity Manager and cannot be configured.

The request model associates the appropriate request dataset when a request is raised for a specific request type. For example, generic Provision Resource request model deals with request for provisioning of any resource objects. Every time a resource is defined, if that resource is expected to be provisioned through a request, then a new dataset can be created with the attributes that need to be collected during request lifecycle. The request datasets that are associated with a nongeneric entity, which is user, has predefined or default request datasets.

Request models that are associated with generic entities does not have default request datasets. For example, the Provision Resource request model is associated with a generic entity, which is resource. For the request models that are associated with non-generic entities, such as user, a default request dataset is available because the user entity has fixed set of default attributes.

[Table 23–1](#) lists the request models and the associated default request dataset file names that are shipped with Oracle Identity Manager.

Table 23–1 Default Request Datasets Shipped with Oracle Identity Manager

Request Model	Default Dataset File Name	Entity
Create User	CreateUserDataSet.xml	User
Delete User	DeleteUserDataset.xml	User
Enable User	EnableUserDataset.xml	User
Disable User	DisableUserDataset.xml	User
Modify User Profile	ModifyUserDataset.xml	User
Self-Register User	SelfCreateUserDataset.xml	User
Modify Self Profile	ModifyUserDataset.xml	User
Create Role	CreateRoleDataSet.xml	Role
Modify Role	ModifyRoleDataSet.xml	Role
Delete Role	DeleteRoleDataSet.xml.	Role
Assign Roles	AssignRolesDataset.xml	Role
Self Assign Roles	AssignRolesDataset.xml	Role
Self Remove Roles	RemoveRolesDataset.xml	Role
Remove from Roles	RemoveRolesDataset.xml	Role
Provision Resource	No request dataset	Resource
Self-Request Resource	No request dataset	Resource
Enable Provisioned Resource	No request dataset	Resource
Modify Provisioned Resource	No request dataset	Resource
Self Modify Provisioned Resource	No request dataset	Resource
Disable Provisioned Resource	No request dataset	Resource
De-provision Resource	No request dataset	Resource
Self De-Provision Resource	No request dataset	Resource

Note:

- It is not mandatory to have a request dataset for each request model. For example, a request dataset is not required for the De-provision Resource request model because there is no specific data to be collected as a part of request submission or approval. But for the Provision Resource request model, if it involves collection of resource-specific data as a part of request submission or approval, then there must be a dataset defined for that model, which dictates what and how the data is to be collected.
- Default request datasets can be customized or configured, such as adding new attributes. See "[User Modifiable Metadata Files](#)" on page 33-4 for information about the location of the default request datasets in MDS.

This section describes the request datasets in the following sections:

- [Elements and Properties](#)
- [Sample Request Dataset](#)
- [Child Data](#)
- [Common Request Dataset](#)
- [Configuring Localized Values for Request Datasets](#)

23.1.1 Elements and Properties

Request dataset is defined by using the following elements and their associated attributes:

- [The request-data-set Element](#)
- [The DataSetValidator Element](#)
- [The AttributeReference Element](#)
- [The Attribute Element](#)

23.1.1.1 The request-data-set Element

The request-data-set element is the root element of the request dataset with the following mandatory attributes:

- **name:** The name of the dataset, such as CreateUserDataSet
- **entity:** The underlying entity, such as user, with which the dataset is associated
- **operation:** The operation associated with the dataset, such as CREATE

The following example shows the request-data-set element:

```
<request-data-set xmlns="http://www.oracle.com/schema/oim/request"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.oracle.com/schema/oim/request"
name="CreateUserDataSet" entity="User" operation="CREATE">
```

This root element is shown without any child elements.

To create request datasets for the resource entity, refer to [Table 23-2](#) that lists the request dataset name format and operation for each request type.

Table 23–2 Request Datasets for Resource Entity

Request Type	Request Dataset Name Format	Operation
Provision Resource	ProvisionResource\${ENTITY-NAME}	PROVISION
Self-Request Resource	ProvisionResource\${ENTITY-NAME}	PROVISION
Modify Provisioned Resource	ModifyResource\${ENTITY-NAME}	MODIFYRESOURCE
Self Modify Provisioned Resource	ModifyResource\${ENTITY-NAME}	MODIFYRESOURCE

The following is a sample dataset tag for the Provision Resource dataset for the E-Business RO resource:

```
<request-data-set xmlns="http://www.oracle.com/schema/oim/request"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.oracle.com/schema/oim/request"
name="ProvisionResourceE-Business RO" entity="E-Business RO"
operation="PROVISION">
```

Here:

- For name, the ProvisionResource\${ENTITY-NAME} format specified in [Table 23–2](#) is replaced by the resource name E-Business RO. Replace \${ENTITY_NAME} with the resource name.
- Specify operation as shown in [Table 23–2](#).
- Values for the properties are:
 - name: ProvisionResourceE-Business RO
 - entity: E-Business RO
 - operation: PROVISION

23.1.1.2 The DataSetValidator Element

The `DataSetValidator` element is an optional element in request dataset. It is one of the child elements of the `request-data-set` element that describes the user-defined plug-in details meant for validating the dataset attribute values. The request engine runs the implemented plug-in to validate request data during submission. If the validation is successful, then the request gets created. Otherwise, the request is not created. You must implement the plug-in logic. Each dataset can have only one `DataSetValidator` defined. Following are the attributes:

- **name:** This attribute specifies a logical name of the `DataSetValidator` plug-in.
- **classname:** This attribute specifies the fully qualified name of the implemented plug-in class.

The following example shows the `DataSetValidator` element:

```
<DataSetValidator name="CreateUserDataValidator"
classname="oracle.iam.requestactions.plugins.datavalidator.CreateUserDataValidator" />
```

In each default request dataset mentioned in [Table 23–1, "Default Request Datasets Shipped with Oracle Identity Manager"](#), the default `DataSetValidator` element is defined and the corresponding implementation is provided by default. The classname in the `DataSetValidators` in these default request datasets can be changed to a customized class to have customized validation.

See Also: ["Validating Request Data"](#) on page 23-31 for information about custom validation of request data after submission.

23.1.1.3 The AttributeReference Element

This child element is used to define the entity attributes at request end that take part in data-flow between request dataset and underlying entity attribute or process-form fields. All the data corresponding to AttributeReference are collected as request data at various stages of the request life cycle based on the configuration.

Multiple AttributeReference elements can be provided in a dataset, one for each attribute.

Mandatory Properties

The following mandatory properties are used to configure AttributeReference:

- **name:** This is the unique name to identify the element. The request refers to an AttributeReference by using this name. Its value is of type String.
- **attr-ref:** This is the mapping property between the data set value and process form field or underlying entity attribute. For example, the definition `<AttributeReference name="Organization" attr-ref="act_key">` in the Create User dataset specifies that the request collects the data as Organization corresponding to the new user being created and gets populated in the act_key data field of the user entity.

Similarly, with the AttributeReference name="Domain" attr-ref="domain" definition associated with a Provision Email resource dataset, the request collects the data as Domain corresponding to the resource being provisioned, and the data is populated to the domain field of the process form. Therefore, for provision resource models, attr-ref attribute value must be the same as the value of the field label (SDC_LABEL) of the process form. For other models, the attr-ref attribute value must be the entity attribute of the underlying entity defined by the feature. Its value is of type String.

DataFlows is required to map the request dataset attributes to the underlying entity attributes. For example, in some request models, such as provision resource, you might need to define a data flow mapping between the request dataset and the process form data fields. The data flow mapping can be achieved by specifying the following in the dataset:

```
<AttributeReference name="ATTRIBUTE_NAME"
attr-ref="DATA_FIELD_NAME_IN_PROCESS_FORM" available-in-bulk="false" />
```

For user-based and role-based request datasets, attr-ref value is the attribute name specified in user and role entity definitions. For resource-based request models, attr-ref value must be the label name of the process form attribute. But for a child form, attr-ref value must be the child table name.

- **type:** This property specifies the data type of the value. For example, type="String" for the First Name attribute specifies that the First Name field in the Create User request UI accepts String type input. The supported data types are:
 - Byte
 - Double
 - Integer
 - String

- Short
- Long
- Date
- Boolean
- ByteArray
- Clob

Note: Out of all the supported types, only Clob data is not displayed in the request management UI.

Attributes of type Clob must not be made as approver-only.

- **widget:** This property is used to specify how the data attribute is to be displayed in the UI at the time of data collection. The value of this property is of type String. The following widgets are supported:
 - **text:** Specifies a text box that allows the user to enter text in a single line. For example, `widget="text"` for the First Name attribute specifies that the First Name field in the Create User UI is a text box.
 - **date:** Specifies a date and time type field. For example, `widget="date"` for the Start Date attribute specifies that the Start Date field in the Create User UI accepts a date as input.
 - **entity:** Specifies an entity type field. When you specify `widget="ENTITY"`, you must specify a value for the `entity-type` property, such as `entity-type="ORGANIZATION"`. This means that the Organization field in the Create User request UI provides the organization lookup from which you can search and select an organization that is present in Oracle Identity Manager. When `widget=ENTITY`, then the value of `entity-type` must be `USER`, `ORGANIZATION`, or `ROLE`.

Note: You must specify a value for the `attr-ref` element, which is the attribute name for organization in the user entity definition. For example, `attr-ref="act_key"` in the User.xml, which contains the entity definition of the user entity.

- **textarea:** Specifies a large text field for entering multiple lines of text.
- **dropdown:** Specifies a List of Values (LOV). When you specify `widget="dropdown"`, you must specify values for a list of `lookupValues encoded-value` and `lookupValues decoded-value` elements, such as `lookupValues encoded-value="End User" decoded-value="Identity Only"` and `lookupValues encoded-value="End-User Administrator" decoded-value="End-User Administrator"`. This means that the User Type field in the Create User request UI is displayed as an LOV from which the user types **Identity Only** or **End-User Administrator** can be selected. However, if the `lookup-code` property is defined, as described later in this chapter, then you do not have to specify `lookupValues`.

The list of values can come from either of the following:

- Static list of `lookupValues` specified in the dataset itself. For example:


```
lookupValues encoded-value="End User" decoded-value="Identity Only" and
lookupValues encoded-value="End-User Administrator" decoded-value="End-User
Administrator"
```

- List of lookup values based on the lookup-code property defined, as described later in this chapter.

- **radio:** Specifies a radio button.
- **checkbox:** Specifies a checkbox field. This widget can be associated with attribute references only with Boolean type.
- **lookup:** Specifies a lookup field that allows you to select a value from a large number of values. If this is used, then the lookup-code property must be specified.
- **lookup-query:** Specifies a search and select widget that is associated with the lookupQuery element.
- **itresource-lookup:** Specifies a search and select widget that is associated with an IT resource and shows available IT resource instances. For more information about this widget, see "[Provision E-Business Resource Dataset](#)" on page 23-16.
- **length:** This attribute specifies the length of the data value. For example, `length="80"` for the First Name attribute specifies that the First Name field in the Create User request UI accepts an input of maximum 80 characters. Its value is of type positive integer.
- **available-in-bulk:** Its value is of type Boolean. This property indicates whether or not the attribute reference is to be displayed during bulk request creation. See "[Bulk Requests and Child Requests](#)" in the *Oracle Fusion Middleware User's Guide for Oracle Identity Manager* for information about bulk requests.

You can always hide the fields related to single user context, such as first name, user ID, and password, from the request dataset by marking these fields as `available-in-bulk="false"` in a bulk request scenario. This is because the bulk request is applicable for multiple users and the single user fields does not make sense to be displayed on the request UI. For provisioning requests, these single user fields can be populated by the prepopulation adapters filling the process forms directly. If an attribute is specified as `available-in-bulk="false"`, then that attribute cannot be made mandatory. If you create request datasets by using prepopulation adapters, then you cannot have mandatory constraints assigned to any of these single user fields such as first name, user ID, and password. If you want to assign mandatory constraints to single user fields for usability when a single user is requesting for the resource by using self service, then use the `PrePopulationAdapter` element in the request dataset for populating user specific data. See "[The PrePopulationAdapter Element](#)" on page 23-9 for information about using the `PrePopulationAdapter` element in request datasets.

Optional Properties

The following optional attributes can be used to configure `AttributeReference`:

- **required:** This is a flag property to indicate that data value must be supplied at the time of request submission. Value is of type Boolean. If this property is not specified, then the default value taken is false.

Any attribute reference for which the corresponding `attr-ref` field is mandatory in the underlying entity, must be specified as `required="true"`. For example, `Organization` is a mandatory attribute in the user entity. Therefore, the

corresponding attribute reference in SelfCreateUserDataset.xml or CreateUserDataSet.xml is specified as `required="true"` reflecting that this field is mandatory in underlying entity as well.

- **Masked:** This is a flag property used to specify if the data value is masked. If the value is set to masked, the request engine always displays it as asterisks. Value is of type Boolean. If this property is not specified, then the default value taken is false.
- **approver-only:** This is a flag property used to specify that data value has to be supplied, edited by approver. By setting this flag, requester is never allowed to supply the corresponding data value. Value is of type Boolean. If this property is not specified, then the default value taken is "false".

If an attribute specified as `required="true"` and `approver-only="true"`, then approver has to provide the value for this attribute before approving the request.

The approval process does not support adding child table data, and therefore, only one value can be provided for a child table (multivalued) field during the approval process.

- **entity-type:** This property is used to associate an entity from which supported data value is derived for selection from the request UI. For example:

```
<AttributeReference name="Organization" attr-ref="act_key"
available-in-bulk="false" type="Long" length="20" widget="ENTITY"
required="true" entity-type="ORGANIZATION"/>
```

With this definition, UI displays a lookup widget by using which user can search and select an organization in Oracle Identity Manager.

If entity-type property is defined, then the widget must be defined as ENTITY, and UI displays a lookup widget by using which user can search and select an entity.

- **lookup-code:** This property is used to associate available LKU/LKV values as supported data based on a defined lookup-code. Example:

```
<AttributeReference name="Responsibility Name" attr-ref="Name" type="String"
length="30" widget="lookup" required="false" available-in-bulk="true"
lookup-code="Oracle.Responsibility.Name"/>
```

This definition renders all the encoded/decoded values for the lookup-code Oracle.Responsibility.Name.

If the lookup-code property is defined, then widget can be defined as lookup, and UI displays a lookup widget by using which user can search and select a lookup value.

If lookup-code is defined, the widget can be dropdown as well. For example:

```
<AttributeReference name="Role" attr-ref="Role" available-in-bulk="false"
type="String" length="20" widget="dropdown" lookup-code="Lookup.Users.Role"
required="true"/>
```

Here, the User types will be displayed as a dropdown, and user can select any of the values.

If the lookup code is associated with limited number of values, then you can use dropdown. But if the lookup code is associated with large number of values, then use the lookup widget, which allows search and selection of value.

- **itresource-type:** This property is used to associate available itresource instance for a defined itresource-type. Example:

```
<AttributeReference name="Server" attr-ref="Server Name" type="String"
widget="itresource-lookup" required="true" itresource-type="EBIZServer"
available-in-bulk="true" length="20"/>
```

This definition renders all IT resource instances for the EBIZServer itresource-type.

If the itresource-type property is defined, then widget must be defined as itresource-lookup, and UI displays a lookup widget by using which user can search and select an itresource instance.

- **primary:** This is a flag property used to specify if the dataset attribute can have more than one value. This flag can be set for a dataset attribute only in the context of child table. See "[Child Data](#)" on page 23-15 for more information about the primary property.
- **mls:** This is a flag property used to specify if the dataset attribute is of type Multi-Language Support (MLS). Value is of type Boolean. If this property is not specified, then the default value taken is false.
- **entitlement:** This is a flag property used to specify if the dataset attribute is of type entitlement. Value is of type Boolean. If this property is not specified, then the default value taken is false.
- **hidden:** This is a flag property used to specify if the data value is hidden from the approver. This data value is not visible only to the approver but data can be collected from the requester at the time of submitting the request or by other means. Value is of type Boolean. If this property is not specified, then the default value taken is false.

23.1.1.3.1 The PrePopulationAdapter Element This child element is used to define an associated Oracle Identity Manager plug-in class that helps in generating data values for the corresponding attribute. Each attribute can have a maximum of one PrePopulationAdapter element associated for an AttributeReference definition. An attribute value is prepopulated during request creation from UI with the value returned by the prepopulation adapter plug-in. The attributes are:

- **name:** This attribute is used to specify a logical name of the adapter.
- **classname:** This attribute is used to specify the fully qualified classname of the plugin class.

Note: Even if a request dataset attribute is configured with a PrePopulationAdapter, its values can be restricted in a request template. As a result, prepopulation does not happen and the values restricted in the template are displayed in the request creation UI.

The following example shows how to associate PrePopulateAdapter for an AttributeReference:

```
<AttributeReference name="Organization" attr-ref="act_key"
available-in-bulk="false" type="Long" length="20" widget="ENTITY" required="true"
entity-type="ORGANIZATION"/>
<PrePopulationAdapter name="prepopulateOrg"
classname="my.sample.package.SamplePrePopulateOrg" />
</AttributeReference>
```

The `my.sample.package.SamplePrePopulateOrg` class must be registered as a plug-in with Oracle Identity Manager.

[Example 23–1](#) shows a sample data set for provisioning the Active Directory (AD) resource, in which prepopulation is used:

Example 23–1 Provisioning AD Resource

```
<?xml version="1.0" encoding="UTF-8"?>
<request-data-set xmlns="http://www.oracle.com/schema/oim/request"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.oracle.com/schema/oim/request"
name="ProvisionResourceAD" entity="AD" operation="PROVISION">
<AttributeReference name="Domain" attr-ref="domain" available-in-bulk="true"
type="String" length="20" widget="text">
<PrePopulationAdapter classname="oracle.iam.request.DomainPrepopulateAdapter"/>
</AttributeReference>
<AttributeReference name="Login" attr-ref="login" available-in-bulk="true"
type="String" length="20" widget="text"/>
<AttributeReference name="Organization" attr-ref="organization"
available-in-bulk="true" type="String" length="20" widget="text" required="true">
<PrePopulationAdapter name="org"
classname="oracle.iam.request.OrgPrepopulateAdapter"/>
</AttributeReference>
<AttributeReference name="EmployeeType" attr-ref="EmployeeType"
available-in-bulk="true" approver-only="true" type="String" length="20"
widget="text"
required="true">
</AttributeReference>
<AttributeReference name="Role" attr-ref="role"
available-in-bulk="true" type="String" length="20" widget="text">
<AttributeReference name="RoleName" attr-ref="role"
available-in-bulk="true" type="String" length="20" widget="text"
entitlement="true">
</AttributeReference>
<AttributeReference name="Description" attr-ref="description"
available-in-bulk="true" type="String" length="20" widget="text">
</AttributeReference>
</AttributeReference>
</request-data-set>
```

Here, Role is a child form with child attributes RoleName and Description.

The dataset for provisioning AD resource shows that the Organization attribute has a prepopulation adapter associated with it. The organization attribute value will be prepopulated during request creation with the value returned by the prepopulation adapter plug-in.

The value returned by the prepopulate method of the plug-in must be of type corresponding to the type configured in the request dataset. For example, for the Organization attribute in [Example 23–1](#), the prepopulate method of `OrgPrepopulateAdapter` returns a value of type `java.lang.String` because the type for Organization attribute is configured as String in the dataset.

23.1.1.3.2 The lookupValues Element The `lookupValues` element, which is a child element of `AttributeReference`, is used to define a set of allowable data values for an entity attribute associated with the `AttributeReference` definition. The attributes of this element are:

- **decoded-value:** This is the data value that is shown to the requester during request creation from UI.
- **encoded-value:** This is the actual data value stored in the request data, and is used for the dataflow. Based on the selected decoded value from the request creation UI, the corresponding encoding value is stored in the request data.

The following sample code snippet shows the AttributeReference that uses lookupValues for the User Type entity attribute:

```
<AttributeReference name="User Type" attr-ref="Xellerate Type"
available-in-bulk="false" type="String" length="30" widget="dropdown"
required="true">
<lookupValues encoded-value="End-User Admin" decoded-value="End-User
Administrator"/>
<lookupValues encoded-value="Identity" decoded-value="Identity"/>
<lookupValues encoded-value="End-User" decoded-value="End-User"/>
</AttributeReference>
```

The User Type attribute can have one of the three possible values: End-User Admin, Identity Only, and End-User. But the corresponding decoded values are displayed in the dropdown list to the requester at the time of data collection: End-User Administrator, Identity, and End-User respectively. The encoded value is populated into the mapped entity attribute fields as a part of data flow.

23.1.1.3.3 The lookupQuery Element This child element of AttributeReference used to derive a set of data value dynamically based on a SQL. Request UI shows all the values based on defined lookupQuery in a lookup widget.

```
<AttributeReference name="adminlogin" attr-ref="adminlogin" type="String"
length="20" widget="lookup-query" available-in-bulk="true">
<lookupQuery lookup-query="SELECT USR_KEY as UKEY, USR_LOGIN as ULOGIN
FROM TEMP_USR where USR_TYPE='$Form Data.admintype' " display-field="ULOGIN"
save-field="UKEY"/>
</AttributeReference>
```

In this example, user key and user login is queried from a table temp_usr based on a SQL query. Following are the properties of this element:

- **lookup-query:** This property value is a generic SQL query supported by the Oracle Identity Manager database. This query can be dependent on another attribute reference of the same dataset. In the example, there is a reference to '\$Form Data.admintype'. This means that the attribute reference 'adminlogin' depends on the attribute reference 'admintype'. The value provided by requester to attribute 'admintype' is used for fetching values for the attribute 'adminlogin' in lookup.
- **display-field:** This property value is one of the alias name from the selected column that needs to be shown to the end-user in the UI attribute, after user selects a value from lookup-widget.
- **save-field:** This property value is one of the alias name from the selected column that needs to be saved internally to the system, after user selects a value from lookup-widget.

The display-field and save-field can be same for an UI attribute.

Note: In the lookup query, it is mandatory to have aliases for the columns that are used as save-field and display-field.

23.1.1.4 The Attribute Element

As part of request creation, collection of data that does not refer to the underlying entity might be required. The Attribute element can be used to achieve this. An Attribute defined in the request dataset does not require mapping to the underlying entity. These attributes do not require any mapping, and therefore, can be defined in the following way:

```
<Attribute name="ATTRIBUTE_NAME" length="10" type="integer" widget="text"
available-in-bulk="false" />
```

The attributes are shown in the request details. These can be viewed by the approver so that these can be used for approval decisions.

The Attribute element is similar to the AttributeReference element with a difference. The Attribute element data values are available only in context of request and cannot take part in dataflow. All other properties available with AttributeReference are also available with attribute, except attr-ref attribute.

23.1.2 Sample Request Dataset

The data that needs to be collected by the request is defined in the request dataset XML file. [Example 23-2](#) is the sample XML code for the Create User dataset:

Example 23-2 Create User Dataset

```
<?xml version="1.0" encoding="UTF-8"?>
<request-data-set xmlns="http://www.oracle.com/schema/oim/request"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.oracle.com/schema/oim/request" name="CreateUserDataSet"
entity="User" operation="CREATE">
  <DataSetValidator name="CreateUserDataValidator"
classname="oracle.iam.requestactions.plugins.datavalidator.CreateUserDataValidator"/>
  <AttributeReference name="First Name" attr-ref="First Name" available-in-bulk="false"
type="String" length="80"
  widget="text" required="false" mls="false"/>
  <AttributeReference name="Middle Name" attr-ref="Middle Name" available-in-bulk="false"
type="String" length="80"
  widget="text" required="false" mls="false"/>
  <AttributeReference name="Last Name" attr-ref="Last Name" available-in-bulk="false"
type="String" length="80"
  widget="text" required="true" mls="false"/>
  <AttributeReference name="User Login" attr-ref="User Login" available-in-bulk="false"
type="String" length="256"
  widget="text" required="false"/>
  <AttributeReference name="Password" attr-ref="usr_password" available-in-bulk="false"
type="String" length="128"
  widget="text" required="false" masked="true"/>
  <AttributeReference name="Password Generated" attr-ref="Password Generated"
available-in-bulk="false" type="Clob" length="1"
  widget="text" required="false"/>
  <AttributeReference name="Organization" attr-ref="act_key" available-in-bulk="false"
type="Long" length="256"
  widget="ENTITY" required="true" entity-type="ORGANIZATION"/>
  <AttributeReference name="User Type" attr-ref="Xellerate Type" available-in-bulk="false"
type="Boolean" length="30"
  widget="checkbox" required="false"/>
  <AttributeReference name="Role" attr-ref="Role" available-in-bulk="false" type="String"
length="255"
  widget="dropdown" lookup-code="Lookup.Users.Role" required="true"/>
```

```

    <AttributeReference name="User Manager" attr-ref="usr_manager_key" available-in-bulk="false"
type="Long" length="382"
        widget="ENTITY" required="false" entity-type="USER"/>
    <AttributeReference name="Country" attr-ref="Country" available-in-bulk="false" type="String"
length="100"
        widget="text" required="false"/>
    <AttributeReference name="Common Name" attr-ref="Common Name" available-in-bulk="false"
type="String" length="240"
        widget="text" required="false" mls="false"/>
    <AttributeReference name="Display Name" attr-ref="Display Name" available-in-bulk="false"
type="String" length="382"
        widget="text" required="false" mls="true"/>
    <AttributeReference name="Department Number" attr-ref="Department Number"
available-in-bulk="false" type="String" length="80"
        widget="text" required="false"/>
    <AttributeReference name="Description" attr-ref="Description" available-in-bulk="false"
type="String" length="2000"
        widget="text" required="false"/>
    <AttributeReference name="Employee Number" attr-ref="Employee Number" available-in-bulk="false"
type="String" length="80"
        widget="text" required="false"/>
    <AttributeReference name="Fax" attr-ref="Fax" available-in-bulk="false" type="String"
length="20"
        widget="text" required="false"/>
    <AttributeReference name="Generation Qualifier" attr-ref="Generation Qualifier"
available-in-bulk="false" type="String" length="20"
        widget="text" required="false" mls="false"/>
    <AttributeReference name="Home Phone" attr-ref="Home Phone" available-in-bulk="false"
type="String" length="20"
        widget="text" required="false"/>
    <AttributeReference name="Hire Date" attr-ref="Hire Date" available-in-bulk="false" type="Date"
length="50"
        widget="date" required="false"/>
    <AttributeReference name="Home Postal Address" attr-ref="Home Postal Address"
available-in-bulk="false" type="String" length="256"
        widget="text" required="false"/>
    <AttributeReference name="Locality Name" attr-ref="Locality Name" available-in-bulk="false"
type="String" length="80"
        widget="text" required="false"/>
    <AttributeReference name="Email" attr-ref="Email" available-in-bulk="false" type="String"
length="256"
        widget="text" required="false"/>
    <AttributeReference name="Mobile" attr-ref="Mobile" available-in-bulk="false" type="String"
length="20"
        widget="text" required="false"/>
    <AttributeReference name="Pager" attr-ref="Pager" available-in-bulk="false" type="String"
length="20"
        widget="text" required="false"/>
    <AttributeReference name="Postal Address" attr-ref="Postal Address" available-in-bulk="false"
type="String" length="256"
        widget="text" required="false" mls="false"/>
    <AttributeReference name="PO Box" attr-ref="PO Box" available-in-bulk="false" type="String"
length="20"
        widget="text" required="false"/>
    <AttributeReference name="Postal Code" attr-ref="Postal Code" available-in-bulk="false"
type="String" length="30"
        widget="text" required="false"/>
    <AttributeReference name="usr_locale" attr-ref="usr_locale" available-in-bulk="false"
type="String" length="80"
        widget="text" required="false"/>

```

Step 1: Creating a Request Dataset for the Resources

```
<AttributeReference name="State" attr-ref="State" available-in-bulk="false" type="String"
length="80"
    widget="text" required="false" mls="false"/>
<AttributeReference name="Street" attr-ref="Street" available-in-bulk="false" type="String"
length="80"
    widget="text" required="false"/>
<AttributeReference name="Telephone Number" attr-ref="Telephone Number"
available-in-bulk="false" type="String" length="20"
    widget="text" required="false"/>
<AttributeReference name="Title" attr-ref="Title" available-in-bulk="false" type="String"
length="80"
    widget="text" required="false" mls="false"/>
<AttributeReference name="Initials" attr-ref="Initials" available-in-bulk="false" type="String"
length="10"
    widget="text" required="false"/>
<AttributeReference name="Start Date" attr-ref="Start Date" available-in-bulk="false"
type="Date" length="50"
    widget="date" required="false"/>
<AttributeReference name="End Date" attr-ref="End Date" available-in-bulk="false" type="Date"
length="50"
    widget="date" required="false"/>
<AttributeReference name="LDAP Organization Unit" attr-ref="LDAP Organization Unit"
available-in-bulk="false" type="String" length="80"
    widget="text" required="false" mls="false"/>
<AttributeReference name="LDAP Organization" attr-ref="LDAP Organization"
available-in-bulk="false" type="String" length="80"
    widget="text" required="false" mls="false"/>
<AttributeReference name="usr_timezone" attr-ref="usr_timezone" available-in-bulk="false"
type="String" length="100"
    widget="text" required="false" mls="false"/>

<AttributeReference name="Number Format" attr-ref="Number Format" available-in-bulk="false"
type="String" length="30"
    widget="dropdown" lookup-code="Lookup.Users.NumberFormat" required="false"
mls="false"/>
<AttributeReference name="Currency" attr-ref="Currency" available-in-bulk="false" type="String"
length="20"
    widget="dropdown" lookup-code="Lookup.Users.Currency" required="false"
mls="false"/>
<AttributeReference name="Date Format" attr-ref="Date Format" available-in-bulk="false"
type="String" length="20"
    widget="dropdown" lookup-code="Lookup.Users.DateFormat" required="false"
mls="false"/>
<AttributeReference name="Time Format" attr-ref="Time Format" available-in-bulk="false"
type="String" length="20"
    widget="dropdown" lookup-code="Lookup.Users.TimeFormat" required="false"
mls="false"/>
<AttributeReference name="Embedded Help" attr-ref="Embedded Help" available-in-bulk="false"
type="String" length="10"
    widget="dropdown" lookup-code="Lookup.Users.EmbeddedHelp" required="false"
mls="false"/>
<AttributeReference name="Font Size" attr-ref="Font Size" available-in-bulk="false"
type="String" length="10"
    widget="dropdown" lookup-code="Lookup.Users.FontSize" required="false"
mls="false"/>
<AttributeReference name="Color Contrast" attr-ref="Color Contrast" available-in-bulk="false"
type="String" length="10"
    widget="dropdown" lookup-code="Lookup.Users.ColorContrast" required="false"
mls="false"/>
```



```

    <AttributeReference name="Accessibility Mode" attr-ref="Accessibility Mode"
available-in-bulk="false" type="String" length="20"
        widget="dropdown" lookup-code="Lookup.Users.AccessibilityMode" required="false"
mls="false"/>
    <AttributeReference name="FA Language" attr-ref="FA Language" available-in-bulk="false"
type="String" length="100"
        widget="text" required="false"/>
    <AttributeReference name="FA Territory" attr-ref="FA Territory" available-in-bulk="false"
type="String" length="100"
        widget="text" required="false"/>
    <AttributeReference name="User Name Preferred Language" attr-ref="User Name Preferred Language"
available-in-bulk="true" type="String" length="20" widget="lookup-query" required="false">
<lookupQuery lookup-query="select mls_locale_code as USR_NAME_PREFERRED_LANG from mls_locale where
( locale_flag=0 OR locale_flag=1 ) order by mls_locale_code asc"
display-field="USR_NAME_PREFERRED_LANG" save-field="USR_NAME_PREFERRED_LANG"/>
    </AttributeReference>

    <Attribute name="Roles" available-in-bulk="false" type="Clob" length="2048" widget="text"
required="false"/>
    <Attribute name="Policy Name" available-in-bulk="false" type="Clob" length="1024" widget="text"
required="false"/>
    <Attribute name="RequestorID" available-in-bulk="false" type="Clob" length="1024" widget="text"
required="false"/>
    <Attribute name="FAOpData" available-in-bulk="false" type="Clob" length="4096" widget="text"
required="false" />
</request-data-set>

```

23.1.3 Child Data

You might need attributes to store multiple values or attributes that are made up of other attributes. To do so, you can configure one or more child attributes. For example, an Email ID attribute of an entity type User needs to store multiple values. Therefore, you can configure it in the request dataset in the following way:

```

<Attribute name="Email">
    <Attribute name="ID" length="20" type="string" widget="text" />
</Attribute>

```

You might also require an attribute to be composed of multiple attributes. For example, a Oracle Apps User Responsibilities attribute needs to be made up of three attributes: Responsibility Start Date, Responsibility End Date, and Responsibility Name. You can configure this attribute in the request data set in the following way:

```

<AttributeReference name="Oracle Apps User Responsibilities" attr-ref="UD_RESPONS"
type="String" length="20" widget="text" available-in-bulk="true">
    <AttributeReference name="Responsibility Start Date" attr-ref="Responsibility
Start Date" type="Date" widget="date" required="false" available-in-bulk="true"
length="100" />
    <AttributeReference name="Responsibility End Date" attr-ref="Responsibility
End Date" type="Date" widget="date" required="false" available-in-bulk="true"
length="100" />
    <AttributeReference name="Responsibility Name" attr-ref="Responsibility Name"
type="String" length="30" widget="lookup" required="false"
available-in-bulk="true" lookup-code="Oracle.Responsibility.Name"
primary="true"/>.
</AttributeReference>

```

Here, the association of Responsibility Start Date, Responsibility End Date, and Responsibility Name are maintained and the three attributes together constitute a value of the Oracle Apps User Responsibilities child attribute.

Note:

- Only one level of child attributes are supported in Oracle Identity Manager. Therefore, in the example, the Responsibility Start Date, Responsibility End Date, or Responsibility Name attributes cannot be composed of other attributes. Similarly, attribute references cannot have child attributes.
 - The values for AttributeReference name and attr-ref must be the same for child table attributes. For instance, in the Oracle Apps User Responsibilities attribute example in this section, the value for both AttributeReference name and attr-ref is Responsibility Start Date.
-
-

During request creation, child data is shown in a table and child data can added from a popup window. In this scenario, the requester might want to add multiple responsibilities with same start date and end date. You can allow the requester to select multiple responsibilities with same start date and end date by specifying the Responsibility Name as primary.

The primary property allows the requester to select multiple values to the Responsibility Name attribute in the window displayed when the requester tries to add a child row. For the Responsibility Start Date and Responsibility End Date attributes, only single value can be provided. With this multiple rows will be added to the child table one for each responsibility name selected with same value of start date and end date.

[Example 23-3](#) shows the sample XML code for a request dataset for provisioning an E-Business resource:

Example 23-3 Provision E-Business Resource Dataset

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<request-data-set
  xmlns = "http://www.oracle.com/schema/oim/request"
  xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
  operation = "PROVISION"
  entity = "eBusiness Suite User"
  name = "ProvisionResourceeBusiness Suite User"
  xsi:schemaLocation = "http://www.oracle.com/schema/oim/request">
  <AttributeReference
    itresource-type = "eBusiness Suite UM"
    available-in-bulk = "true"
    required = "true"
    length = "20"
    widget = "itresource-lookup"
    type = "Long"
    attr-ref = "EBS Server"
    name = "EBS Server"/>
  <AttributeReference
    available-in-bulk = "true"
    length = "240"
    widget = "text"
    type = "String"
```

```

        attr-ref = "Description"
        name = "Description"/>
<AttributeReference
  available-in-bulk = "false"
  length = "240"
  widget = "text"
  type = "String"
  attr-ref = "Email"
  name = "Email"/>
<AttributeReference
  available-in-bulk = "true"
  length = "80"
  widget = "text"
  type = "String"
  attr-ref = "Fax"
  name = "Fax"/>
<AttributeReference
  available-in-bulk = "false"
  length = "256"
  widget = "text"
  type = "String"
  attr-ref = "SSO User ID"
  name = "SSO User ID"/>
<AttributeReference
  available-in-bulk = "false"
  length = "30"
  widget = "text"
  type = "String"
  attr-ref = "Person ID"
  name = "Person ID"/>
<AttributeReference
  available-in-bulk = "true"
  length = "10"
  widget = "text"
  type = "String"
  attr-ref = "UD_EBS_RESP"
  name = "eBusiness Suite Responsibilities">

  <AttributeReference
    name = "Application Name"
    attr-ref = "Application Name"
    type = "String"
    length = "256"
    widget = "lookup-query"
    available-in-bulk = "true"
    required = "true">
    <lookupQuery
      lookup-query = "select lkv_encoded as Value, lkv_decoded as
Description from lkv lkv, lku lku where lkv.lku_key=lku.lku_key and
lku_type_string_key='Lookup.EBS.Application' and instr(lkv_encoded,concat('$Form
data.EBS Server', '~'))>0"
      display-field = "Description"
      save-field = "Value"/>
    </AttributeReference>
  <AttributeReference
    name = "Responsibility Name"
    attr-ref = "Responsibility Name"
    type = "String"
    length = "256"
    widget = "lookup-query"

```

```

        available-in-bulk = "true"
        required = "true"
        primary = "true">
        <lookupQuery
            lookup-query = "select lkv_encoded as Value, lkv_decoded as
Description from lkv lkv, lku lku where lkv.lku_key=lku.lku_key and
lku_type_string_key='Lookup.EBS.Responsibility' and
instr(lkv_encoded, concat('$Form data.Application Name', '~'))>0"
            display-field = "Description"
            save-field = "Value"/>
        </AttributeReference>
    </AttributeReference>
    <AttributeReference
        available-in-bulk = "true"
        length = "20"
        widget = "date"
        type = "Date"
        attr-ref = "Effective Start Date"
        name = "Effective Start Date"/>
    </AttributeReference>
    <AttributeReference
        available-in-bulk = "true"
        length = "10"
        widget = "text"
        type = "String"
        attr-ref = "UD_EBS_RLS"
        name = "eBusiness Suite User Role Grants">
        <AttributeReference
            name = "Application Name"
            attr-ref = "Application Name"
            type = "String"
            length = "256"
            widget = "lookup-query"
            available-in-bulk = "true"
            required = "true">
            <lookupQuery
                lookup-query = "select lkv_encoded as Value, lkv_decoded as
Description from lkv lkv, lku lku where lkv.lku_key=lku.lku_key and
lku_type_string_key='Lookup.EBS.Application' and instr(lkv_encoded, concat('$Form
data.EBS Server', '~'))>0"
                display-field = "Description"
                save-field = "Value"/>
            </AttributeReference>
        </AttributeReference>
        <AttributeReference
            name = "Role Name"
            attr-ref = "Role Name"
            type = "String"
            length = "256"
            widget = "lookup-query"
            available-in-bulk = "true"
            required = "true"
            primary = "true">
            <lookupQuery
                lookup-query = "select lkv_encoded as Value, lkv_decoded as
Description from lkv lkv, lku lku where lkv.lku_key=lku.lku_key and
lku_type_string_key='Lookup.EBS.UMX.Roles' and instr(lkv_encoded, concat('$Form
data.Application Name', '~'))>0"
                display-field = "Description"
                save-field = "Value"/>
            </AttributeReference>
        </AttributeReference>
    </AttributeReference>

```

```

        available-in-bulk = "true"
        length = "20"
        widget = "date"
        type = "Date"
        attr-ref = "Start Date"
        name = "Start Date"/>
    </AttributeReference>
</request-data-set>

```

In the sample XML code for provisioning an E-Business resource dataset:

- The Oracle Apps User Responsibilities attribute is defined as a parent attribute for the Responsibility Start Date, Responsibility End Date, and Responsibility Name child attributes. Users can specify one or more values the Oracle Apps User Responsibilities. In request creation UI, this is shown as a table with header "Oracle Apps User Responsibilities" and with the Responsibility Start Date, Responsibility End Date, and Responsibility Name columns.

For the parent attribute, value of attr-ref should be the child table name in the process form. In this example it is "UD_RESPONS".

The Responsibility Start Date, Responsibility End Date, and Responsibility Name attributes are the columns of the child table "UD_RESPONS".

For the child attributes, the attr-ref value must be the Field Label value in the child table of the process form.

Oracle Identity Manager allows you to define a child process form and associates it to a parent process form for a resource. The attributes in the parent form are modeled as attribute references in the request dataset. The attributes in the child form are modeled as attribute references in the child data.

Consider the example of a request based on the Provision Resource request model for a E-Business resource. The following table shows details of the parent process form definition along with child process form details:

Child Form	Attribute Name
UD_RESPONS	Responsibility Start Date
	Responsibility End Date
	Responsibility Name

See Also: [Example 23–3, "Provision E-Business Resource Dataset"](#) for the Provision E-Business Resource request dataset

- For AttributeReference name="Server", the value of widget is specified as itresource-lookup. This indicates that for the Server field, a lookup with available IT resource parameters will be available to the user. If widget="itresource-lookup", then a value for the itresource-type element must be specified. For example, itresource-type="EBIZServer" indicates that for the Server lookup field, all the IT resource parameters for the EBIZServer IT resource type must be available for selection. User can search and select an it resource instance by using this lookup.

Note: IT resource type is a template for all IT resource definitions associated with the connector. An IT resource type specifies the parameters that are common to all IT resource instances, such as host servers and computers, of that particular IT resource type. See "Managing Connector Lifecycle" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager* for more information about IT resources and IT resource types.

See Also: "Creating Request Templates" in the *Oracle Fusion Middleware User's Guide for Oracle Identity Manager* for information about how the attributes are displayed in the Create Request Template wizard

23.1.4 Common Request Dataset

Oracle Identity Manager has a default dataset that is common for all the resources. The common request dataset defines an attribute that is common to all the resources.

ResourceCommonDataset is the default common dataset that is common for all the resources. This defines the Service Account attribute, which is common across all resources.

As a result, even if the selected resource does not have a dataset, the attribute from the common dataset is shown in request data collection during request creation. During the request data collection, attributes from both common dataset and the resource specific dataset are shown. In other words, the request collection data is a union of the common dataset and the dataset associated with the entity.

Note: The common request dataset is shipped with Oracle Identity Manager, and it cannot be customized.

23.1.5 Configuring Localized Values for Request Datasets

All default request datasets have translations for dataset attributes. But for custom request datasets that you create, localization for those dataset attributes must be added to custom resource bundles.

This section describes the configuration of localized values for request datasets and dataset attributes in the following sections:

- [Localization for Request Dataset Attributes](#)
- [Localization of Column Names in LookupQuery for Dataset Attributes](#)

23.1.5.1 Localization for Request Dataset Attributes

This section describes the conventions for enabling localization support for dataset attributes. As an example, the CreateUserDataSet.xml defines the role attribute as follows:

```
<AttributeReference name="Role" attr-ref="Role" available-in-bulk="false" type="String" length="20" widget="dropdown" lookup-code="Lookup.Users.Role" required="true"/>
```

The translation for this attribute is configured as:

```
request.dataset.User.Role=USER_TYPE
```

Here, `request.dataset.User.Role` is the translation key, and `USER_TYPE` is the actual translation or translation value. Framing the translation key decides the actual translation. Any translation key that is related to request dataset attributes must start with `request.dataset`. This is followed by an object name, such as an entity type, entity subtype if present, parent attribute name if present, attribute name, and predefined value for attribute. The types of translation keys are divided into the following categories:

- [Request Dataset Attributes for Nongeneric Request Models](#)
- [Request Dataset Attributes for Generic Request Models](#)
- [Child Attributes](#)
- [Predefined Values for Attributes](#)

Request Dataset Attributes for Nongeneric Request Models

The translation key for request datasets of nongeneric request models must not contain an entity subtype because as it is not present in the dataset. For example, the Role attribute in `CreateUserDataSet.xml` is defined as follows:

```
<AttributeReference name="Role" attr-ref="Role" available-in-bulk="false"
type="String" length="20" widget="dropdown" lookup-code="Lookup.Users.Role"
required="true"/>
```

The translation key is:

```
request.dataset.User.Role
```

Here:

- **request.dataset** is the fixed string that must be present at the beginning of any dataset attribute.
- **User** corresponds to the entity type. This must be same as the entity-type property of request-model element in the `CreateUserRequestModel.xml` file. It can be Resource or Role depending on the request model.
- **Role** corresponds to the actual attribute for which translation is being added. This corresponds to the name property of the Role attribute reference in the dataset.

Request Dataset Attributes for Generic Request Models

The translation key for request datasets of generic request models must contain entity subtype. For example, for EBS Server attribute in `ProvisionResourceBusiness Suite User.xml`, the following must be defined:

```
<AttributeReference name="EBS Server" attr-ref="EBS Server" type="Long"
widget="itresource-lookup" required="true" available-in-bulk="true"
itresource-type="eBusiness Suite UM" length="40"/>
```

The translation key is:

```
request.dataset.Resource.eBusiness\ Suite\ User.EBS\ Server=EBS Server
```

Here:

- **request.dataset** is the fixed string that must be present at the beginning of any dataset attribute.
- **Resource** corresponds to the entity type in the `ProvisionResourceRequest.xml` file.

- **eBusiness\ Suite\ User** is the entity subtype, which is the same as entity property of request-data-set element in ProvisionResourceBusiness Suite User.xml. This is optional and is present in this example because the Provision Resource request model is of generic type.
- **EBS\ Server** corresponds to the actual attribute for which translation is being added. This corresponds to name property of EBS Server attribute reference in the dataset.

Child Attributes

For child attributes, the translated key additionally contains the parent attribute name that is necessary to uniquely identify the attribute. For example, the Application Name attribute in ProvisionResourceBusiness Suite User.xml, as shown:

```
<AttributeReference available-in-bulk="true" length="10" widget="text"
type="String" attr-ref="UD_EBS_RESP" name="EBS_RSO">
  <AttributeReference name="Application Name" attr-ref="APPLICATION_NAME"
type="String" length="256" widget="lookup-query" available-in-bulk="true"
required="true">
    <lookupQuery lookup-query="select lkv_encoded, lkv_decoded from lkv lkv,
lku lku where lkv.lku_key=lku.lku_key and
lku_type_string_key='Lookup.EBS.Application' and instr(lkv_encoded,concat('$Form
data.EBS Server', '~'))>0" display-field="lkv_decoded" save-field="lkv_encoded"/>
  </AttributeReference>
</AttributeReference>
```

APPLICATION_NAME is a child attribute of *EBS_RSO* attribute. The translation key for *APPLICATION_NAME* consists of:

- `request.dataset.Resource.eBusiness\ Suite\ User.EBS_RSO.Application\ Name = APPLICATION_NAME`
- **request.dataset** is the fixed string that must be present at the beginning of any dataset attribute.
- **Resource** corresponds to the entity type in the ProvisionResourceRequest.xml file.
- **eBusiness\ Suite\ User** is the entity property of request-data-set element in ProvisionResourceBusiness Suite User.xml. This is also referred as entity subtype. This is optional and is present in this example because Provision Resource request model is of generic type.
- **EBS_RSO** is the parent attribute of Effective Start Date attribute, which identifies it uniquely. This corresponds to the name property of *EBS_RSO* attribute reference.
- **Application\ Name** corresponds to the actual attribute for which translation is being added. This corresponds to the name property of *APPLICATION_NAME* attribute reference.

Predefined Values for Attributes

You can have attributes that have predefined set of values, such as lookup values or lookup-code values. Translations can be added for these values as well because these values are displayed on the user interface. The translation key for this type of value is similar to that of child attributes. For example, the Role attribute in the CreateUserDataSet.xml request dataset has predefined set of values, including Employee, Full-Time Employee, and Part-Time Employee. These values are displayed as drop-down in the UI for Create User Request. These values are predefined by configuring a Lookup Definition with code Lookup.Users.Role. This is specified as

lookup-code="Lookup.Users.Role" in the Role AttributeReference, as shown in the following example:

```
<AttributeReference name="Role" attr-ref="Role" available-in-bulk="false"
type="String" length="20" widget="dropdown" lookup-code="Lookup.Users.Role"
required="true"/>
```

Translation for Employee value of Role attribute is:

```
request.dataset.User.Role.LOV.Employee=Employee
```

Here:

- **request.dataset** is the constant part which must be prefixed to every dataset attribute.
- **User** corresponds to the entity type.
- **Role** corresponds to the actual attribute for which Employee is a predefined value. This corresponds to name property of Role AttributeReference in the dataset.
- **LOV** is to be added to specify that the string following LOV is the predefined value for Role.
- **Employee** is the predefined value for which the translation is being added. This must be the value in Decode column of lookup definition.

Note: The lookup definitions have Code Key and Decode columns. For example, Code Key = EMP, Decode=Employee.

The attributes in the default request datasets are already present. However, if you want to add any new attributes to the dataset, then you can also add translations for these attributes in the same way as described in this section, depending on which category they fall into.

23.1.5.2 Localization of Column Names in LookupQuery for Dataset Attributes

In request datasets, there might be attributes that are of type lookup-query. For example:

```
<AttributeReference name="Application Name" attr-ref="APPLICATION_NAME"
type="String" length="256" widget="lookup-query" available-in-bulk="true"
required="true">
    <lookupQuery lookup-query="select lkv_encoded as Application
Key,lkv_decoded as Application Name from lkv lkv, lku lku where
lkv.lku_key=lku.lku_key and lku_type_string_key='Lookup.EBS.Application'"
display-field="APPLICATION_NAME" save-field="APPLICATION_KEY"/>
</AttributeReference>
```

The columns mentioned in the query are displayed on the UI in a data collection step during request creation. You can localize the lkv_encoded and lkv_decoded column names by adding the column aliases as keys in resource bundle. For example, in the above query, Application Key and Application Name are the aliases for lkv_encoded and lkv_decoded columns respectively. They can be localized by adding translation to custom resource bundles, as follows:

```
Application\ Key=APPLICATION_KEY
```

```
Application\ Name=APPLICATION_NAME
```

23.2 Step 2: Uploading Request Datasets into MDS

After creating a request dataset XML file, it must be uploaded to MDS, which can be done by using the MDS import/export utility tools provided by Oracle Identity Manager. When the upload is done, request engine loads the dataset during request creation, the attribute references and attributes are shown in data collection step. Similarly, you can delete or export any dataset file from MDS repository by using similar tools.

See Also: ["Chapter 33, "MDS Utilities and User Modifiable Metadata Files"](#) for detailed information about the MDS utilities used for importing and exporting files and modifying Oracle Identity Manager metadata

To upload the request datasets to MDS:

Note:

- The dataset must be updated from time to time based on the resource requirement changes.
 - For updating an existing dataset, make sure that there are no pending requests that use that dataset.
-
-

1. The `metadata_from_loc` property in the `weblogic.properties` file specifies the top-level directory from which to import XML files. Create a subdirectory to keep the request datasets in, and copy the dataset into this directory. It is recommended that you create a subdirectory structure such as `/custom/RESOURCE_NAME`. For example, if the `metadata_from_loc` property is set to `/scratch/datasets/upload` and you are creating a dataset for the EBS resource, then the dataset is to be placed in the `/scratch/datasets/upload/custom/EBS/` directory.

Note: Make sure that this directory contains only the required datasets and no other files.

2. Go to the `OIM_HOME/bin` directory and run `weblogicImportMetadata.sh` or `weblogicImportMetadata.bat` script. See ["Setting up the Environment for MDS Utilities"](#) on page 33-1 for details about running the `weblogicImportMetadata.sh` or `weblogicImportMetadata.bat` script.

23.3 Step 3: Creating SOA Composites Required for Approval

Oracle Identity Manager provides a few predefined Service-Oriented Architecture (SOA) composites. However, you can define your own composites and use them in request approvals. See ["Creating New SOA Composites"](#) on page 25-2 for information about creating the composites.

23.4 Step 4: Registering the SOA Composites in Oracle Identity Manager

See ["Registering a SOA Composite with Oracle Identity Manager"](#) on page 25-4 to register the SOA composites in Oracle Identity Manager.

23.5 Step 5: Defining Request Approvals

A request goes through multiple approvals before it is executed. After the request is submitted, it must obtain approvals at different levels. Approvals are controlled and configured by a set of approval policies.

An approver is able to view the request data. Approver cannot change the data provided by the requester. Approver can only provide data for the attributes that are set as `approver-only="true"` in the request dataset.

This section describes the following topics:

- [Approval Workflows](#)
- [Approval Levels](#)
- [Creating Approval Policies](#)

23.5.1 Approval Workflows

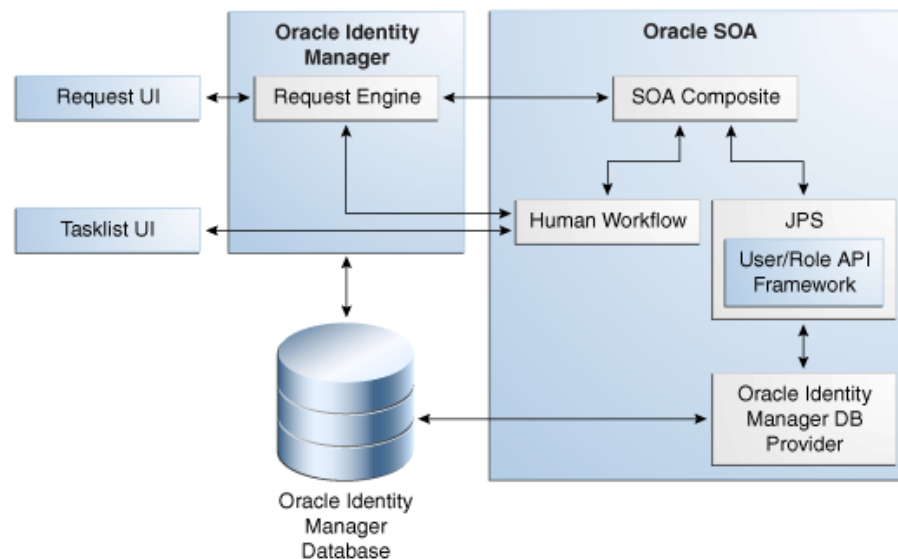
After a request is submitted, if any approval needs to be initiated, then the request service starts the workflow process in the backend workflow engine. Oracle SOA is used as workflow engine by the request service.

See Also:

- "Managing Approval Policies" in the *Oracle Fusion Middleware User's Guide for Oracle Identity Manager* for information about approval policies
- [Chapter 25, "Developing SOA Composites"](#) for detailed information about workflow service

SOA server hosts SOA composites and human workflow. The integration of the request service and SOA can be explained with the help of [Figure 23-1](#):

Figure 23-1 Request Service and SOA Integration



The following process describes how Oracle SOA works with request service for selecting an approval workflow:

1. A request is created by using the request management UI, which is Oracle Identity Manager Self Service or Advanced Administration.
2. When the request is submitted, the request engine calls the SOA composites that are deployed in Oracle SOA.

Note: Oracle SOA is independent of Oracle identity Manager. The backend Business Process Execution Language (BPEL) service invokes the approval workflow. In addition to the default BPEL workflows shipped with Oracle Identity Manager, you can define your own workflows in BPEL based on your requirement. For information about customizing the BPEL workflows, see JDeveloper documentation on the Oracle Technology Network (OTN) Web site at the following URL:

<http://www.oracle.com/technetwork/developer-tools/jdev/documentation/index.html>

3. Oracle SOA determines whom to assign the request by using the SOA composite logic through Java Platform Security (JPS). Oracle SOA uses the same set of users and roles as in Oracle Identity Manager. This is enabled by the Oracle Identity Manager database provider.
4. Oracle SOA assigns the task to the assignee based on the information provided by the DB provider.
5. The list of assigned tasks to the logged in user and role is displayed in the TaskList UI.
6. Using the TaskList, the approver approves or rejects the requests.
7. The approval outcome is send back to the request engine through SOA.
8. If a request has been approved, then the next action is determined based on the request type, beneficiary, or associated resource. If a request has been rejected, then the request processing stops.

After the task is assigned, the user can login to the TaskList UI in Oracle Identity Manager Self Service to get a consolidated view of Human tasks and Oracle Identity Manager requests.

TaskList uses the task query service APIs to communicate with Oracle SOA. These APIs are provided by Oracle SOA server. Oracle Identity Manager uses the SOAP or RMI protocol to communicate with Oracle SOA based on the configuration. RMI is the default protocol. For information about the configuration between Oracle Identity Manager and Oracle SOA, see [Chapter 25, "Developing SOA Composites"](#).

See Also: "Managing Tasks" in the *Oracle Fusion Middleware User's Guide for Oracle Identity Manager* for information about performing various request-related operations by using the TaskList

23.5.2 Approval Levels

Each request may need to go through three levels of approvals: template-level, request-level, and operation-level. SOA composites must be registered with Oracle Identity Manager in order to use those as approval processes in Oracle Identity Manager. Registering means letting Oracle Identity Manager know what approval processes are deployed and can be used at run time.

The approval levels are described in the following sections:

- [Template-Level Approval](#)
- [Request-Level Approval](#)
- [Operation-Level Approval](#)

23.5.2.1 Template-Level Approval

These are the approvals defined at the request templates. Each template can define additional approvals on top of what is defined by the approval policy configuration. At this level, either the complete request is approved or rejected. For bulk requests, there is no partial approval or rejection. Each template can define an optional approval process that must be initiated at the template level. If no approval process is defined in the template, then the template level is auto approved.

Note: Request-level and operation-level approval are associated with approval policies. For template-level approval, there are no approval policies associated. The approval processes are defined at the template and are run directly, without an association with approval policies. For information about approval policies, see "Managing Approval Policies" in the *Oracle Fusion Middleware User's Guide for Oracle Identity Manager*.

An example of template-level approval can be the approval required by an HR representative for the user creation of all contract employees in addition to the approval required from the employees' managers and IT administrators. The additional approval from the HR representative can be configured as an approval process while creating a template. The template can be used to create and submit the request.

Note: This level of approval is not required for child requests.

23.5.2.2 Request-Level Approval

These are the approval for the entire request. These are based on the approval policy configuration.

An example of request-level approval is the approval required by the requestor's manager when a request is raised to provision a laptop to a user.

Approval process that needs to be used for the request-level approval for a request is determined by the approval policies defined at the request level. If no approval policies are defined for the request level for the given type of request, then the default request approval process is used. By default, all the request-level approval are assigned to the administrator. Therefore, the default configuration is secure. If multiple approval policies exist, then the approval policy rules are evaluated in the order of approval policy priority to figure out the appropriate approval policy. The approval policy rules indicate the request engine which approval process to pick up for the particular approval. The request engine selects the approval process defined in the approval policy that is selected based on the approval policy rule evaluation.

For example, when a request to create a user is submitted, the approval policy selection methodology finds out how many approval policies exist for the Create User request model in the order of priority. The approval policy rule for the highest approval policy priority is evaluated. An example of approval policy rule is that manager's first name and last name must be John and Doe respectively. If the approval

policy rule for the highest priority does not match, then the approval policy rule for the next approval policy priority is evaluated. When the first approval policy rule matches the criteria specified in the approval policy, the corresponding approval process for that approval policy is selected for that request at the request level. If all the approval policy rules do not match, then the default approval process for the request level is selected.

Note:

- For information about creating approval policy rules, see "Creating Approval Policies" in the *Oracle Fusion Middleware User's Guide for Oracle Identity Manager*.
 - Request-level approval is not required for child requests because the bulk request is broken down to child requests after successful approval of the bulk request at the request level. See Section 4.3, "Bulk Requests and Child Requests" in the *Oracle Fusion Middleware User's Guide for Oracle Identity Manager* for information about bulk requests and child requests.
 - The Self-Register User request template has organization as the approver-only attribute. Therefore, any approval policy associated with this template must not have auto-approval set at request-level because the attribute is mandatory and must be provided by the approver.
-
-

23.5.2.3 Operation-Level Approval

These are the approvals for the operation being performed by this request type. This level requires approval selection methodology name and parameters to be passed to the methodology. A methodology suggests which approval workflow is to be used for this operation. The request type and scope may also define the methodology-specific parameters that are required for determining the approval process. Scope is a key associated with the types of request types, as shown in [Table 23-3](#):

Table 23-3 Request Types and Associated keys

Request Types	Scope
All request types related to the user entity, such as:	organization
<ul style="list-style-type: none"> ■ Create User ■ Self-Register User ■ Modify Self Profile ■ Modify User Profile ■ Delete user ■ Enable User ■ Disable User 	

Table 23–3 (Cont.) Request Types and Associated keys

Request Types	Scope
All request models related to resources, such as:	resource
<ul style="list-style-type: none"> ■ Provision Resource ■ Modify Provisioned Resource ■ Self-Request Resource ■ Enable Provisioned Resource ■ Disable Provisioned Resource ■ Modify Provisioned Resource ■ Self Modify Provisioned Resource ■ De-provision Resource ■ Self De-Provision Resource 	
All request types related to the role entity, such as:	role
<ul style="list-style-type: none"> ■ Assign Roles ■ Remove from Roles 	

Note: For the Create Role, Modify Role, and Delete Role types, the operation-level approval is auto-approved.

For example, based on the scope, for the request of type Provision Resource, you must select the resource to associate the approval policy at operation level at the time of approval policy creation. Similarly, for a Create User request type, you must select an organization, and for the Assign Roles request type, you must select a role at the operation level during approval policy creation.

The approval policies that you create, along with the approval policy priorities, approval policy rules, and scope, decide which approval process is to be selected for a request at the operation level.

An example of operation-level approval is the approval required by the IT administrator, who is responsible for issuing a laptop to users, after the request-level approval is obtained for provisioning a laptop to a user.

For a bulk request, operation-level approval are required for individual child requests. Each individual child request can be approved or rejected independently. For example, for a provision resource to user request, there can be multiple beneficiaries, multiple resources, or both. Therefore, at the operational level, provisioning of each resource to each user generates a child request, which can be approved or rejected independently.

23.5.3 Creating Approval Policies

Create the required approval policies for selecting appropriate SOA composites for approval. See ["Step 5: Defining Request Approvals"](#) on page 23-25 for the concepts related to approvals. See ["Managing Approval Policies"](#) in the *Oracle Fusion Middleware User's Guide for Oracle Identity Manager* for the procedure to create approval policies.

23.6 Step 6: Creating Request Templates

A request template lets you customize a request type for a purpose. In other words, it allows you to control the attributes of the request by controlling the various capabilities in the UI. See "Managing Request Templates" in the *Oracle Fusion Middleware Developer's Guide for Oracle Identity Manager* for details about creating and managing request templates.

23.7 Extending Request Management Operations

You can customize certain aspects of request management operations to allow greater flexibility and implement customized logic for additional functionality. To achieve this, you can use request management plug-ins. There are plug-in points that you can use to implement customization.

See Also:

- "Managing Requests" in the *Oracle Fusion Middleware User's Guide for Oracle Identity Manager* for detailed information about the concepts and tasks related to requests
- "Creating and Searching Requests" in the *Oracle Fusion Middleware User's Guide for Oracle Identity Manager* for detailed information about creating and managing requests in Oracle Identity Manager

This section discusses the plug-in points in the following topics:

- [Running Custom Code Based on Request Status Change](#)
- [Validating Request Data](#)
- [Prepopulation of an Attribute Value During Request Creation](#)

23.7.1 Running Custom Code Based on Request Status Change

In Oracle Identity Manager, a request undergoes change in status at each stage of its lifecycle. The request engine exposes a plug-in point that allows running of custom code during request status change. A plug-in with custom code that extends this plug-in point can be implemented and registered for running the code. The plug-in point is the `oracle.iam.request.plugins.StatusChangeEvent` interface with the **public void followUpActions(String reqId)** method. This method consists of the request id parameter, using which the request details can be obtained with the help of request management APIs.

See Also: [Chapter 7, "Developing Plug-ins"](#) for detailed information about plug-ins and plug-in points

Any code that is to be run during the status change must be implemented in the `followUpActions()` method in a plug-in class that implements the `oracle.iam.request.plugins.StatusChangeEvent` interface. You must specify at which request status change this plug-in is to be run in the `plugin.xml` file.

For example, when a request in Oracle Identity Manager moves to the Request Failed status, you want to run a custom code that sends a notification to an administrator. To do so:

1. Create a new plug-in class with name `RequestFailedChangeEvent` that implements the `oracle.iam.request.plugins.StatusChangeEvent` interface. This class must have

the logic of sending a notification to the administrator in the `followUpActions(String reqId)` method.

2. Define `plugin.xml` in following standard format, as specified by the plug-in framework:

```
<oimplugins xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <plugins pluginpoint="oracle.iam.request.plugins.StatusChangeEvent">
    <plugin pluginclass="com.mycompany.RequestFailedChangeEvent"
      version="1.0" name="RequestFailedChangeEvent">
      <metadata name="status">
        <value>Request Failed</value>
      </metadata>
    </plugin>
  </plugins>
</oimplugins>
```

In this XML definition, the metadata part specifies at which stage the plug-in must be run. This is done by specifying the metadata value as `Request Failed`, which means that the `com.mycompany.RequestFailedChangeEvent` plug-in will run when a request moves to the Request Failed status.

3. Register the plug-in with Oracle Identity Manager. See ["Registering Plug-ins"](#) on page 7-6 for information about registering plug-ins in Oracle Identity Manager.

23.7.2 Validating Request Data

You can use the `RequestDataValidator` plug-in to add custom validation of request data after submission. The plug-in point for this is the `oracle.iam.request.plugins.RequestDataValidator` interface with public `void validate(RequestData requesterData)` method. See ["The DataSetValidator Element"](#) on page 23-4 for information about the `DataSetValidator` element.

23.7.3 Prepopulation of an Attribute Value During Request Creation

Prepopulation plug-in is associated with an attribute reference or attribute in request dataset. This can be used to prepopulate an attribute value by running custom code during request creation. Requester can modify the value that is prepopulated if required.

The plug-in point for this is `oracle.iam.request.plugins.PrePopulationAdapter` with public `Serializable prepopulate(RequestData requestData)` method. Use this plug-in only for the following request types:

Provision Resource, Self-Request Resource, Create User, Self-Register User.

See ["The PrePopulationAdapter Element"](#) on page 23-9 for more details.

Understanding Approval Process Development in Oracle SOA Suite

Workflow-based provisioning is a key feature of Oracle Identity Manager that enables you to automate the business processes that manage user access in an organization. Oracle Identity Manager leverages services enabled and managed by Oracle Service-Oriented Architecture (SOA) Suite to provide an interactive environment to request, approve, and manage user access. Oracle SOA Suite provides the back-end services and management capabilities required to implement SOA.

Oracle Identity Manager makes use of the following components of the SOA Suite:

- **BPEL Process Manager**, which provides the end-to-end solution for creating and managing business processes
- **Human Workflow**, which manages the lifecycle of human tasks, including creation, assignment, deadlines, expiration, and notifications
- **Oracle Business Rules**, which allows you to define complex business rules to support request assignment, process selection, and approver resolution

This chapter contains the following sections:

- [Integration with Oracle SOA Suite](#)
- [Predefined SOA Composites](#)
- [Developing an Approval Process for Oracle Identity Manager](#)
- [Monitoring Oracle Identity Manager SOA Composites](#)
- [Enabling Oracle Identity Manager to Connect to SOA](#)

24.1 Integration with Oracle SOA Suite

In Oracle Identity Manager, SOA composites are used as approval processes. Integration of Oracle Identity Manager with Oracle SOA Suite is described in the following sections:

- [Integration Prerequisites](#)
- [Integration Components](#)

24.1.1 Integration Prerequisites

Before developing SOA composites for Oracle Identity Manager, the following prerequisites are recommended:

- Knowledge of XML and XPath

- Knowledge of Oracle Identity Manager APIs.
- Knowledge of developing and deploying SOA composites. See [Chapter 25, "Developing SOA Composites"](#) for details.
- Knowledge of using JDeveloper and the SOA Composite Editor. For details about using the JDeveloper IDE, see the JDeveloper Tutorial Series at following URL:
<http://www.oracle.com/technetwork/developer-tools/jdev/overview/index.html>

See Also: *Oracle Fusion Middleware Tutorial for Running and Building an Application with Oracle SOA Suite* for information about working with Oracle SOA Suite

24.1.2 Integration Components

The integration of Oracle Identity Manager with Oracle SOA Suite consists of the following components:

- SOA Suite installation as a part of installing Oracle Identity Manager.
- One or more SOA composites. You can either extend the default composites shipped with Oracle Identity Manager or develop your own composites.

See Also: *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite* for information about developing SOA composites

- The SOA composite, which consists of:

- The request payload.

Oracle Identity Manager provides the SOA composite the details of the request by using XML. This is called the request payload. The SOA composite can use all or part of the payload to determine the next step(s) to take in the approval process. The payload format is fixed.

- Oracle Identity Manager API calls (optional).

Oracle Identity Manager provides only the most essential information to the SOA composite to keep the payload small and also to ensure security. If the business process requires additional data, then you can use a Java embedding step to obtain more information about the requester, the beneficiary, or what is being requested. For information on how to invoke Oracle Identity Manager APIs, see [Chapter 31, "Using APIs"](#).

- One or more Human Tasks.

Human tasks are steps in the overall business process where manual intervention, in the form of approvals, is required. A human task can consist of multiple steps, serial or parallel or a combination of both, where the task is assigned to one or more users or roles or a combination of both. You can define these human tasks and add notification, deadlines, and escalation rules. For information about how to design human tasks, see *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

Note: In most approval scenarios, the composite contains only one human task. In some instances, additional human tasks may be required if the routing rules cannot be satisfied by using Oracle Business Rules. For example, the composite for resource request type contains multiple human tasks, one per resource. As a best practice, you must try to streamline the approval rules to facilitate reuse of the composites and human tasks.

- One or more rulesets.

There are specific business requirements that must be met when fulfilling requests. SOA composites leverage Oracle Business Rules to satisfy these requirements. A collection of rules developed by using Oracle Business Rules is called a ruleset. A composite can have one or more rulesets. Human tasks can also leverage these rules to determine the participants and the task routing. For information about how to design human tasks, see *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

- A certified version of JDeveloper, for example, JDeveloper 11.1.1.3.
- The SOA Design Time, also known as the SOA Composite Editor Extension for JDeveloper.

24.2 Predefined SOA Composites

Table 24–1 lists the predefined SOA composites in Oracle Identity Manager that can be used as approval processes.

Table 24–1 Predefined Workflow Composites

Workflow Composite	Description
DefaultRequestApproval	This is the default request-level approval. By default, the request-level approval goes to the System Administrator, xelsysadm, for request-level approval.
DefaultOperationalApproval	This is the default operation-level approval. By default, the approval task is assigned to the System Administrator, xelsysadm, for operation-level approval.
BeneficiaryManagerApproval	This acquires approval from the beneficiary's manager. This can be associated with the following: <ul style="list-style-type: none"> ■ The request types that have a beneficiary. Examples of such request types are Provision Resource and Assign Roles. ■ All user models except Create User and Self-Register User. <p>This composite must be associated at the operational level of approval because a request can have multiple beneficiaries at the request level.</p>
DefaultRoleApproval	This SOA composite creates a single approval task that is assigned to the SYSTEM ADMINISTRATORS role for approval.
RequesterManagerApproval	This SOA composite creates a single approval task that is assigned to the requester's manager for approval. <p>Note: This cannot be associated with unauthenticated request types, such as Self Register User.</p>

Table 24–1 (Cont.) Predefined Workflow Composites

Workflow Composite	Description
ResourceAdministratorApproval	This SOA composite creates a single approval task that is assigned to the SYSTEM ADMINISTRATORS role for approval. This must be associated with the request types that are related to resources. This composite is used at the operational level of approval.
ResourceAuthorizerApproval	This SOA composite creates a single approval task that is assigned to resource authorizers (with highest priority) for approval. This must be associated with the request types that are related to resources. This composite is used at the operational level of approval.
DefaultSODApproval	This SOA composite creates an approval task that is assigned to the system administrator, starts SoD check, and after the SoD result is available, it creates another approval task assigned to the SOD Administrators role. This must be associated with request types to provision or modify resources at the operational level if SoD check is required.

Note: Human tasks in these default composites are configured to send notifications to the assignee of the human task.

For configuring Oracle SOA server to send e-mail notifications, see "Configuring Oracle User Messaging Service" in the *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite and Oracle Business Process Management Suite*.

In addition to the SOA composites listed in [Table 24–1](#), the AutoApproval composite is available in SOA. But Oracle recommends that users must select the **Auto Approval** option rather than using the AutoApproval composite to avoid a round trip from Oracle Identity Manager to SOA. This is because in an approval policy, if a user selects the **Auto Approval** option, then for a request that matches this approval policy, it is auto approved in Oracle Identity Manager. The approval does not go to SOA and come back. However, in an approval policy, if a user selects the AutoApproval as the approval process, then for a request that matches this approval policy, the AutoApproval composite is initiated in SOA and immediately a response 'Approved' is sent back to Oracle Identity Manager.

24.3 Developing an Approval Process for Oracle Identity Manager

To develop an approval process for Oracle Identity Manager:

Note: As a part of developing an approval process for Oracle Identity Manager, you must create request datasets, upload the request datasets to MDS, create or use request templates, and create approval policies. For details, see [Chapter 23, "Configuring Requests"](#).

1. Create a JDeveloper workspace by using the new_project.xml utility. This utility is in the `OIM_HOME/workflows/new-workflow/` directory. See "[Creating a New SOA Composite](#)" on page 25-2 for details.
2. Open the JDeveloper workspace and modify the BPEL process and the human task as required.

3. Deploy the composite in any one of the following ways:
 - By directly deploying the composite by using SOA Deployer. See "Deploying a SOA Composite Application" in the *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.
 - By moving SOA composite applications from one environment to another. See "Moving SOA Composite Applications to and from Development, Test, and Production Environments" in the *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite* and "Administering SOA Composite Applications" in the *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite*.

24.4 Monitoring Oracle Identity Manager SOA Composites

Oracle Identity Manager SOA composites are managed and monitored by using Oracle Enterprise Manager Fusion Middleware Control Console. For more information about managing and monitoring deployed SOA composites, see *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite*.

24.5 Enabling Oracle Identity Manager to Connect to SOA

Oracle Identity Manager connects to SOA as SOA administrator, for which the username is "weblogic" by default. During the Oracle WebLogic Server domain creation, if the username provided is other than this, then Oracle Identity Manager is not able to connect to SOA as SOA administrator. To enable Oracle Identity Manager to work with any Oracle WebLogic Server administrator user, and thereby, connect to SOA without any problem, perform the following postinstallation steps:

1. Login to Enterprise Manager by using the following URL:
`http://ADMINSERVER_HOST:ADMINSERVER_PORT/em`
2. Right click **Identity and Access/oim(11.1.1.3.0)**, and select **System Mbean Browser**.
3. Expand **oracle.iam** under application-defined Mbeans, and select **Server: OIM_SERVER_NAME, Application: oim, XML config, config, XMLConfig.SOAConfig**, and then select the **SOAConfig** Mbean.
4. View the username attribute. By default, the value of this attribute is weblogic. Change this to the correct Oracle WebLogic Server administrator username.
5. Click **Apply**.
6. Navigate to the `OIM_ORACLE_HOME/common/bin/` directory in the Oracle Identity Manager deployment.
7. Start the WebLogic Scripting Tool (WLST) by running the following command:
`./wlst.sh`
8. At the prompt, enter `connect()`. When prompted, enter the Oracle WebLogic Server administrator username, password, and administrative server connection string.
9. To delete the default SOA administrator username and password credential from CSF, run the following command:
`deleteCred(map="oim", key="SOAAdminPassword");`
10. To create the new credential that Oracle Identity Manager uses to connect to SOA as SOA administrator, run the following command:

```
createCred(map="oim", key="SOAAdminPassword", user="xelsysadm",  
password="ADMINISTRATOR_PASSWORD");
```

Replace *ADMINISTRATOR_PASSWORD* with the actual password.

11. Confirm that the correct value has been seeded by running the following command:

```
listCred(map="oim", key="SOAAdminPassword");
```

12. Exit WLST shell by running the following command:

```
exit()
```

13. Login to Oracle Identity Manager Administrative and User Console by using the administrator login credentials.
14. Create a new user with the same login ID as the Oracle WebLogic Server administrator username.
15. Search for the Administrators role. Open the role details and click the Members tab.
16. Remove all the existing members of the role.
17. Add the newly created user as member of this role.
18. Confirm that this role has only one member. This member must be the user created in step 14.
19. Restart Oracle Identity Manager managed server.

Developing SOA Composites

The primary goal of any provisioning system is to manage requests submitted by users and provision resources to users. Request completion involves execution of associated approval processes. These approval processes are deployed as Service Oriented Architecture (SOA) composites running on the SOA Server. Request service is responsible for execution and management of such approval processes. [Figure 23-1, "Request Service and SOA Integration"](#) shows the integration of the request service and SOA.

The interaction between Oracle Identity Manager and SOA Server is explained in the following steps:

1. The user creates a request by using the Oracle Identity Manager Self Service. The request can be of any one of all the request types supported by Oracle Identity Manager.
2. Request service evaluates the approval policy, and the SOA composite to be instantiated is selected.

Note: The composites must be registered with Oracle Identity Manager in order for them to be kicked off when the request is submitted. For information about registering workflows with Oracle Identity Manager, see ["Registering a SOA Composite with Oracle Identity Manager"](#) on page 25-4.

3. Request service contacts SOA Server to instantiate the selected SOA composite. Instantiates selected composite instance on SOA server.
4. SOA composite run starts and human approval task is assigned for approval.
5. The approver logs in to the Task List in the Oracle Identity Manager Self Service console, and approves the request.
6. After the approval, the composite instance run is completed, which is notified to the request service.
7. Request service moves the request to the next stage.

This chapter discusses the following topics:

- [Creating New SOA Composites](#)
- [Modifying Existing SOA Composites](#)

25.1 Creating New SOA Composites

To create a new SOA composite that can be used as an approval process, you must perform the following steps:

1. [Creating a New SOA Composite](#)
2. [Deploying a SOA Composite in Oracle SOA Server](#)
3. [Prerequisites for Communication to Oracle Identity Manager Through SSL Mode](#)
4. [Registering a SOA Composite with Oracle Identity Manager](#)

25.1.1 Creating a New SOA Composite

To use a SOA composite as an approval process, it must adhere to certain standards. These standards ensure that the request service is able to instantiate and manage such composites correctly. These standards are:

- The following attributes are mandatory for BPEL process:
 - RequestID of type String
 - RequestModel of type String
 - RequestTarget of type String
 - URL of type String
 - RequesterDetails of XML Element
 - BeneficiaryDetails of XML Element
 - ObjectDetails of XML Element
 - OtherDetails of XML Element

The RequestID, RequestModel, RequestTarget, and URL attributes are always set with valid values for all types of requests.

RequesterDetails is an XML element. This element is filled up with valid values for all requests that requires authentication. Requester details is empty for the requests of type Self-Register User because the requester is anonymous user.

BeneficiaryDetails is an XML element. This element is filled up with valid values for all requests that have a beneficiary, for example, Provision Resource and Assign Roles. This is filled up only if the request is associated with single beneficiary. If the request is associated with multiple beneficiaries, then BeneficiaryDetails is empty. BeneficiaryDetails element always has valid value for simple requests and child requests that have a beneficiary. Therefore, it is recommended to use this XML element in SOA composites that are used as approval processes at the operational level of approval. This is because at the operational level of approval, the request is associated with only one beneficiary.

ObjectDetails is an XML element. This element is filled up with valid values for all requests that are associated with the Resource entity. This is filled up only if the request is associated with single resource. If the request is associated with multiple resources, then ObjectDetails is empty. The ObjectDetails element always has valid value for simple and child requests that are associated with resource. Therefore, it is recommended to use this XML element in SOA composites that are used as approval processes at the operational level of approval. This is because at the operational level of approval, the request is associated with only one resource.

- All the attributes that are mandatory for the BPEL process are referred from RequestDetails.xsd and ApprovalProcess.xsd. These files are present in the template SOA composite, which must not be modified or deleted.

Oracle Identity Manager provides a helper utility for creating custom SOA composites. This utility creates a template SOA project that adheres to all the necessary standards. This utility is located in the `OIM_HOME/workflows/new-workflow` directory.

Note:

- JAVA_HOME environment variable must be set before running this utility.
 - This utility requires Apache Ant version 1.7 or later.
-
-

To create a custom SOA composite by running the helper utility:

1. Run the following commands:

```
cd OIM_HOME/workflows/new-workflow
ant -f new_project.xml
```

2. Enter the JDeveloper application name when the following prompt is displayed:

```
Please enter application name
```

3. Enter the JDeveloper project name when the following prompt is displayed:

```
Please enter project name
```

4. Enter the name of the ADF binding service for the composite when the following prompt is displayed:

```
Please enter the service name for the composite. This needs
to be unique across applications
```

The new application is created in the `OIM_HOME/workflows/new-workflow/process-template/` directory. You can open the new application in JDeveloper for modification.

Human task in the template SOA composite is configured to send notifications to the assignee of the human task. In the custom composite that is created, the notification message can be modified based on the requirement. All the notifications to be sent to the approver must be configured in the SOA composite. For configuring Oracle SOA server to send notifications, refer to "Configuring Oracle User Messaging Service" in the *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite and Oracle Business Process Management Suite*.

Human task in the template SOA composite is configured to be assigned to the SYSTEM ADMINISTRATORS role.

25.1.2 Deploying a SOA Composite in Oracle SOA Server

For information about deploying the workflow composite in BPEL, see *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

Note: If a composite is redeployed in SOA, then all the pending approvals in Oracle Identity Manager initiated by the composite becomes stale and are removed from the user's TaskList. See "Deploying an Existing SOA Archive in Oracle JDeveloper" in the *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite* for information about deploying existing SOA composites.

25.1.3 Prerequisites for Communication to Oracle Identity Manager Through SSL Mode

If the communication to Oracle Identity Manager is through the SSL mode, then you must:

Note: For a non-SSL connection, skip this section.

- Set the *TRUSTSTORE_LOCATION* environment variable, where *TRUSTSTORE_LOCATION* is the trusted key store file location.
- Use t3s protocol instead of t3. For example, the URL for Oracle Identity Manager is:

t3s://HOST_NAME:PORT

25.1.4 Registering a SOA Composite with Oracle Identity Manager

The SOA composite must be registered with Oracle Identity Manager before it can be used as an approval process. To register a SOA composite with Oracle Identity Manager:

1. Create the *COMPOSITE_NAME.props* property file in the *OIM_HOME/workflows/registration/* directory with the following content:

```
name=COMPOSITE_NAME
category=Approval
providerType=BPEL
serviceName=REQUEST_APPROVAL_SERVICE
domainName=DOMAIN
version=REVISION_ID
payloadID=PAYLOAD
operationID=OPERATION_ID
listOfTasks=HUMAN_TASK_NAMES
```

Where:

- Replace *COMPOSITE_NAME* with the name of the SOA composite as specified in the composite.xml file.
- Replace *REQUEST_APPROVAL_SERVICE* with the name of the service exposed in the composite. This service is invoked while instantiating the composite for approval.
- Replace *OPERATION_ID* with the name of operation to be invoked on the service mentioned for the serviceName property.
- Replace *PAYLOAD* with the part name of the operation specified as the value of the operationID property.
- Replace *REVISION_ID* with the SOA composite revision as specified in the composite.xml file.

- Replace *DOMAIN* with name of SOA partition in which the composite is deployed. By default, SOA has one partition named default.
- Replace *HUMAN_TASK_NAMES* with the names of the approval tasks associated with the SOA composite. Separate the names with colon (:).

Note: Do not add any extra spaces in the properties file.

2. Run the following command from the *OIM_HOME/workflows/registration/* directory:

Note:

- JAVA_HOME environment variable must be set before running this utility.
 - This utility requires Apache Ant version 1.7 or later.
 - For a connection over SSL, you must meet the prerequisites mentioned in "[Prerequisites for Communication to Oracle Identity Manager Through SSL Mode](#)" on page 25-4.
-

```
ant -f registerworkflows-mp.xml register
```

3. Enter Oracle Identity Manager administrator username when prompted.
4. Enter Oracle Identity Manager administrator password when prompted.

Note: When you login to the SOA server by using the Oracle Identity Manager administrator credentials, the login attempt fails if your challenge questions and answers and password are not reset in Oracle Identity Manager. You must reset the challenge questions and answers and password when you first login to Oracle Identity Manager.

5. Enter Oracle Identity Manager server t3 URL when prompted.
6. Enter the complete path of the property file name that you created in step 1 when prompted.

25.2 Modifying Existing SOA Composites

To modify an existing SOA composite that can be used as an approval process, you must perform the following:

1. [Modifying a SOA Project in JDeveloper](#)
2. [Disabling a SOA Composite on Oracle Identity Manager](#)
3. [Deploying a SOA Composite in Oracle SOA Server](#)
4. [Enabling a SOA Composite with Oracle Identity Manager](#)

Note: If the SOA composite modification involves adding or removing human tasks, then perform the following steps:

1. Modify the SOA composite in JDeveloper. While modifying the composite, make sure that the SOA composite version ID is changed. The existing SOA composite version ID, which is already registered with Oracle Identity Manager, cannot be used.
2. Disable the existing SOA composite in Oracle Identity Manager.
3. Deploy the SOA composite in Oracle SOA Server with new version ID.
4. Register the SOA composite with the new version to Oracle Identity Manager.

25.2.1 Modifying a SOA Project in JDeveloper

You can modify SOA composites by using JDeveloper. If you want to modify the default SOA composites, then you can access the respective JDeveloper projects in the paths listed in [Table 25-1](#):

Table 25-1 Location of Default SOA Composites

SOA Composite	Path
BeneficiaryManagerApproval	<i>OIM_HOME</i> /workflows/composites/BeneficiaryManagerApproval.zip
DefaultOperationalApproval	<i>OIM_HOME</i> /workflows/composites/DefaultOperationalApproval.zip
DefaultRequestApproval	<i>OIM_HOME</i> /workflows/composites/DefaultRequestApproval.zip
DefaultRoleApproval	<i>OIM_HOME</i> /workflows/composites/DefaultRoleApproval.zip
ResourceAuthorizerApproval	<i>OIM_HOME</i> /workflows/composites/ResourceAuthorizerApproval.zip
ResourceAdministratorApproval	<i>OIM_HOME</i> /workflows/composites/ResourceAdministratorApproval.zip
RequesterManagerApproval	<i>OIM_HOME</i> /workflows/composites/RequesterManagerApproval.zip
DefaultSODApproval	<i>OIM_HOME</i> /workflows/composites/DefaultSODApproval.zip

For information about using JDeveloper to build applications with Oracle SOA Suite, see *Oracle® Fusion Middleware Developer's Guide for Oracle SOA Suite*.

25.2.2 Disabling a SOA Composite on Oracle Identity Manager

You can disable a SOA composite on Oracle Identity Manager by using the `registerworkflows-mp` utility located in the `OIM_HOME/workflows/new-workflow/` directory.

Note:

- `JAVA_HOME` environment variable must be set before running this utility.
- This utility requires Apache Ant version 1.7 or later.
- For a connection over SSL, you must meet the prerequisites described in "[Prerequisites for Communication to Oracle Identity Manager Through SSL Mode](#)" on page 25-4.

To disable a SOA composite on Oracle Identity Manager:

1. Using a command prompt, go to the `OIM_HOME/workflows/registration/` directory.
2. Run the following command:

```
ant -f registerworkflows-mp.xml disable
```
3. When prompted to enter username, enter the Oracle Identity Manager administrator username.
4. When prompted to enter password, enter the Oracle Identity Manager administrator password.
5. When prompted to enter the server t3 URL, enter the Oracle Identity Manager server t3 URL, for example, `t3://localhost:7001`.
6. Specify the domain of the workflow. By default, this is set to the `default` domain.
7. When prompted for the name of the workflow, enter the SOA composite name.
8. When prompted for the version of the workflow, enter the SOA composite version.

25.2.3 Deploying a SOA Composite in Oracle SOA Server

For information about deploying the workflow composite in BPEL, see *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

25.2.4 Enabling a SOA Composite with Oracle Identity Manager

To enable a SOA Composite with Oracle Identity Manager:

Note:

- `JAVA_HOME` environment variable must be set before running this utility.
 - This utility requires Apache Ant version 1.7 or later.
 - For a connection over SSL, you must meet the prerequisites described in "[Prerequisites for Communication to Oracle Identity Manager Through SSL Mode](#)" on page 25-4.
-
-

1. Run the following command:

```
ant -f registerworkflows-mp.xml enable
```
2. Perform steps 3 through 8 in listed "[Disabling a SOA Composite on Oracle Identity Manager](#)" on page 25-6.

Using Oracle Identity Manager APIs in SOA Composites

If the business process requires data that is be used in addition to the standard payload data in the SOA composite, then you can use a Java embedding step to obtain more information about the requester, the beneficiary, or what is being requested.

This chapter describes how to use Oracle Identity Manager APIs from SOA composites in the following topics:

- [Software Prerequisites](#)
- [Configuring the SOA Composite By Using JDeveloper](#)

26.1 Software Prerequisites

Before you configure SOA Server to load Oracle Identity Manager APIs from SOA composites, perform the following:

1. Install the version of JDeveloper that is compatible with the Oracle Identity Manager deployment. In addition, install any patches for JDeveloper so that JDeveloper works correctly with the SOA composites.
2. Ensure that *OIM_HOME* points to the directory on which Oracle Identity Manager is installed. For example: `/scratch/shiphome/beahome/Oracle_IDM1/server/` must point to *OIM_HOME*.

In this document, *OIM_HOME* refers to the directory in which Oracle Identity Manager is deployed. For example, `/scratch/shiphome/beahome/Oracle_IDM1/` must point to *OIM_HOME*.

26.2 Configuring the SOA Composite By Using JDeveloper

This section describes the configuration required in JDeveloper as well as in the SOA composite so that the required Java code can be introduced in the composite and deployed to the SOA server.

This section contains the following topics:

- [Setting an Application Server Connection in JDeveloper](#)
- [Setting Up the SOA Composite in JDeveloper](#)
- [Updating the SOA Composite](#)
- [Deploying the SOA Composite](#)
- [Testing the Setup](#)

26.2.1 Setting an Application Server Connection in JDeveloper

Ensure that a new application server connection, which represents the application server on which Oracle Identity Manager is installed, is first setup in JDeveloper . Make sure that the WebLogic Administrative Server and the SOA server are running before performing these steps.

To set up the new application server connection:

1. From the File menu, select **New**. The New Gallery dialog box is displayed.
2. From the left menu, select **All Items**. On the right pane, select **Application Server Connection**, and then click **OK**. The Create Application Server Connection wizard is displayed.
3. In the Name and Type window, enter a name that will identify the application server in JDeveloper. Select **Weblogic 10.3** as the connection type, and then click **Next**.
4. In the Authentication window, provide the username and password of the WebLogic user. Click **Next**.
5. In the Configuration window, enter the host name, port number, and the WebLogic domain name in which the SOA managed server is running. The port must be the WebLogic Administrative Server port (usually 7001). Click **Next**.
6. In the Test window, click **Test Connection** to make sure all the information entered is correct. The test passes with success status. Click **Next**.
7. Click **Finish** to exit the wizard. This creates the connection to the application server. This connection is required to deploy the composite to the server after making all changes.

26.2.2 Setting Up the SOA Composite in JDeveloper

To set up the SOA composite in JDeveloper for editing:

1. Copy the DefaultRequestApproval.zip file from the `OIM_HOME/workflows/composites/` directory to your JDeveloper working directory. Unzip it in the same directory to create the DefaultRequestApproval directory.
2. Start JDeveloper in the Default Role.
3. From the File menu, select **Open**. The file-open dialog box is displayed. Select the **DefaultRequestApproval.jpr** file in the DefaultRequestApproval directory. This opens the composite in JDeveloper. Click **OK** or **Yes** while the project file is created.
4. To successfully compile the Java code that you want to write in the composite, the `oimclient.jar` file must be in the JDeveloper copy of the composite. Copy the `oimclient.jar` file from the `OIM_HOME/server/client/` directory to the `JDEVELOPER_WORKING_DIRECTORY/DefaultRequestApproval/SCA-INF/lib/` directory. This directory is the `lib/` directory of the composite that you are editing.

See Also: "Deploying a Single SOA Composite in Oracle JDeveloper" in the *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite* for more information about setting up the SOA composite in JDeveloper

26.2.3 Updating the SOA Composite

To edit the SOA composite in JDeveloper:

1. In the left pane, click the **Projects** tab.
2. Select the **ApprovalProcess.bpel** file under the DefaultRequestApproval project and open it. This displays the approval workflow.
3. In the right pane, in the Component Palette, select the **Java Embedding** task, and drag and drop it after the receiveInput task in the workflow, before the ApprovalTask_1 human task. This creates a new task called Java_Embedding_1. Optionally, you can rename it to Invoke_OIM_API.
4. Double-click the **Invoke_OIM_API** Java task. This opens an editor in which you can add the required Java code. Add the Java code as shown in [Example 26–1](#):

Example 26–1 Embedded Java Source Code

```
try {
    System.out.println("Prototype for invoking an OIM API from a SOA
Composite");
    System.out.println("RTM Usecase: Organization Administrator");
    String oimUserName = "";
    String oimPassword = "";
    String oimURL = "";
    String roleApprover = "";
    String actKey = "";

    //get oimuser credentials
    oracle.security.jps.JpsContext ctx =

oracle.security.jps.JpsContextFactory.getContextFactory().getContext();

    final oracle.security.jps.service.credstore.CredentialStore cs =
        (oracle.security.jps.service.credstore.CredentialStore)

ctx.getServiceInstance(oracle.security.jps.service.credstore.CredentialStore.class
);

    oracle.security.jps.service.credstore.CredentialMap cmap =
        cs.getCredentialMap("oracle.oim.sysadminMap ");

    oracle.security.jps.service.credstore.Credential cred =
cmap.getCredential("sysadmin");

    if (cred instanceof
oracle.security.jps.service.credstore.PasswordCredential) {
        oracle.security.jps.service.credstore.PasswordCredential pcred =
(oracle.security.jps.service.credstore.PasswordCredential)cred;
        char[] p = pcred.getPassword();
        oimUserName = pcred.getName();
        oimPassword = new String(p);
    }

    //get oimurl
    Object obj = getVariableData("oimurl");
    oimURL = obj.toString();

    // set the initial context factory
    String oimInitialContextFactory = "weblogic.jndi.WLInitialContextFactory";
```

```

// set up the environment for making the OIM API invocation
java.util.Hashtable env = new java.util.Hashtable();

env.put(oracle.iam.platform.OIMClient.JAVA_NAMING_FACTORY_INITIAL,
    oimInitialContextFactory);

env.put(oracle.iam.platform.OIMClient.JAVA_NAMING_PROVIDER_URL, oimURL);
oracle.iam.platform.OIMClient client = new
oracle.iam.platform.OIMClient(env);
client.login(oimUserName, oimPassword.toCharArray());
System.out.println("Login Successful");

// get Service
oracle.iam.request.api.RequestService reqSvc =
    client.getService(oracle.iam.request.api.RequestService.class);

oracle.iam.identity.rolemgmt.api.RoleManager roleSvc =
client.getService(oracle.iam.identity.rolemgmt.api.RoleManager.class);

oracle.iam.identity.usermgmt.api.UserManager usersvc =
client.getService(oracle.iam.identity.usermgmt.api.UserManager.class);

Object reqIdXMLElement = getVariableData("inputVariable", "payload",
    "/ns3:process/ns4:RequestID");
String reqId = ((oracle.xml.parser.v2.XMLElement)reqIdXMLElement).getText();
System.out.println("The request ID is "+reqId);

// invoke the getBasicRequestData() method on the RequestService API
oracle.iam.request.vo.Request req = reqSvc.getBasicRequestData(reqId);
java.util.List<oracle.iam.request.vo.Beneficiary> beneficiaries =
req.getBeneficiaries();

if(beneficiaries != null){
    for(oracle.iam.request.vo.Beneficiary benf: beneficiaries){

        //get org key
        java.util.HashSet<String> searchAttrs = new
java.util.HashSet<String>();
        searchAttrs.add(oracle.iam.identity.usermgmt.api.UserManagerConstants.
            AttributeName.USER_LOGIN.getId());

        searchAttrs.add(oracle.iam.identity.usermgmt.api.UserManagerConstants.
            AttributeName.USER_ORGANIZATION.getId());

        oracle.iam.identity.usermgmt.vo.User user1 =
            usersvc.getDetails(benf.getBeneficiaryKey(), searchAttrs,
false);

        actKey = user1.getAttribute("act_key").toString();

        //get org admin
        if(actKey != "" && actKey != " ") {
            Thor.API.Operations.tcOrganizationOperationsIntf orgAPI =
(Thor.API.Operations.tcOrganizationOperationsIntf)client.getService(
            Thor.API.Operations.tcOrganizationOperationsIntf.class);

```

```

Thor.API.tcResultSet rset =
    orgAPI.getAdministrators(Long.parseLong(actKey));

StringBuffer sb = new StringBuffer();

for (int i = 0; i < rset.getRowCount();i++){
    rset.goToRow(i);
    sb.append(rset.getStringValue("Groups.Group Name")) ;
    if(i >= 0 && i < (rset.getRowCount()-1)){
        sb.append(",");
    }
}
String grpNames = sb.toString();
System.out.println("Groups="+grpNames);
setVariableData("orgAdmin",grpNames);
}

//get role approver
java.util.List<oracle.iam.request.vo.RequestBeneficiaryEntity> rbes =
benf.getTargetEntities();

for(oracle.iam.request.vo.RequestBeneficiaryEntity rbe : rbes){
    String key = rbe.getEntityKey();
    String type = rbe.getEntityType();

    if(type.equalsIgnoreCase("Role")){
        java.util.HashSet<String> roleAttrs = new
            java.util.HashSet<String>();
        roleAttrs.add("Role Approver");
        oracle.iam.identity.rolemgmt.vo.Role role =
            roleSvc.getDetails(key,roleAttrs);
        roleApprover = (String)role.getAttribute("Role Approver");
        setVariableData("roleApprover", roleApprover);
        break;
    }
}
break;
}
}
System.out.println("OrgAdmin=" + getVariableData("orgAdmin").toString());
System.out.println("roleApprover=" +
getVariableData("roleApprover").toString());
}

catch (Exception e){
System.out.println("-----");
e.printStackTrace();
System.out.println("-----");
}
}

```

In [Example 26–1](#), to retrieve the organization administrator by using Oracle Identity Manager APIs, the following is performed in the Java code:

a. Get credentials for the system administrator.

Credentials of the system administrator are stored in a credential store (cwallet). First, the credential store, then the credential map, and then the credential by using the key are retrieved. This is shown in the following code snippet:

```
//get Credential store
oracle.security.jps.JpsContext ctx =

oracle.security.jps.JpsContextFactory.getContextFactory().getContext();
    final oracle.security.jps.service.credstore.CredentialStore cs =
(oracle.security.jps.service.credstore.CredentialStore)ctx.getServiceInstance(
oracle.security.jps.service.credstore.CredentialStore.class);

//get Credential
oracle.security.jps.service.credstore.CredentialMap cmap =
    cs.getCredentialMap("oracle.oim.sysadminMap");
oracle.security.jps.service.credstore.Credential cred =
cmap.getCredential("sysadmin");
```

b. Login as system administrator.

The environment is setup, and then logged in to Oracle Identity Manager as system administrator. This is shown in the following code snippet:

```
//setup the environment
String oimInitialContextFactory = "weblogic.jndi.WLInitialContextFactory";
java.util.Hashtable env = new java.util.Hashtable();
env.put(oracle.iam.platform.OIMClient.JAVA_NAMING_FACTORY_INITIAL,
    oimInitialContextFactory);
env.put(oracle.iam.platform.OIMClient.JAVA_NAMING_PROVIDER_URL, oimURL);

//login to OIM
oracle.iam.platform.OIMClient client = new
oracle.iam.platform.OIMClient(env);
client.login(oimUserName, oimPassword.toCharArray());
    System.out.println("Login Successful");
```

c. Retrieve the organization administrator and the role approver.

This is done by using the following Oracle Identity Manager APIs:

- * **oracle.iam.request.api.RequestService:** Used to retrieve the request object
- * **Thor.API.Operations.tcOrganizationOperationsIntf:** Used to retrieve the organization administrator
- * **oracle.iam.identity.usermgmt.api.UserManager:** Used to retrieve the role approver

See Also: *Oracle Fusion Middleware Java API Reference for Oracle Identity Manager* for information about Oracle Identity Manager APIs

26.2.4 Deploying the SOA Composite

Before deploying the SOA composite, set the BPELClasspath property in the System MBean Browser of the Enterprise Manager.

After updating the composite, you must deploy the composite to SOA. To do so:

1. In the Projects section, right-click the composite name, and select **Deploy**.
2. Select the **DefaultRequestApproval ...** option. A wizard is displayed that prompts you to select the application server on which the composite is to be deployed. Make sure you select the application server connection created earlier. In addition, select the **Override any existing composites with the same revision ID** option if you want to override the composite that is already deployed.

After deploying the composite, either re-register the composite or disable and enable the composite from the Oracle Identity Manager side. This is to ensure that Oracle Identity Manager is able to invoke the composite correctly.

See Also: "Deploying a Single SOA Composite in Oracle JDeveloper" in the *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite* for more information about deploying the SOA composite

26.2.5 Testing the Setup

After the SOA composite is updated, you can test the changes to the composites to make sure that the Oracle Identity Manager API is being loaded. The Java code runs as soon as the Request Approval is started in SOA because the code is added to the SOA composite and before the human task in the workflow.

To create a request and test the Java code:

1. Login to Oracle Identity Manager Administrative and User Console.
2. Click **Advanced** to go to Advanced Administration.
3. In the Welcome page, under Administration, click **Requests**. Alternatively, click the **Administration** tab, and then click **Requests**.
4. From the Actions menu, select **Create Request**. Alternatively, click the Create Request icon on the toolbar. The Request Creation wizard is displayed.

Note:

- This must be performed in the test environment.
 - Make sure that no approval polices are associated with the Create User request type.
-
-

5. From the Type of Request list, select **Create User**. Then, click **Next**.
6. In the Enter Details page, enter sample values in the fields to create the user. Then, click **Next**.
7. In the Confirm page, click **Finish**.
8. Monitor the SOA server console for output from the Java code that you have embedded. Clicking Finish runs the SOA composite. The following text is displayed in the SOA server console:

```
Prototype for invoking an Oracle Identity Manager API from a SOA Composite
Login Successful
<Request ID and other request data>
```

This output is displayed if the code is successfully run.

Part VI

Segregation of Duties

This part contains a chapter describing segregation of duties (SoD).

It contains the following chapter:

- [Chapter 27, "Using Segregation of Duties \(SoD\)"](#)

Using Segregation of Duties (SoD)

The concept of Segregation of Duties (SoD) is aimed at applying checks and balances on business processes. Each stage of a business process may require the involvement of more than one individual. An organization can convert this possibility into a requirement for all IT-enabled business processes by implementing SoD as part of its user provisioning solution. The overall benefit of SoD is the mitigation of risk arising from intentional or accidental misuse of an organization's resources. This chapter contains the following sections:

- [Understanding the SoD Validation Process](#)
- [Introducing the SoD Invocation Library](#)
- [Installing the SoD-enabled Connectors](#)
- [Deploying the SIL and SIL Providers](#)
- [Configuring the SoD Engine](#)
- [Enabling and Disabling SoD](#)
- [Enabling SSL Communication](#)
- [Configuring Workflows on Non SoD-enabled Connectors](#)
- [Marking Fields as Entitlements](#)
- [Custom Combination of Target Systems and SoD Engines](#)
- [Performing Role SoD Check with Oracle Identity Analytics](#)
- [Using SoD in Provisioning Workflow](#)
- [Enabling Logging for SoD-Related Events](#)
- [Troubleshooting SoD Check](#)

27.1 Understanding the SoD Validation Process

Oracle Identity Manager is a user provisioning solution with which entitlement requests can also be validated and managed. In the Oracle Identity Manager implementation of SoD, user requests for IT privileges (entitlements) are checked and approved by an SoD engine and other users. Multiple levels of system and human checks ensure that even changes to the original request are vetted before the request is cleared. This preventive approach helps identify and correct potentially conflicting entitlement assignments *before* the requested entitlements are assigned.

The SoD validation process in Oracle Identity Manager occurs when a user creates a request for an entitlement on a particular target system. The request is funneled

through a resource approval workflow and, if it passes that initial workflow, a resource provisioning workflow.

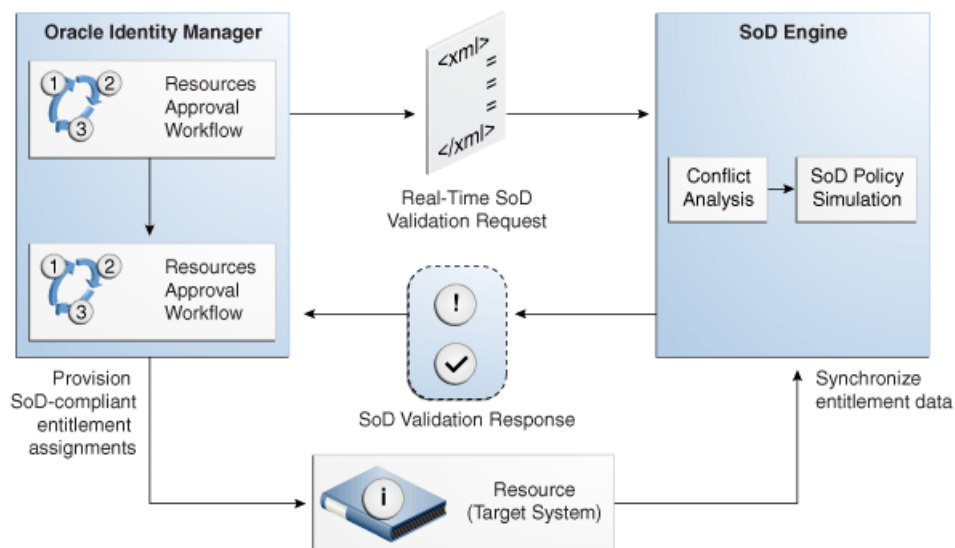
- The *resource approval workflow* is configured to validate requests in real time using an SoD engine. The SoD engine has predefined rules that are used to determine if the entitlement assignment would lead to SoD violations. The determination, once made, is returned to Oracle Identity Manager.
- The *resource provisioning workflow* provisions an entitlement request that has passed the resource approval workflow on the target system.

If the user's request passes SoD validation (and an approver approves the request), the resource provisioning workflow is initiated. If the request fails SoD validation, the resource approval workflow can be configured to take remediation steps.

Note: The resource provisioning workflow can be configured to perform the SoD validation again - immediately before the entitlement assignment is provisioned to the target system - to ensure SoD compliance.

Oracle Identity Manager communicates with both the SoD engine and the target system. In addition, the target system and SoD engine communicate with each other to enable the synchronization of entitlement data. Figure 27-1 shows the flow of data during the SoD validation process.

Figure 27-1 SoD Validation Process in Oracle Identity Manager



27.2 Introducing the SoD Invocation Library

The SoD Invocation Library (SIL) forms the basis of the SoD implementation in Oracle Identity Manager. The SIL is a collection of Java-based adapters that enable integration with predefined Oracle Identity Manager connectors. The connectors, in turn, link Oracle Identity Manager with the target systems. The following Oracle Identity Manager connectors are preconfigured with adapters for SoD validation:

- Oracle e-Business User Management release 9.1.0 and later
- SAP User Management release 9.1.0 and later

The SIL also acts as the base for specialized adapters that integrate the SIL with SoD engines. These adapters are called SIL providers. A SIL provider acts as the interface between the SIL and a specific SoD engine. There are predefined SIL providers for the following SoD engines:

- SIL Provider for SAP GRC

This provider is also known as the SAP GRC SIL Provider. The certified versions of SAP GRC are versions 5.2 SP4 or later and 5.3 SP5 or later.

- SIL Provider for Oracle Application Access Controls Governor (OAACG) release 8.2.1 or later

This provider is also known as the OAACG SIL Provider.

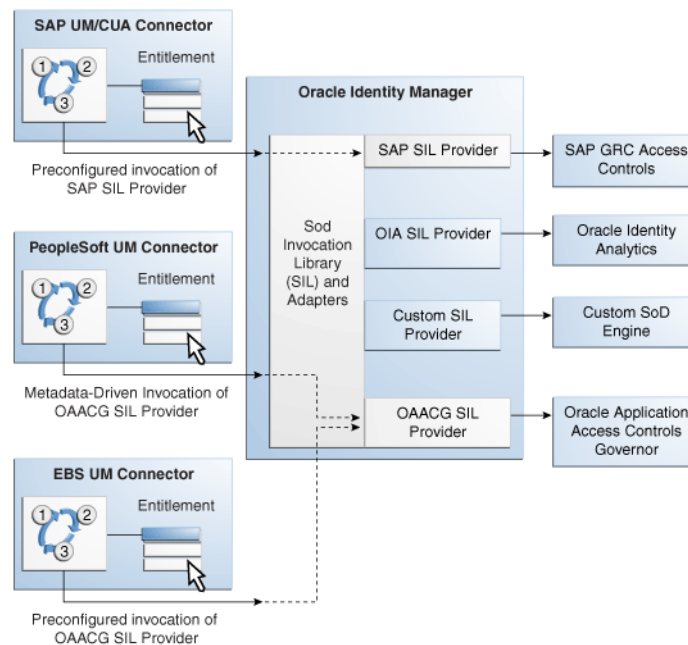
Note: Install the latest patch set for OAACG before implementing and using SoD in Oracle Identity Manager. Contact Oracle Support for more information.

- SIL Provider for Oracle Identity Analytics (OIA) release 11.1.1.3. or higher

This provider is also known as the OIA SIL Provider.

Figure 27–2 shows the architecture of SoD implementation in Oracle Identity Manager.

Figure 27–2 Architecture of SoD Implementation in Oracle Identity Manager



If required, you can configure any Oracle Identity Manager connector with either the SAP GRC SIL Provider, the OAACG SIL Provider or the OIA SIL Provider. For example, you can use the PeopleSoft User Management connector and the OAACG SIL Provider to automate SoD validation of requests for entitlements on PeopleSoft Enterprise Applications.

You can also create and use a SIL provider for a custom SoD engine, along with either one of the preconfigured Oracle Identity Manager connectors or an Oracle Identity Manager connector that you configure for SoD validation.

27.3 Installing the SoD-enabled Connectors

Instructions to install the SoD-enabled connectors listed below can be found in the specific connector documentation. The Oracle Identity Manager Connectors Documentation page is located at the following UR:

http://download.oracle.com/docs/cd/E11223_01/index.htm

- Oracle e-Business User Management release 9.1.0 and later
- SAP User Management release 9.1.0 and later

27.4 Deploying the SIL and SIL Providers

SIL registration is provided by default for the some target systems and SoD engines. No deployment steps are required for these default combinations of target systems and SoD engines. Target systems for which SIL registration is provided include:

- EBS and OAACG
- PSFT and OAACG
- SAP and SAP-GRC
- OIA

OIA SoD Engine synchronizes data with Oracle Identity Manager rather than any target system so the topology registered for OIA can be used with any connector configured with Oracle Identity Manager. OIA imports all data from Oracle Identity Manager. Therefore, from OIA perspective, Oracle Identity Manager is the target system.

You must perform the SIL registration process if you want to use any other combination of target systems or SoD engines. For more information, see [Section 27.10, "Custom Combination of Target Systems and SoD Engines."](#)

27.5 Configuring the SoD Engine

You must import entitlement data from the target system to the SoD engine. If required, you must also configure SoD validation rules on the SoD engine. The following sections provide these instructions for the preconfigured SoD engines.

- [Configuring Oracle Application Access Controls Governor](#)
- [Configuring SAP GRC](#)
- [Configuring Oracle Identity Analytics](#)

27.5.1 Configuring Oracle Application Access Controls Governor

Configuring Oracle Application Access Controls Governor (OAACG) involves the following procedures:

- [Installing Oracle Application Access Controls Governor](#)
- [Creating an Oracle Application Access Controls Governor Account for SoD Operations](#)

- [Synchronizing Role and Responsibility Data from Oracle e-Business Suite to Oracle Application Access Controls Governor](#)
- [Defining Access Policies in Oracle Application Access Controls Governor](#)

Installing Oracle Application Access Controls Governor

OAACG 8.6.x is supported with Oracle Identity Manager 11g Release 1 (11.1.1.4) onward. OAACG 8.6.0.203 is the recommended version that must be installed. Further, this must be upgraded to OAACG 8.6.0.219 or OAACG 8.6.0.240.

To install OAACG 8.6.0.203:

1. Logon to My Oracle Support.
2. Click the **Patches & Updates** tab.
3. Click **Advanced Search**.
4. Select Product Family as **Oracle Application Access Controls Governor** and release as **AAACG 8.6.0**. Select the appropriate platform, and click **Search**.
5. Select latest patch. See the Oracle Identity Manager Bundle Patch Readme to confirm if this is 8.6.0.219 or 8.6.0.240.

Note: Oracle Identity Manager SoD has been certified against OAACG 8.6.0.219 and OAACG 8.6.0.240.

6. Download the patch or update.
7. Perform the OAACG upgrade by referring to the OAACG upgrade guide.

OAACG 8.6.0.203 must be upgraded to OAACG 8.6.0.219 or OAACG 8.6.0.240. To do so:

1. Logon to My Oracle Support.
2. Click the **Patches & Updates** tab.
3. Search for the Patch ID.
4. Select the Patch ID.
5. Download the patch or update.
6. Perform the OAACG upgrade by referring to the OAACG upgrade guide.

Creating an Oracle Application Access Controls Governor Account for SoD Operations

Create an account of the Basic type for SoD validation operations. While performing the procedure described in "[Creating an IT Resource to Hold Information about the SoD Engine](#)" on page 27-43, provide the user name and password of this account.

See Oracle Application Access Controls Governor documentation for information about creating the account.

Synchronizing Role and Responsibility Data from Oracle e-Business Suite to Oracle Application Access Controls Governor

You must import (synchronize) role and responsibility data from Oracle e-Business Suite into Oracle Application Access Controls Governor. After first-time synchronization, you must schedule periodic synchronization of data.

See Oracle Application Access Controls Governor documentation for more information.

Defining Access Policies in Oracle Application Access Controls Governor

After you import role and responsibility data, set up access policies in Oracle Application Access Controls Governor. These access policies are based on various combinations of roles and responsibilities.

See Oracle Application Access Controls Governor documentation for more information.

27.5.2 Configuring SAP GRC

SAP GRC uses user, role, and profile data from SAP R/3 to validate requests for accounts, roles, and responsibilities. Configuring SAP GRC involves the following procedures:

- [Creating an SAP GRC Account for SoD Operations](#)
- [Generating the Keystore](#)
- [Configuring the Risk Terminator](#)
- [Synchronizing User, Role, and Profile Data from SAP ERP to SAP GRC](#)
- [Defining Risk Policies in SAP GRC](#)

Creating an SAP GRC Account for SoD Operations

You must create an SAP GRC account for SoD operations. During SoD operations, this account is used to call the SAP GRC Web service.

When you create this user account, you must assign it to the following groups:

- Everyone
- Authenticated Users

You must not assign any roles to this account.

Generating the Keystore

To generate the keystore:

1. In a Web browser, open the Web Services Navigator page of SAP GRC Access Control. The URL is similar to the following:

```
https://SAP_GRC_HOST:PORT_NUMBER/VirsaCCRiskAnalysisService/Config1?wsdl
```
2. Export the certificate.
3. Copy the certificate into the bin directory inside the JDK installation directory of SAP GRC.
4. Run the following command to create the keystore from the certificate file that you download:

```
keytool -import -v -trustcacerts -alias sapgrc -file CERTIFICATE_FILENAME  
-keystore sgil.keystore -keypass changeit -storepass changeit
```

Note: In this sample command, the keystore file name is sgil.keystore.

5. When prompted for the keystore password, specify `changeit`. This is the default keystore password.
6. When prompted to specify whether you want to trust the certificate, enter `yes`.
7. The `sgil.keystore` file is created in the `bin` directory. Copy the file to the `OIM_HOME/config` directory.

Configuring the Risk Terminator

The Risk Terminator is a feature of GRC Access Control. It is the main component of the SoD validation functionality of SAP GRC. Whenever a role is created in the profile generator or assigned to a user, the Risk Terminator verifies if this role creation or assignment would result in an SoD violation.

See the Risk Terminator Configuration document for detailed information.

Synchronizing User, Role, and Profile Data from SAP ERP to SAP GRC

User, role, and profile data must be imported (synchronized) from SAP ERP into SAP GRC. After first-time synchronization, you must schedule periodic synchronization of data.

Defining Risk Policies in SAP GRC

After you import role and responsibility data, use the Risk Analysis and Remediation feature of SAP GRC to define risk policies of type Segregation of Duty.

See SAP GRC documentation for more information.

27.5.3 Configuring Oracle Identity Analytics

Configuring Oracle Identity Analytics involves the following procedures:

- [Creating an Oracle Identity Analytics Account for SoD Operations](#)
- [Synchronizing Oracle Identity Manager Metadata With Oracle Identity Analytics](#)
- [Defining Identity Audit Policies in Oracle Identity Analytics](#)

Creating an Oracle Identity Analytics Account for SoD Operations

Create an account on Oracle Identity Analytics and assign to it the SRM Admin role for SoD validation operations. When performing the procedure described in "[Creating an IT Resource to Hold Information about the SoD Engine](#)" on page 27-43, provide the user name and password of this account.

See the Oracle Identity Analytics documentation for information about creating the account.

Note: The Oracle Identity Analytics Admin account with username `rbacxadmin` can also be used.

Synchronizing Oracle Identity Manager Metadata With Oracle Identity Analytics

Import the resource metadata and resources from Oracle Identity Manager to Oracle Identity Analytics.

See the Oracle Identity Analytics documentation for more information.

Defining Identity Audit Policies in Oracle Identity Analytics

Set up identity audit rules and policies using Oracle Identity Analytics. Rules are created on the resource attributes. For entitlement SoD Check, give encoded values for roles and responsibilities as in Oracle Identity Manager.

See the Oracle Identity Analytics documentation for more information.

27.6 Enabling and Disabling SoD

The following sections contain information on enabling and disabling SoD.

- [Enabling SoD](#)
- [Disabling SoD](#)

27.6.1 Enabling SoD

To enable the SoD feature:

1. Set the Segregation of Duties (SOD) Check Required system property to `true`. See "Administering System Properties" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager* for information about system properties.
2. Set the `topologyName` parameter in the Connector IT Resource instance to the value present in `SILConfig.xml`. If you are using default SIL registration, set the `topologyName` parameter in connector IT Resource to one of the following:
 - `sodoaacg` if you are using the EBS connector and OAACG as the SoD engine
 - `oaacgpsft` if you are using the PSFT connector and OAACG as the SoD engine
 - `sodgrc` if you are using GRC as the SoD engine
 - `sodoia` if you are using OIA as the SoD engine

Note: Connector IT resource must have the ALL USERS role so that any user is able to access the IT Resource information. This is required for SOD requests raised by users.

3. Deploying SIL and SIL Providers

To deploy SIL and SIL providers for default combination of target systems and SoD engines:

- a. Create a new IT Resource for Sod Engine with the name (type) as follows:
 - For EBS-OAACG: OAACG-ITRes (eBusiness Suite OAACG)
 - For SAP-GRC: GRC-ITRes (SoD Provider)
 - For OIA: IT resource with name OIA-ITRes (OIA) is predefined.
 - For PSFT-OAACG: IT resource with name PSFT-OAACG-ITRes(OAACG) is predefined.
- b. Edit the created IT Resource as described in "[Creating an IT Resource to Hold Information about the SoD Engine](#)" on page 27-43.

Note: ■ To configure with OAACG8.5, add a new field to this IT resource with name as `sodServerURL` and value `http(s)://HOST_NAME:PORT/URI`, where `URI` is `grcc/services/GrccService`. For OAACG8.2.1, the value of `URI` is `ags/services/AGService`.

- See "Administering System Properties" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager* for information about how to set system property values.
-
-

4. Enabling SoD in Direct Provisioning and Access Policy Based Provisioning:

SoD is enabled only if Holder and SODChecker tasks are present in the provisioning workflow.

Enabling SoD in Request Provisioning:

Steps 1 and 2 enables default SoD check in approval. The default SoD check is performed before the request goes for approval. If the SoD check is required after one level of approval, then default SoD check approval workflow, which is `DefaultSODApproval`, must be attached by creating an approval policy. SoD check can also be performed in any approval workflow on demand. This can be done by calling the SoD check Web service from BPEL. For more information, see "[Modifying the Approval Workflow for SoD](#)" on page 27-13.

Note: If `DefaultSODApproval` workflow is attached to operational level of approval, then the system administrator first needs to approve, and then only SoD Check is performed. After SoD Check is performed, approval is required by SoD Administrators role. According to this workflow, first approval task is generated that is assigned to the system administrator, then SoD Check is performed, and then an approval task is generated. Any user who is a member of the SoD Administrators role can approve the second approval task after viewing the SoD Check results.

5. Adding CSF Credentials for SoD Check:

- a. Login to the Enterprise Manager console and on the left tab, expand **Weblogic Domain**.
- b. Open **base_domain**.
- c. On top of the right pane, from the WebLogic Domain list, select **Security**, and then open **Credentials**.
- d. Select the **Create Key** option, and then select **Map 'oim'**.
- e. Provide the key as `sodcheck.credentials`, and select Type as **Password**.
- f. Provide Username as **oiminternal** and password as **not used**. Click **OK** to save the key.

27.6.2 Disabling SoD

You can disable SoD by performing any one of the following:

- Set the Segregation of Duties (SOD) Check Required system property to `false`.

- Remove the value of the topologyName parameter for the connector IT Resource so that its value is set to blank. If the topologyName parameter in ITResource is set to None, then SoD check is not performed.

Disabling SoD in Direct Provisioning and Access Policy Based Provisioning

Disable the Holder and SODChecker process tasks.

See Also: The connector guide for detailed information about disabling these process tasks.

Disabling SoD in Request Provisioning

For disabling the default SoD check in approval, you can perform any one of the steps to disable SoD. If you want to perform the default SoD check in approval and only disable the SoD check in BPEL, then remove approval policy for SoD or remove call to SoD Check Web service from the approval workflow.

27.7 Enabling SSL Communication

The following sections contain information on enabling Secure Sockets Layer (SSL) communication for various SoD purposes.

- [Enabling SSL Between Oracle Application Access Controls Governor and Oracle Identity Manager](#)
- [Enabling SSL Between SAP GRC and Oracle Identity Manager](#)
- [Calling SoD Check Web Service Over SSL](#)

27.7.1 Enabling SSL Between Oracle Application Access Controls Governor and Oracle Identity Manager

To enable SSL communication between Oracle Application Access Controls Governor and Oracle Identity Manager:

Note: It is assumed that you have set sslEnable to true during the registration process.

1. Export the certificate on the Oracle Application Access Controls Governor host computer as follows:

Note: In Step 1, *JAVA_HOME* refers to the directory on the Oracle Application Access Controls Governor host computer.

- a. Run the following commands from the *JAVA_HOME*/bin directory:

```
keytool -genkey -alias tomcat -keyalg RSA -keystore
JAVA_HOME/lib/security/.keystore
keytool -certreq -alias tomcat -file JAVA_HOME/lib/security/xell.cvs
-keystore JAVA_HOME/lib/security/.keystore
keytool -export -alias tomcat -file JAVA_HOME/lib/security/server.cert
-keystore JAVA_HOME/lib/security/.keystore
```

After you run these commands, the server certificate (server.cert) is created in the *JAVA_HOME*/lib/security directory.

- b. In the `TOMCAT_HOME/conf/server.xml` file, enter the details of the keystore as attributes of the Connector element. See the following example:

```
<Connector port="8443" maxHttpHeaderSize="8192"
maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
enableLookups="false" disableUploadTimeout="true"
acceptCount="100" scheme="https" secure="true"
clientAuth="false" sslProtocol="TLS"
keystoreFile="JAVA_HOME/lib/security/.keystore">
```

- c. Restart Oracle Application Access Controls Governor.
2. Import the certificate on the Oracle Identity Manager host computer as follows:

Note: In Step 2, `JAVA_HOME` refers to the directory on the Oracle Identity Manager host computer.

- a. Copy the server certificate created on the Oracle Application Access Controls Governor host computer to the `JAVA_HOME/lib/security` directory of the Oracle Identity Manager host computer.
- b. Run the following command from the `JAVA_HOME/bin` directory:

```
keytool -import -alias oaacg_trusted_cert -file
JAVA_HOME/lib/security/server.cert -trustcacerts -keystore
JAVA_HOME/lib/security/cacerts -storepass changeit
```

27.7.2 Enabling SSL Between SAP GRC and Oracle Identity Manager

To enable SSL communication between SAP GRC and Oracle Identity Manager export the certificate on the SAP GRC host computer as follows:

Note: In this section, `JAVA_HOME` refers to the directory on the Oracle Identity Manager host computer that is used to run the application server.

1. In a Web browser, open the Web Services Navigator page of SAP GRC Access Control. The URL is similar to the following:

```
https://mysapserver01:50001/VirsaCCRiskAnalysisService/Config1?wsdl
```
2. The next step depends on the browser that you are using:
 - On Microsoft Internet Explorer: In the Security Alert dialog box, click **View Certificate**. On the Details tab of the dialog box, use the **Copy to file** button to export the certificate.
 - On Mozilla Firefox: Export the certificate as a `.pem` file. To be able to perform this step, you might need to download and install the Certificate Viewer enhancement from the Mozilla Web site.
3. Copy the certificate into the `JAVA_HOME/lib/security` directory used by the application server hosting Oracle Identity Manager.
4. In a terminal window, change to the `JAVA_HOME/bin` directory.
5. Run the following command to import the GRC certificate to cacerts:

```
keytool -import -alias sapgrc_trusted_cert -file  
JAVA_HOME/lib/security/CERTIFICATE_FILENAME -trustcacerts -keystore  
JAVA_HOME/lib/security/cacerts -storepass changeit
```

In this command:

- *CERTIFICATE_FILENAME* is the name of certificate that has been exported from the SAP GRC host computer
 - The `-storepass changeit` clause specifies the password for the cacerts keystore.
6. When prompted to specify whether or not you want to trust the certificate, enter *yes*.
- The "Certificate was added to keystore" message is displayed.

27.7.3 Calling SoD Check Web Service Over SSL

SOA calls the Oracle Identity Manager Web service over the URL given as the `oimFrontEndURL`, which is the URL used to access the Oracle Identity Manager UI, in the `oim-config.xml` file. By default, this is a HTTP URL. You can change this to HTTPS so that communication takes place over SSL. The SSL port for Oracle Identity Manager can be viewed on the WebLogic Administrative Console.

To call SoD check Web service over SSL:

1. Locate the Oracle Identity Manager SSL port. To do so:
 - a. Login to the WebLogic Administrative Console.
 - b. Go to **servers, oim_server1**. You can see that SSL Listen Port is enabled.
2. Change the `oimFrontEndURL` through the MBeans browser in Enterprise Manager. To do so:
 - a. Login to Enterprise Manager.
 - b. Go to **oim_server1**.
 - c. From the list on the top of the page, select **System Mbeans Browser**.
 - d. Go to **Application Defined Mbeans, oracle.iam, Server: oim_server1, Application: oim, XMLConfig, Config, XMLConfig.DiscoveryConfig, and Discovery**. The attributes are displayed to the right.
 - e. Click **oimFrontEndURL**, and change its value, as shown:

`https://HOST_NAME:SSL_PORT`

Note: The value of `oimFrontEndURL` can also be set at the time of installing Oracle Identity Manager.

3. Restart Oracle Identity Manager.
4. Create a request for SoD-enabled resource. You can view the new workflow instance in Enterprise Manager. The Web service will be called on SSL port.

Note: It is assumed that Oracle Identity Manager and SOA are running on the same Java Runtime Environment (JRE). If SOA and Oracle Identity Manager are running on different JREs, then WebLogic certificate exchange is required for SSL communication. For details, see Oracle WebLogic Server 10g Release 3 (10.3) documentation in the Oracle Technology Network (OTN) Web site by using the following URL:

<http://www.oracle.com/technetwork/middleware/weblogic/documentation/index.html>

27.8 Configuring Workflows on Non SoD-enabled Connectors

Perform the procedures described in this section only if you are not using one of the preconfigured SoD-compatible connectors (Oracle e-Business User Management, SAP User Management, and SAP CUA). This section discusses the following procedures:

- [Modifying the Approval Workflow for SoD](#)
- [Modifying the Provisioning Workflow for SoD](#)

27.8.1 Modifying the Approval Workflow for SoD

To modify the approval workflow for SoD validation:

1. Instead of object forms in earlier releases of Oracle Identity Manager, the data to be entered while creating a request in 11g Release 1 (11.1.1) is entered in request datasets. If the request datasets are not already present for the target system resource to be provisioned, then create new parent and child datasets and import them to MDS by using the MDS Import Utility.

Note: ■ No SoD Check fields must be added in the request dataset. These fields are part of common dataset and is available by default irrespective of the connector being used. The SoD fields in the common dataset are

```
<!-- Common SoD check attributes used for Provision Resource
-->
<AttributeReference name="SoDCheckStatus"
attr-ref="SoDCheckStatus" length="50" type="String"
widget="text" read-only="true" available-in-bulk="false"
system-type="true"/>
<AttributeReference name="SoDCheckTrackingID"
attr-ref="SoDCheckTrackingID" length="50" type="String"
widget="text" read-only="true" available-in-bulk="false"
system-type="true"/>
<AttributeReference name="SoDCheckResult"
attr-ref="SoDCheckResult" length="4000" type="String"
widget="text" read-only="true" available-in-bulk="false"
system-type="true"/>
<AttributeReference name="SoDCheckTimestamp"
attr-ref="SoDCheckTimestamp" length="50" type="Date"
widget="text" read-only="true" available-in-bulk="false"
system-type="true"/>
<AttributeReference name="SoDCheckEntitlementViolation"
attr-ref="SoDCheckEntitlementViolation" length="4000"
type="String" widget="text" read-only="true"
available-in-bulk="false" system-type="true"/>
```

Here, the <attr-ref> tag values represent mapping to process form fields. Therefore, any connector enabled for SoD must have these specific form field labels in the parent process form. For example, SoDCheckStatus attribute value is mapped to parent process form field with label as SoDCheckStatus.

- For detailed information about request datasets, see ["Step 1: Creating a Request Dataset for the Resources"](#) on page 23-1.
 - For information about MDS Import/Export Utility, see [Chapter 33, "MDS Utilities and User Modifiable Metadata Files"](#).
 - Object forms in earlier releases of Oracle Identity Manager have been replaced by request datasets in 11g Release 1 (11.1.1). Therefore, although the object forms may be present in the connector, they are not used.
-

The following is a sample request dataset for the eBusiness Suite User TCA Foundation resource. Create an appropriate request dataset for the resource being used.

```
<request-data-set xmlns="http://www.oracle.com/schema/oim/request"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.oracle.com/schema/oim/request"
name="eBusiness Suite User TCA Foundation" entity="eBusiness Suite
User TCA Foundation" operation="PROVISION">

    <!-- Parent form having fields -->

    <AttributeReference name="Effective Date From" attr-ref="Effective Date
From" type="Date" length="50" widget="date" required="true"
```



```

available-in-bulk="true"/>
  <AttributeReference name="SSO GUID" attr-ref="SSO GUID" type="String"
length="256" widget="text" required="false" available-in-bulk="true"/>
  <AttributeReference name="Last Name" attr-ref="Last Name" type="String"
length="150" widget="text" required="false" available-in-bulk="true"/>
  <AttributeReference name="EBS Server" attr-ref="EBS Server"
type="String" length="50" widget="itresource-lookup" available-in-bulk="true"
itresource-type="eBusiness Suite UM" required="true"/>
  <AttributeReference name="Password Expiration Interval"
attr-ref="Password Expiration Interval" type="Long" length="50" widget="text"
required="false" available-in-bulk="true"/>
  <AttributeReference name="Fax" attr-ref="Fax" type="String" length="80"
widget="text" required="false" available-in-bulk="true"/>
  <AttributeReference name="Password" attr-ref="Password" type="String"
length="30" widget="text" masked="true" required="true"
available-in-bulk="true"/>
  <AttributeReference name="Effective Date To" attr-ref="Effective Date
To" type="Date" widget="date" length="50" required="false"
available-in-bulk="true"/>
  <AttributeReference name="Description" attr-ref="Description"
type="String" length="240" widget="text" required="false"
available-in-bulk="true"/>
  <AttributeReference name="Password Expiration Type" attr-ref="Password
Expiration Type" type="String" length="30" widget="lookup" required="false"
available-in-bulk="true" lookup-code="Lookup.EBS.PasswordExpirationType"/>
  <AttributeReference name="SSO User ID" attr-ref="SSO User ID"
type="String" length="256" widget="text" required="false"
available-in-bulk="true"/>
  <AttributeReference name="User Name" attr-ref="User Name" type="String"
length="100" widget="text" required="true" available-in-bulk="true"/>
  <AttributeReference name="First Name" attr-ref="First Name"
type="String" length="150" widget="text" required="false"
available-in-bulk="true"/>
  <AttributeReference name="Party ID" attr-ref="Party ID" type="Long"
widget="text" length="50" required="false" available-in-bulk="true"/>
  <AttributeReference name="User ID" attr-ref="User ID" type="Long"
widget="text" length="50" required="false" available-in-bulk="true"/>
  <AttributeReference name="Email" attr-ref="Email" type="String"
length="240" widget="text" required="false" available-in-bulk="true"/>

  <!-- Child form EBS -->
  <AttributeReference name="UD_EBST_RSP" attr-ref="UD_EBST_RSP"
type="String" length="20" widget="text" available-in-bulk="true">
  <AttributeReference name="Effective Start Date" attr-ref="Effective
Start Date" type="Date" length="50" widget="date" required="false"
available-in-bulk="true"/>
  <AttributeReference name="Effective End Date" attr-ref="Effective End
Date" type="Date" length="50" widget="date" required="false"
available-in-bulk="true"/>
  <AttributeReference name="Application Name" attr-ref="Application Name"
type="String" length="256" widget="lookup" required="false"
available-in-bulk="true" lookup-code="Lookup.EBS.Application"/>
  <AttributeReference name="Responsibility Name" attr-ref="Responsibility
Name" type="String" length="256" widget="lookup-query" available-in-bulk="true"
required="true">
  <lookupQuery lookup-query="select lkv_encoded as
lkv_encoded,lkv_decoded as lkv_decoded from lkv lkv,lku lku where
lkv.lku_key=lku.lku_key and
lku_type_string_key='Lookup.EBS.Responsibility' and

```

```

instr(lkv_encoded,concat('$Form data.Application Name','~'))>0"
display-field="lkv_decoded" save-field="lkv_encoded"/>
</AttributeReference>

</AttributeReference>

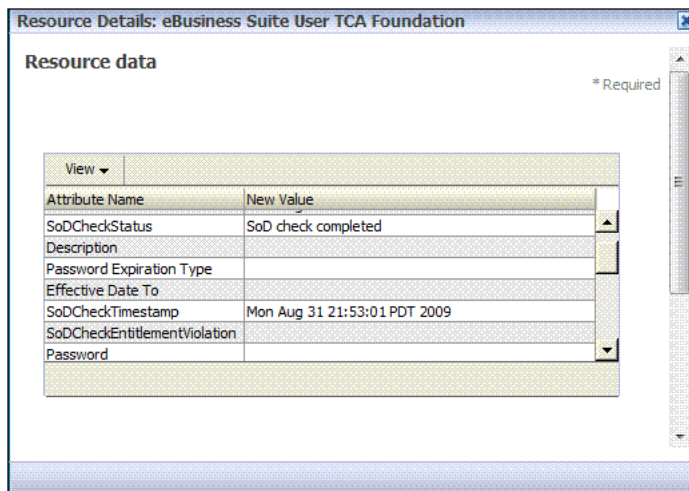
<AttributeReference name="UD_EBST_RLS" attr-ref="UD_EBST_RLS"
type="String" length="20" widget="text" available-in-bulk="true">
<AttributeReference name="Start Date" attr-ref="Start Date" type="Date"
widget="date" length="50" required="false" available-in-bulk="true"/>
<AttributeReference name="Expiration Date" attr-ref="Expiration Date"
type="Date" widget="date" length="50" required="false"
available-in-bulk="true"/>

<AttributeReference name="Application Name" attr-ref="Application Name"
type="String" length="256" widget="lookup" required="false"
available-in-bulk="true" lookup-code="Lookup.EBS.Application"/>
<AttributeReference name="Role Name" attr-ref="Role Name" type="String"
length="256" widget="lookup-query" available-in-bulk="true" required="true">
<lookupQuery lookup-query="select lkv_encoded as lkv_encoded,lkv_decoded as
lkv_decoded from lkv lkv,lku lku where lkv.lku_key=lku.lku_key and
lku_type_string_key='Lookup.EBS.UMX.Roles' and instr(lkv_encoded,concat('$Form
data.Application Name','~'))>0" display-field="lkv_decoded"
save-field="lkv_encoded"/>
</AttributeReference>
</AttributeReference>
</request-data-set>

```

Figure 27–3 shows the SoD Check attributes system attributes as displayed in the request details page after SoD Check is completed:

Figure 27–3 The SoDCheckAttributes System Attributes



2. In the IT resource of the connector, create the TopologyName parameter if it does not already exist. Figure 27–4 shows a sample IT resource in which this parameter has been added:

Figure 27–4 The TopologyName Parameter

Native connection pool class definition	
Pool excluded fields	Configuration Lookup Na
Pool preference	Default
ResourceConnection class definition	
Retry Interval	10000
SSL Enabled	No
SSO Enabled	No
SSO IT Resource	
SSO Identifier	
SSO Login Attribute	
Statement Timeout	1200
Target supports only one connection	False
Timeout check interval	30
TopologyName	sodoaacg
Validate connection on borrow	true

Update Cancel

If SoD Check property is set to true and topologyName parameter is set to the appropriate value in the connector IT Resource, then default SoD Check is performed in the preprocess stage of the approval workflow. After the request is created, the request status is changed to SoD check result pending for asynchronous SoD check and SoD check completed status for synchronous SoD check. For asynchronous SoD check, the Get SOD Check Results Approval scheduled job must be run to complete the SoD check.

Note: If there is an error while performing SoD check, then the SoD Check Status attribute in the request dataset is set to SoD check completed with error and the request moves for approval. Final decision is on the approver whether or not to approve the request although SoD check is not performed successfully.

Figure 27–5 shows the request history for asynchronous SOD check:

Figure 27–5 Request History for Asynchronous SoD Check

Request Information

Request ID	4	Beneficiary	
Request Type	Provision Resource	Justification	
Status	Request Completed	Parent Request ID	
Date Requested	8/31/09 8:41 PM		
Effective Date			
Requester	System Administrator		

Request History

Status	Updated On	Updated By
Request Created	8/31/09 8:41 PM	System Administrator
SoD check not initiated	8/31/09 8:41 PM	System Administrator
SoD check result pending	8/31/09 8:42 PM	System Administrator
SoD check completed	8/31/09 8:43 PM	System Administrator
Obtaining Request Approval	8/31/09 8:43 PM	System Administrator
Request Approval Approved	8/31/09 8:43 PM	System Administrator
Obtaining Operation Approval	8/31/09 8:43 PM	System Administrator

3. In addition to the SoD check being triggered by default before any level of approval, it can be triggered by attaching the predefined DefaultSODApproval workflow. The workflow can be attached to the operational level of approval by creating an approval policy.

Note: In Oracle Identity Manager 11g, patches were required on SOA and WebLogic Server to allow the SoD workflow to work. No patch is required with the release of Oracle Identity Manager 11g PS1.

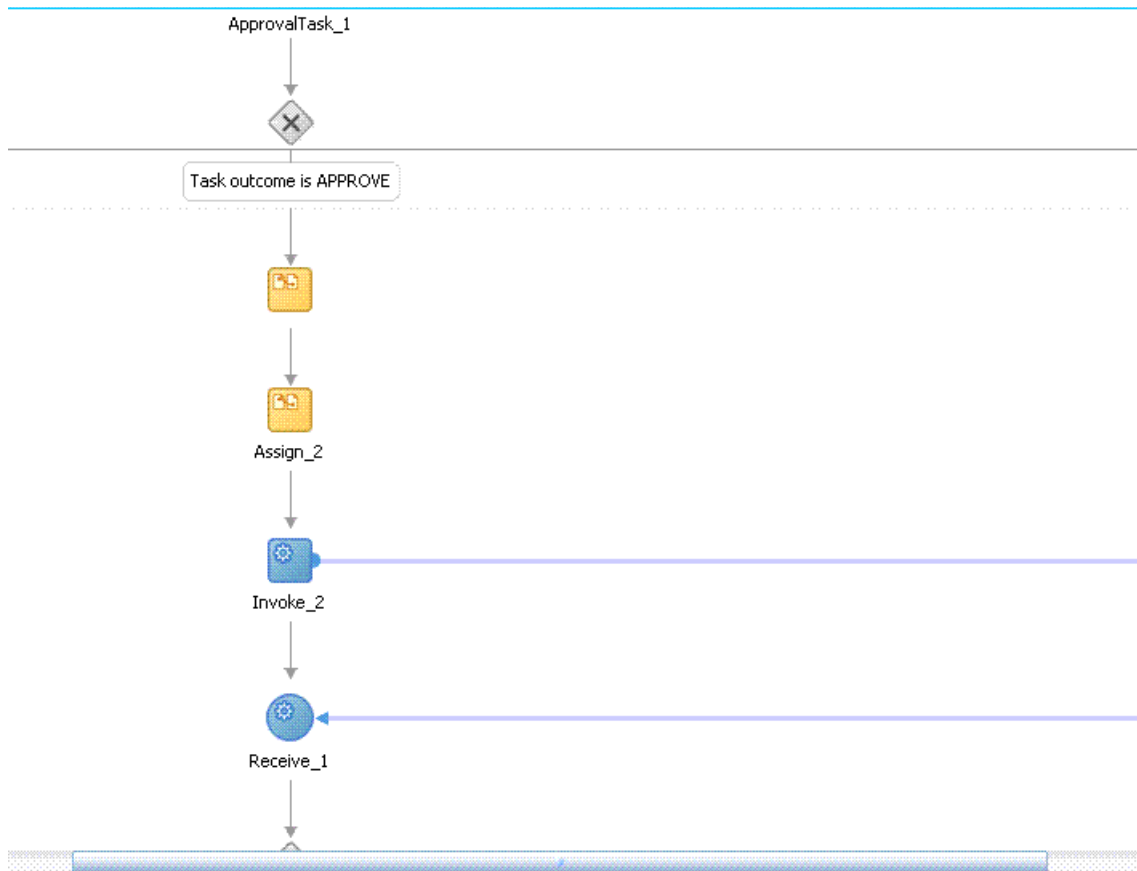
For using the default workflow, see "[Appying the Workflow By Using Approval Policy](#)" on page 27-30. This workflow contains an approval task that is assigned to the system administrator. After this approval task, a call is made to the asynchronous SoDCheck Web service to start SoD check. To complete and callback the SoDCheck Web service, you must run the 'Get SOD Check Results Approval' scheduled job. The workflow with SoDCheck Web service call is shown in [Figure 27-6](#).

Note: There are three levels of approval for any request: template-level approval, request-level approval, and operational-level approval. DefaultSODApproval workflow must only be attached at the operational level. For bulk requests, the request-level approval is common for all resources and users. After this level of approval, separate requests are created for each combination of resource and user. The workflow performs SoD check separately for each resource and user at a time.

The workflow is useful when SoD Check is required after request-level approval. One such instance is when the connector IT resource information is entered by the approver. Here, the IT Resource field in the request dataset is made approver-only, as shown below:

```
<AttributeReference name="EBS Server" attr-ref="EBS Server"
type="String" length="50" widget="itresource-lookup"
available-in-bulk="true" itresource-type="eBusiness Suite UM"
required="true" approver-only="true"/>
```

In this example, the IT resource information is not available when the request is raised. It is available only after the approver enters it. If it is entered during request-level approval, then the DefaultSODApproval workflow can be used at operational level.

Figure 27–6 Workflow with SoDCheck Web Service Call

After this, a switch case with approval tasks are assigned to the SoD Administrators role. Any user that has this role can claim the task and approve it. The switch is based on whether the SoD check result has passed or failed, as shown in [Figure 27–7](#):

Figure 27-7 Switch Case With Approval Tasks

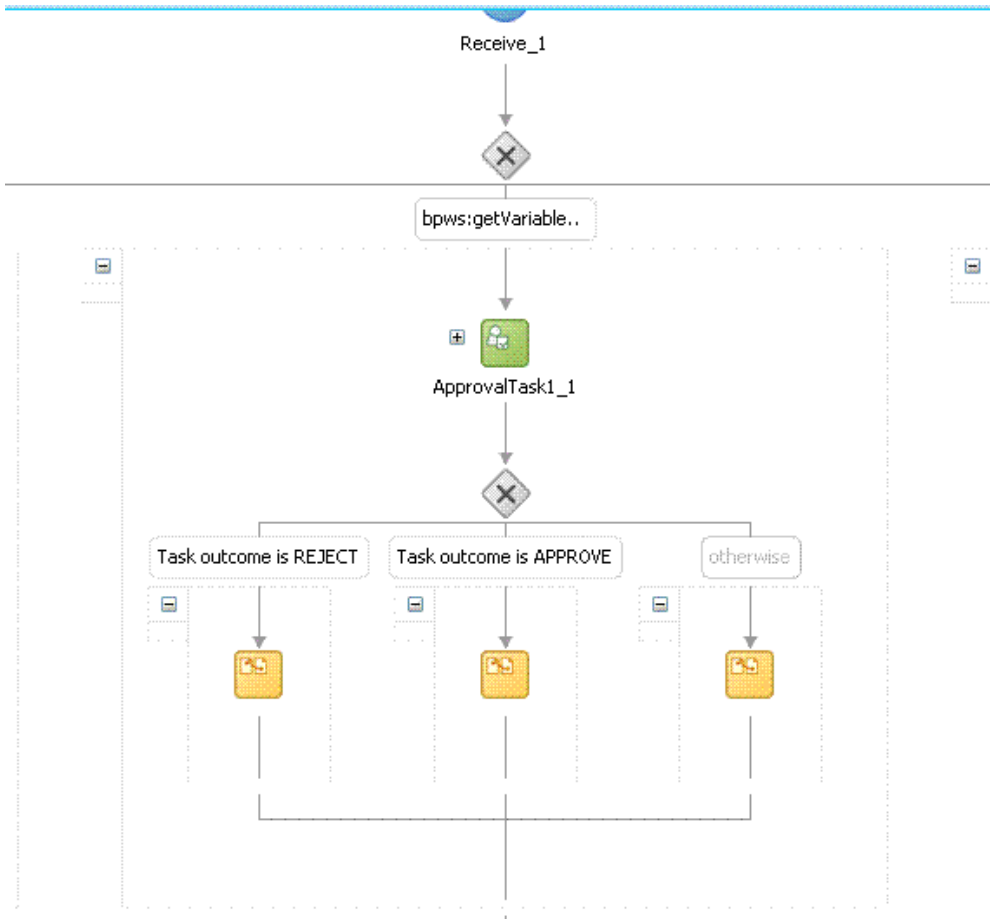
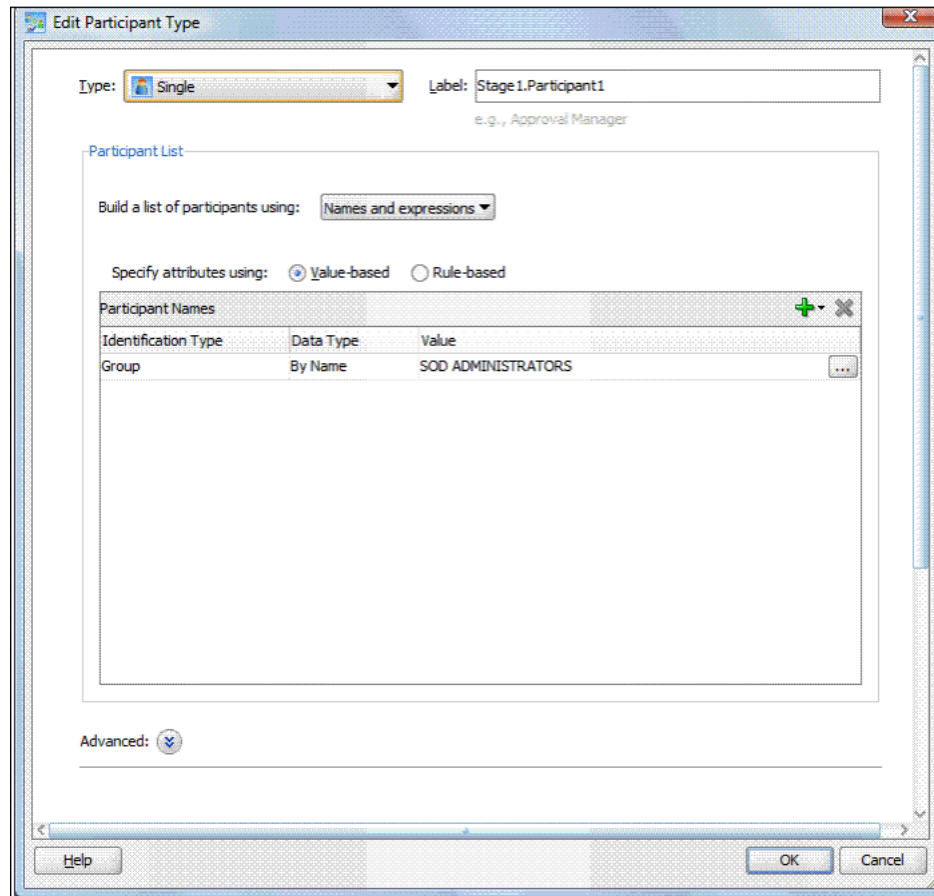


Figure 27-8 shows the assignment of the approval task.

Figure 27–8 Assignment of the Approval Task

Approval workflow has migrated to BPEL in Oracle Identity Manager 11g Release 1 (11.1.1), and therefore, you must use JDeveloper to view or modify the default workflows. The default SoD workflow is available in the `OIM_HOME/workflows/composites/DefaultSODApproval.zip` file. You can unzip this file and open the `DefaultSODApproval.jpr` in JDeveloper. In addition, you can create a new workflow by modifying any of the default approval workflows to call the SoD Check Web service and start SoD check on demand. To do so:

Creating and Deploying Workflows on SOA

- a. Create a new workflow project by running `OIM_HOME/workflows/new-workflow/new_project.xml`.

Here:

- `WEBLOGIC_HOME` is the directory on which Oracle WebLogic Server is installed.
- `NEW_PROJECT` is the name of the new project that you want to create.

To create the new workflow project, run the following command:

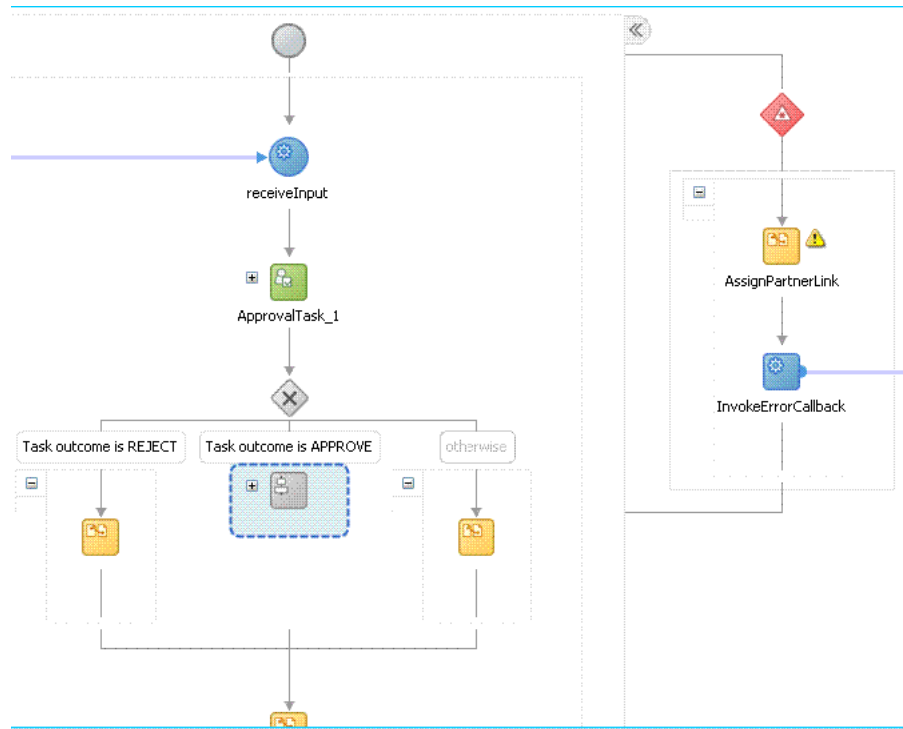
```
ant -f new_project.xml
```

This prompts for Project Name, Application Name, and Service Name for the new workflow name. Provide any name, such as `SODWorkflow` for all three. This creates a new project with the provided name in the `workflows/new-workflow/process-template/` directory.

- b. Navigate to process-template/*APPLICATION_NAME*/*PROJECT_NAME*/ and open *PROJECT_NAME*.jpr from JDevepoler, where *APPLICATION_NAME* and *PROJECT_NAME* are the names of the application and project respectively.

The *PROJECT_NAME*.jpr workflow is same as the DefaultRequestApproval workflow. You can modify this workflow to call the SoDCheck Web Service. [Figure 27–9](#) shows the default workflow modified to perform SoD Check after human approval:

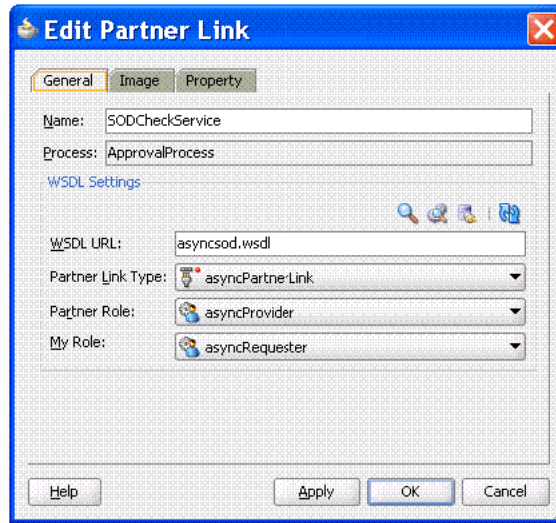
Figure 27–9 Modified Workflow To Perform SoD Check



- c. Extract *OIM_HOME*/workflows/composites/DefaultSODApproval.zip and copy *asynsod.wSDL* from the extracted directory to *OIM_HOME*/workflows/process-template/*APPLICATION_NAME*/*PROJECT_NAME*/. Add a Web service, such as *SODCheckService1*, in the *composite.xml* and provide the *asynsod.wSDL* as the WSDL file. The SoDCheck partner link is as shown in [Figure 27–10](#):

Note: BPEL connects to all external entities through a partner link.

Figure 27–10 SoD Check Partner Link



- d. In the ApprovalProcess.bpel design, include the following BPEL activities:
 - **ASSIGN:** An assign activity must be added before calling the SoD Check Web Service. This activity initializes the parameters required to call the Web Service. To create an assign activity:
 - i) Drag and drop the activity in the BPEL process opened in JDeveloper.
 - ii) After the activity is created, double-click the activity, and click the **Copy Operation** tab.
 - iii) Click **Add**, and then select **Copy Operation**. Provide the values for the variables, as shown in [Table 27–1](#):

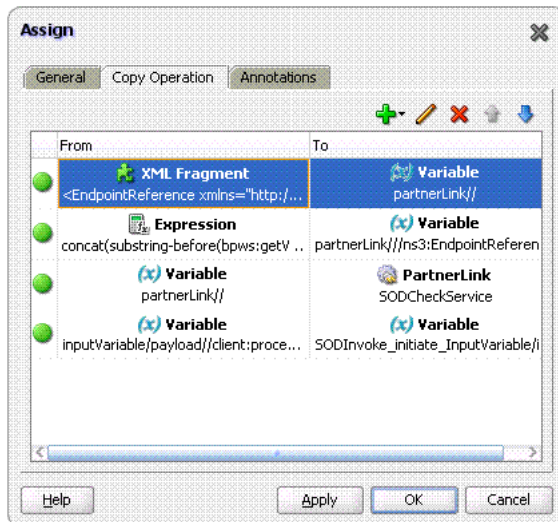
Table 27–1 Variables to Assign

Copy From	Copy To
XML Fragment <code><EndpointReference xmlns="http://www.w3.org/2005/08/addressing"> <Address/> </EndpointReference></code>	Variable partnerlink
Expression <code>concat(substring-before(bpws:getVariableData('inputVariable','payload','/client:process/ns1:url'), "/workflowservice/CallbackService"), '/sodcheck/SoDCheckInitiateService')</code>	Partnerlink, EndpointReference, Address
Variable partnerlink	Partner Link SODCheckService1
Variable Payload, RequestId	Variable SODInvoke_initiate_InputVariable, where SODInvoke_initiate_InputVariable is the variable defined in Invoke BPEL Activity

The following figures show the values to be added:

Figure 27–11 shows the final assign activity:

Figure 27–11 Final Assign Activity



- **INVOKE:** The details for this activity are:

Interaction Type: Partnerlink

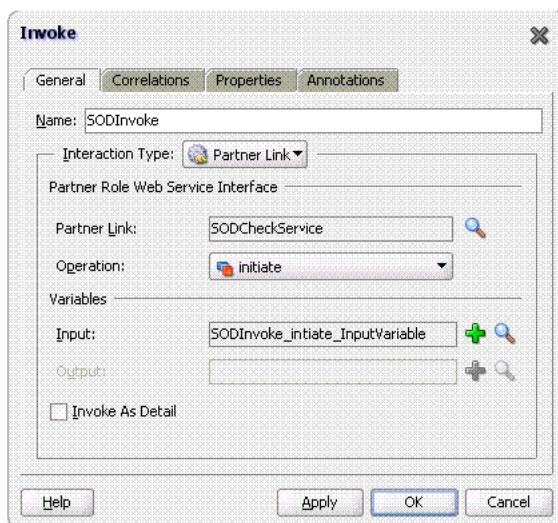
Partnerlink: SODCheckService

Operation: Initiate

Input Variable: SODInvoke_initiate_InputVariable

Figure 27–12 shows the Invoke dialog box with sample values in the fields:

Figure 27–12 The Invoke Dialog Box



- **RECEIVE:** The details for this activity are:

Interaction Type: Partnerlink

Partnerlink: SODCheckService

Operation: Result

Variable: SODResultReceive_result_InputVariable

Figure 27–13 shows the Receive dialog box with sample values in the fields:

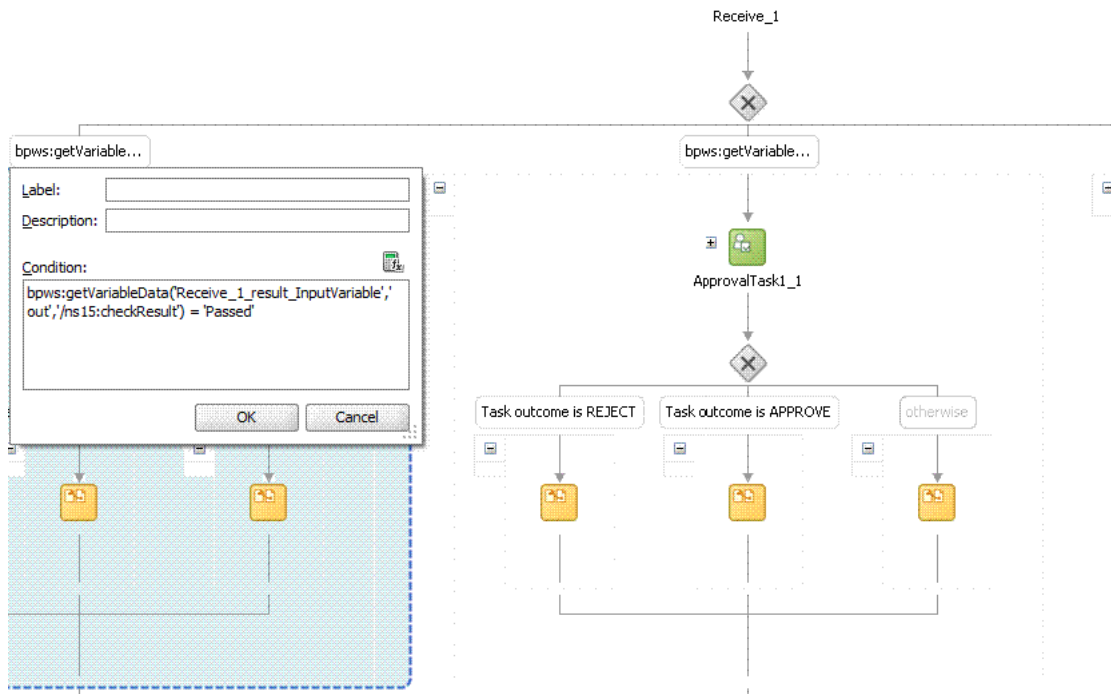
Figure 27–13 The Receive Dialog Box

The screenshot shows a dialog box titled "Receive" with a close button (X) in the top right corner. It has four tabs: "General", "Correlations", "Properties", and "Annotations". The "General" tab is selected. The dialog contains the following fields and controls:

- Name:** SODResultReceive
- Interaction Type:** Partner Link (with a dropdown arrow)
- My Role Web Service Interface:** (empty text field)
- Partner Link:** SODCheckService (with a search icon)
- Operation:** result (with a dropdown arrow)
- Variable:** SODResultReceive_result_InputVariable (with a plus icon and a search icon)
- Create Instance:**
- Buttons:** Help, Apply, OK, Cancel

- **SWITCH:** This activity is to switch between workflows based on SODCheck Result. The switch case is as shown in Figure 27–14:

Figure 27–14 Switch Case



- **New Human Tasks:** A new human task may be created and assigned to an approver other than the system administrator. The new approval task is same as the old one already present in the workflow, except that the approver is different. This human task is used in the switch case. For example, if the SoD check passes, then the approval task can be assigned to a role. If the SoD check fails, then the approval task can be assigned to the SOD administrators role. DefaultSODApproval always assigns approval task to the SoD administrators role.

Note: The SoDCheck Web service can be called multiple times.

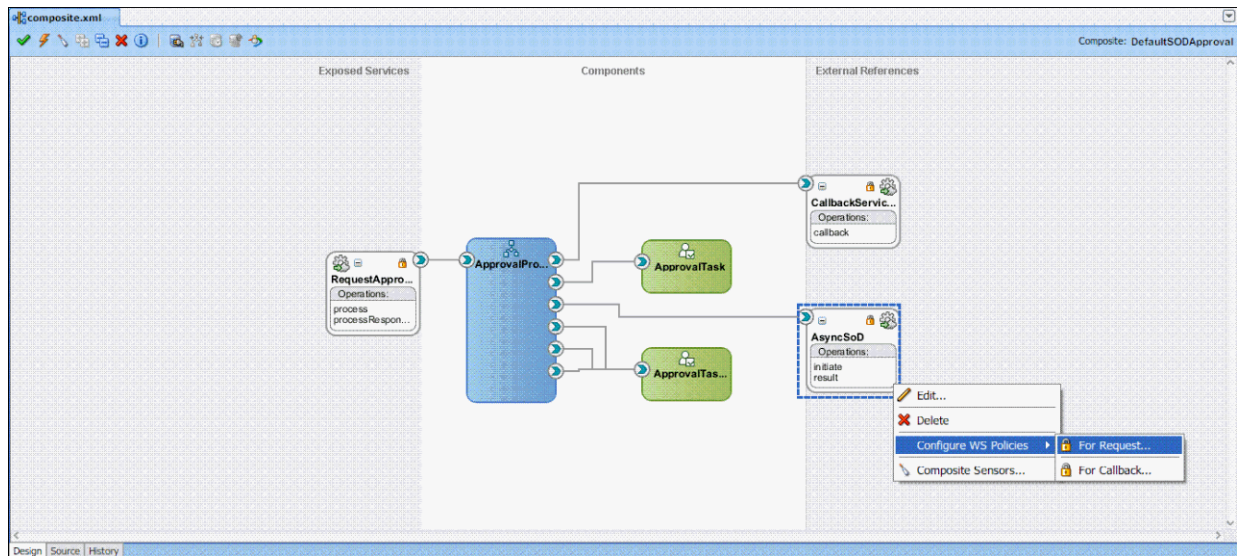
e. Applying SAML policy for request and callback for the AsyncSoD Web service:

OWSM SAML token with Message Protection Policy, which is based on Security Assertion Markup Language (SAML), is used as security policy for message protection in asynchronous calls for SoD checks from the SOA composite to Oracle Identity Manager. In asynchronous SoD check Web service, it is mandatory to use SAML token with Message Protection Client Policy for Request and SAML token with Message Protection Service Policy for Callback, as described in this section.

To apply SAML token with Message Protection Client policy for request:

- Right-click **AsynchSoD** Web service, and select **Configure WS Policies**, and then select **For Request**, as shown in [Figure 27–15](#):

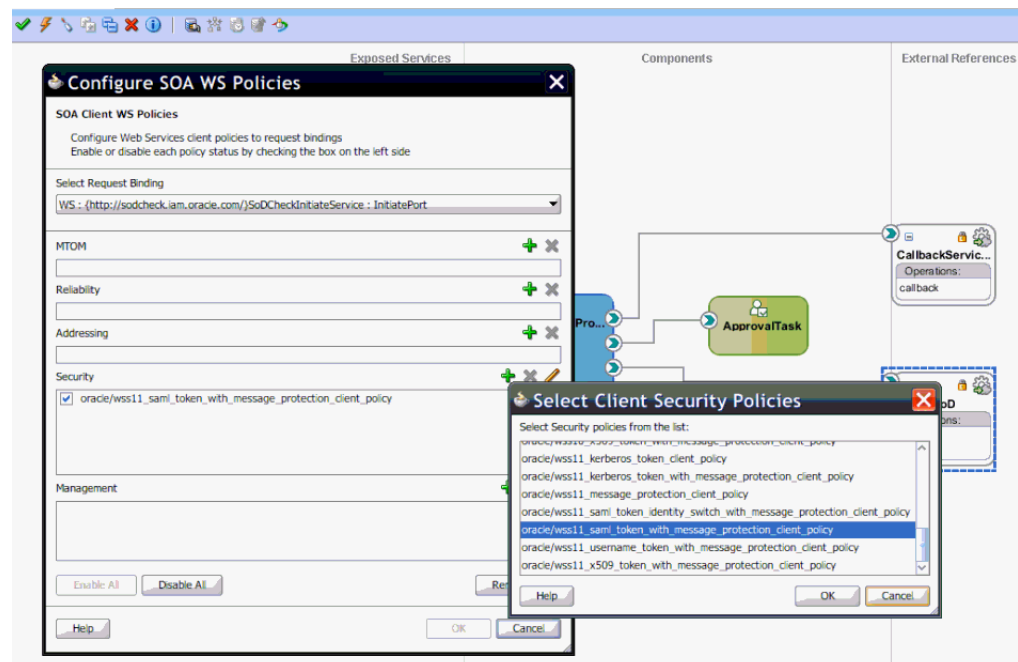
Figure 27–15 Configuring WS Policies for Request



ii) In the Configure SOA WS Policies dialog box, in the Security section, click the plus (+) icon to add a security policy.

iii) In the Select Client Security Policies dialog box, select `wss11_saml_token_with_message_protection_client_policy` as shown in Figure 27–16, and then click OK.

Figure 27–16 Select Client Security Policies

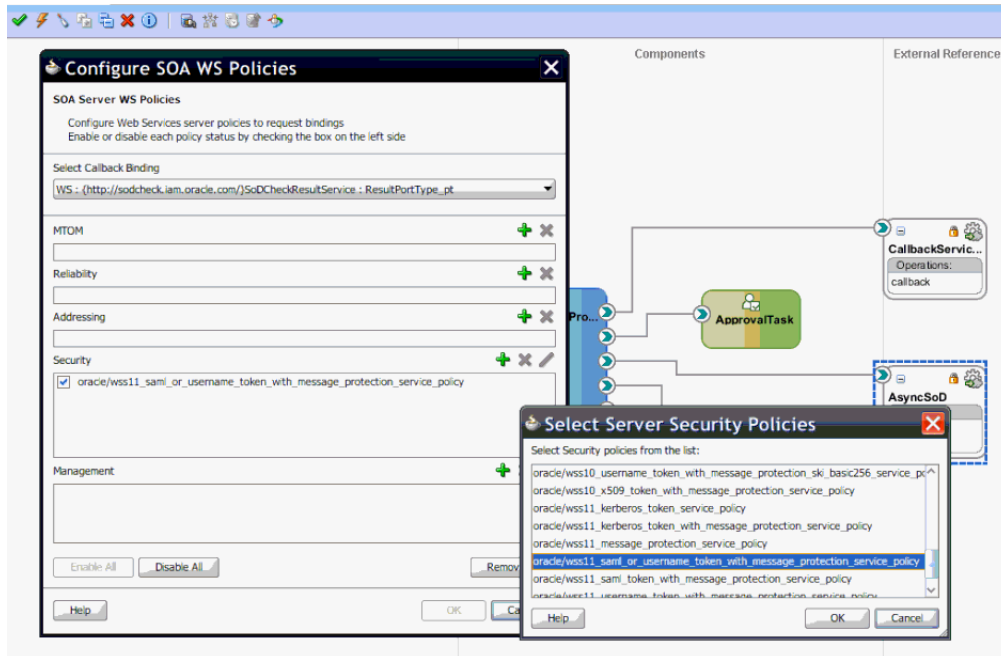


To apply SAML or Username token with Message Protection Service Policy for callback:

i) Right-click **AsynchSoD** Web service, and select **Configure WS Policies**, and then select **For Callback**.

- ii) In the Configure SOA WS Policies dialog box, in the Security section, click the plus (+) icon to add a security policy.
- iii) In the Select Server Security Policies dialog box, select `wss11_saml_or_username_token_with_message_protection_service_policy` as shown in Figure 27-17, and then click OK.

Figure 27-17 Select Server Security Policies



- f. Compile the project to see if there are any errors. If there are no errors, then right-click the project, and select **Deploy**. In the dialog box that is displayed, select any one of the following options:
 - **Deploy to Application Server:** Select this option and then select the appropriate server. The workflow is directly deployed to the application server.
 - **Deploy to JAR:** A JAR file is created under the JDeveloper deploy directory with the name `sca_PROJECT_NAME_rev1.0.jar`, where `PROJECT_NAME` is the name of the project.
- g. From the `SOA_HOME/bin/` directory, deploy the workflow on SOA server by running the following command:

Note:

- In this guide, `SOA_HOME` refers to the directory on which SOA server is installed.
- Before running this command, ensure that the SOA server is running.

```
ant -f ant-sca-deploy.xml -DserverURL=http://SOA_SERVER_HOSTNAME:SOA_PORT
-DsarLocation=JDeveloper/deploy/sca_PROJECT_NAME_rev1.0.jar -Duser=SOA_USER
-Dpassword=SOA_PASSWORD
```

Note: You must replace the following with valid values:

- *SOA_SERVER_HOSTNAME*
 - *SOA_PORT*
 - *PROJECT_NAME*
 - *SOA_USER*
 - *SOA_PASSWORD*
-
-

This deploys a new composite on SOA server. You can check if the composite is deployed by navigating to the following URL:

`http://SOA_SERVER_HOSTNAME:SOA_PORT/soa-infra`

In the URL, replace *SOA_SERVER_HOSTNAME* with the host name of the SOA server, and *SOA_PORT* with the port on which the SOA server is installed.

- h. Restart the SOA server.

See Also: [Chapter 25, "Developing SOA Composites"](#) for general procedure for creating a new workflow and registering it with Oracle Identity Manager

Registering the Workflow

- a. In the *OIM_HOME/workflows/registration/* directory, create a *NEW_PROJECT_NAME.props* file by copying the *DefaultRequestApproval.props*. Modify the *NEW_PROJECT_NAME.props* by changing the name attribute.

Here, *NEW_PROJECT_NAME* is the name of the new project that you created.

The *NEW_PROJECT_NAME.props* file has the following contents:

```
#This is is the input file for registering the default workflow
# <new project name>
name=NEW_PROJECT_NAME
category=Approval
providerType=BPEL
serviceName=RequestApprovalService
domainName=default
version=1.0
payLoadID=payload
operationID=process
listOfTasks=ApprovalTask
```

Here,

- The *version* parameter is the version of the workflow deployed on BPEL.
 - The *listOfTasks* parameter is the colon-separated list of approval tasks. For example, if you add a new approval task as *Approval_Task1*, then you must provide *ApprovalTask:ApprovalTask1* as the value for this parameter.
- b. Run *OIM_HOME/workflows/registration/registerworkflows-mp.xml* as shown:

```
ant -f registerworkflows-mp.xml register
```

This commands prompts for the following:

UserName: Enter Oracle Identity Manager administrator user name.

Password: Enter Oracle Identity Manager administrator Password

oim server t3 URL: Enter

t3://OIM_HOST_NAME/OIM_MANAGED_SERVER_PORT. Here, replace *OIM_HOST_NAME* with the host name of the computer on which Oracle Identity Manager is installed, and *OIM_MANAGED_SERVER_PORT* with the port on which Oracle Identity Manager is installed.

inputpath (complete file name) of the property file:

OIM_HOME/workflows/registration/NEW_PROJECT_NAME.props. Here, replace *NEW_PROJECT_NAME* with the name of the project that you created.

Appying the Workflow By Using Approval Policy

- a. Create approval policy for the request model to which you want to apply the SoD workflow. For example, if you want to perform SoD check while provisioning a resource, then create a policy for the Provision Resource request model. See "Creating Approval Policies" in the *Oracle Fusion Middleware User's Guide for Oracle Identity Manager* for information about creating approval policies.

Note:

- Always attach SoD workflow at the operational level of approval because SoD is triggered separately for each resource.
 - Whether SoD Engine is asynchronous or synchronous, the SoD Check Web Service is always asynchronous and workflow modification remains the same for both.
-
-

27.8.2 Modifying the Provisioning Workflow for SoD

Each process definition has a process task attached to provision entitlements to a user. The SoD validation process must be performed before triggering this task and immediately after inserting *all* data in the child table that holds entitlements on the target system. Therefore, you must hold this process task until the SoD validation process is completed after inserting the data in child tables. To achieve this, you create a Holder task that precedes the provisioning of an entitlement to a user.

The Holder task is added to prevent provisioning of a resource to a user before the SoD validation process is completed. User entitlements are provisioned only if this task is complete. The task is completed when the SoD engine validates that SoD policies or rules are not violated by the assignment of the entitlements.

If an SoD validation process has been performed in approval workflow, then the SoD validation process need not be performed again even if the SoD validation process is enabled at the provisioning level. Whether the SoD validation process needs to be performed or not can be assessed by checking the following before the SoD validation process at the provisioning level:

- Is the provisioning related to a request?
- If yes, is the SoDCheckStatus field set to SoDCheckCompleted?

- If yes, then do not perform the SoD validation process during entitlement provisioning.

Note: The SoD validation process will be performed again only when the process child form is edited to add, update, or remove entitlements.

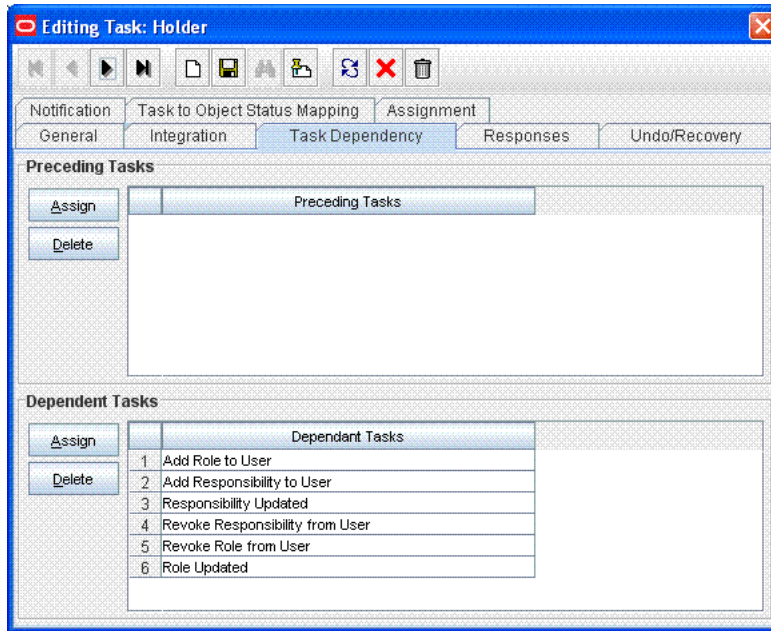
To modify the provisioning workflow for SoD validation:

1. Add a Holder task to the provisioning workflow. This task must be made conditional and the Allow Multiple instances option must be selected.

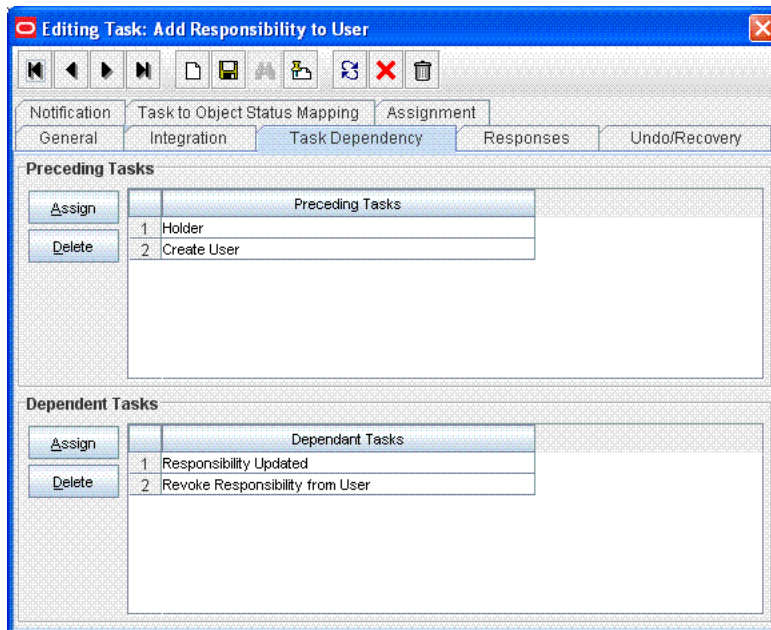
The following figure shows this Holder task:

2. Make the connector insert, update, and revoke entitlement tasks dependent on the Holder task.

The following figure shows all entitlement tasks of the Oracle e-Business User Management connector dependent on the Holder task:

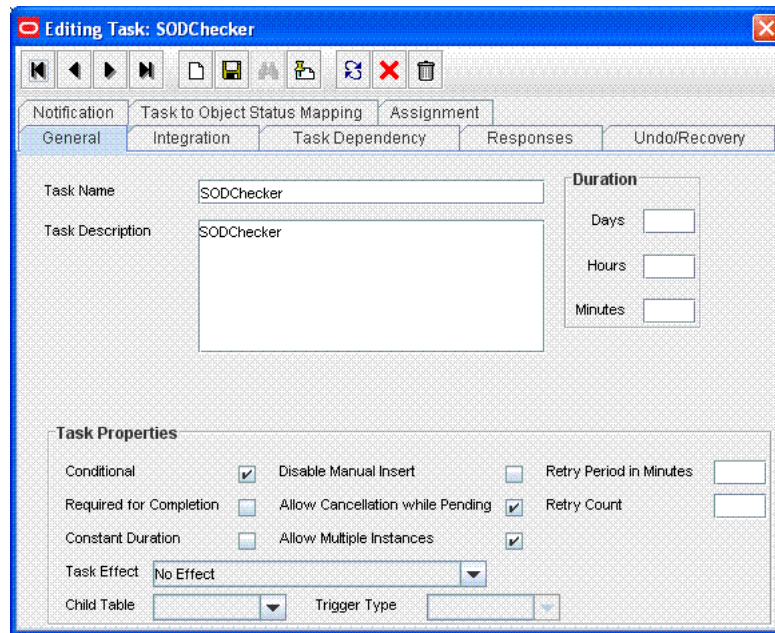


The following figure shows the Holder task as a preceding task of the Add Responsibility to User task:



3. Add the SODChecker task (any task whose name starts with SODChecker). This task must be made conditional.

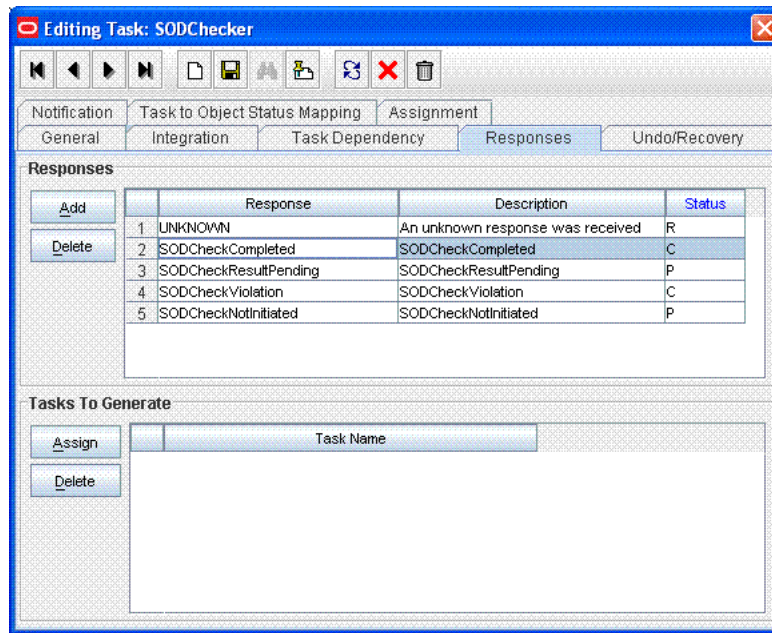
The following figure shows the SODChecker task:



4. Attach the InitiateSODCheck process task adapter to the SoDChecker task.
Attach the following response codes to the SODChecker task:

Response Code	Task Status	Description
SODCheckResultPending	P	The SoD validation process is initiated and results are awaited. Note: This response code is for an SoD engine that returns responses asynchronously.
SODCheckCompleted	C	The SoD validation process results have been returned, and the response shows that there is no SoD violation.
SODCheckViolation	C	The SoD validation process results have been returned, and the response shows that there is an SoD violation.
SODCheckNotInitiated	C	The SoD validation process has not been initiated because SoD has not been enabled in Oracle Identity Manager.

The following figure shows these response codes:



27.9 Marking Fields as Entitlements

This section contains the following topics:

- [Marking Request Dataset Attributes That Hold Entitlement Data](#)
- [Marking Child Process Form Tables That Hold Entitlement Data](#)

27.9.1 Marking Request Dataset Attributes That Hold Entitlement Data

The request dataset attribute that holds the entitlement shall be marked with entitlement property set to true. Below is an example:

```
<AttributeReference name="Responsibility Name" attr-ref="Responsibility Name"
type="String" length="256" widget="lookup-query" available-in-bulk="true"
required="true" entitlement="true">
    <lookupQuery lookup-query="select lkv_encoded as
lkv_encoded,lkv_decoded as lkv_decoded from lkv lkv,lku lku where
lkv.lku_key=lku.lku_key and
lku_type_string_key='Lookup.EBS.Responsibility' and
instr(lkv_encoded,concat('$Form data.Application Name','~'))>0"
display-field="lkv_decoded" save-field="lkv_encoded"/>
</AttributeReference>
```

See Also: ["Step 1: Creating a Request Dataset for the Resources"](#) on page 23-1 for information about creating the request dataset

27.9.2 Marking Child Process Form Tables That Hold Entitlement Data

Child process form tables can hold different types of multivalued data, for example, role data, profile data, and address information. You must mark the child process form tables holding entitlement data that you want to use for SoD operations. See ["Marking Entitlement Attributes on Child Process Forms"](#) on page 37-4 for information.

27.10 Custom Combination of Target Systems and SoD Engines

This section contains the following topics:

- [Using a Custom Target System](#)
- [Adding Custom SoD Engine](#)

27.10.1 Using a Custom Target System

Note:

Perform the procedure described in this section only if you want to use a target system other than Oracle e-Business Suite, SAP CUA, and SAP R3. You must also perform the procedures given in "[Adding Custom SoD Engine](#)" on page 27-42 if you are using an SoD engine other than Oracle Application Access Controls Governor and SAP GRC.

You can perform this procedure either before or at any time after first-time implementation of SoD in Oracle Identity Manager.

The following is a summary of the procedure to configure the SIL for a new target system:

1. Follow instructions given in the section "[Addressing Prerequisites](#)" on page 27-43.
2. Create Java class implementations of the `IdMvsSoDDataTransformationOper` interface for the connector. See "[Creating the Transformation Layer](#)" on page 27-36 for instructions.
3. Deploy the transformation service component. See "[Deploying the Transformation Layer](#)" on page 27-36.
4. Add entries in the registration XML file for the new target system. See "[Modifying the Registration XML File](#)" on page 27-36 for instructions.
5. Perform the procedure described in "[Configuring Workflows on Non SoD-enabled Connectors](#)" on page 27-13.
6. Mark child process forms that hold entitlement data. See "[Marking Fields as Entitlements](#)" on page 27-34 for instructions.
7. Register the new target system. See "[Registering the New Target System](#)" on page 27-38 for instructions.

27.10.1.1 Addressing Prerequisites

Ensure that the following prerequisites are addressed:

1. Load entitlement data from the target system to the SoD engine.
For details, see vendor documentation for the SoD engine.
2. Deploy the Oracle Identity Manager connector for the target system. See the connector documentation for more information.

27.10.1.2 Creating the Transformation Layer

The transformation layer is used to transform target system attribute values into values that can be used by the SoD engine. The transformation layer is required to be created for any new SoD engine or target system type.

You must create the transformation layer as an implementation of the `IdMvsSoDDataTransformationOper` interface. Create implementations of the `transformInput` and `transformSoDAnalysisInput` methods in the implementation class of the `IdMvsSoDDataTransformationOper` interface.

See Also: *Oracle Fusion Middleware Java API Reference for Oracle Identity Manager* for information about the implementation methods

In earlier releases of Oracle Identity Manager, the approval workflow data is read from the object forms. In Oracle Identity Manager 11g Release 1 (11.1.1), object forms are replaced by request datasets in the approval processes. As a result, the transformation layer must be changed so that entitlement data is read from the request dataset instead of object forms.

Transformation layer must also check the request model. If the request model is Provision Resource, then data must be read only from the request dataset. But if the request model is Modify Provisioned Resource, then data must be read both from the request dataset and process form.

27.10.1.3 Deploying the Transformation Layer

Transformation Service component is deployed as follows:

1. Create a JAR file for the Java classes that you created for implementation of the `IdMvsSoDDataTransformationOper` service component type.
2. Use the `UploadJar` utility to upload the JAR file as `ThirdParty`.

Note: The `UploadJar.sh` or `UploadJar.bat` utility is in the `OIM_HOME/bin/` directory. Run the utility from this location to upload the created JAR file to MDS.

27.10.1.4 Modifying the Registration XML File

Enter the details of the transformation layer in the `registration.xml` file as follows:

1. Import the `Registration.xml` file from the MDS. The `Registration.xml` file is present with namespace `/metadata/iam-features-sil/db/Registration.xml` in MDS.
2. Open the `Registration.xml` file in a text editor.
3. Add the `SystemType` and `ServiceComponent` elements as shown in this block of XML lines:

Note: Values that you must set are highlighted in bold. Guidelines and sample values are given after this block of XML.

```
<SystemType name="SYSTEM_TYPE_NAME" type="Sod Source
DataStore"></SystemType>

<ServiceComponent type="IdMvsSoDDataTransformationOper"
name="NAME_FOR_IMPLEMENTATION"
<Impl-Class>NAME_OF_IPMLEMENTATION_CLASS</Impl-Class>
```

```

        <IdMSystemType>OIM</IdMSystemType>
        <SoDEngineType>SoD_ENGINE</SoDEngineType>
        <srcSystemType>SYSTEM_TYPE_NAME</srcSystemType>

        <DataTransformation>
            <AttrSoD type="user"
name="NAME_OF_ATTRIBUTE_ON_TARGET_SYSTEM"
sourceIdMAttrName="NAME_OF_ATTRIBUTE_ON_SOD_ENGINE" isSourceKey="true"/>
            <AttrSoD type="user" name="firstname"
sourceIdMAttrName="firstname" isSourceKey="false"/>
            <AttrSoD type="user" name="lastname"
sourceIdMAttrName="lastname" isSourceKey="false"/>
            <AttrSoD type="duty" dutyType="ENTITLEMENT_TYPE"
name="accessorigid" sourceIdMAttrName="ENTITLEMENT_NAME" isSourceKey="true"/>
        </DataTransformation>

        <DataTransformation>
            .
            .
            .
        </DataTransformation>

        <DataTransformation>
            .
            .
            .
        </DataTransformation>
    </ServiceComponent>

```

Apply the following guidelines while adding the SystemType and ServiceComponent elements in the registration.xml file:

- Replace the placeholders with the following values:
 - **SYSTEM_TYPE_NAME**: Specify a name for the system type.
 - In the <SystemType> tag, type can have the SoD Source DataStore value for a custom target system, or SoD Engine as value for a custom SoD engine.
 - **NAME_FOR_IMPLEMENTATION**: Specify a name for the service component. For example: DBToOAACG
 - **NAME_OF_IPMLEMENTATION_CLASS**: Specify the name that you have set for the class that you create by performing the procedure described in ["Creating the Transformation Layer"](#) on page 27-36. For example: oracle.iam.grc.sod.scomp.impl.oaacg.transformation.IdMvsSoDDataTransformationOperDBvsOAACG
 - **SoD_ENGINE**: Enter OAACG if you are using Oracle Application Access Controls Governor as the SoD engine. Enter GRC if you are using SAP GRC as the SoD engine. If you are using a custom SIL provider, then enter the name that you set for that SoD engine.

See Also: ["Adding Custom SoD Engine"](#) on page 27-42

- **SYSTEM_TYPE_NAME**: Specify the system type name that you entered earlier.
- **NAME_OF_ATTRIBUTE_ON_TARGET_SYSTEM**: Specify the name of the attribute on the target system.
- **NAME_OF_ATTRIBUTE_ON_SOD_ENGINE**: Specify the name of the corresponding attribute on the SoD engine.
- **ENTITLEMENT_TYPE**: Enter the type of entitlement. For example: ROLE

- *ENTITLEMENT_NAME*: Enter the name of one instance of the entitlement.
For example: Resource Manager
 - Add one DataTransformation element for each attribute mapping that you want to create.
4. Save and close the Registration.xml file.
 5. Export the Registration.xml file back to MDS.

27.10.1.5 Registering the New Target System

To register the new target system, perform the procedure described in the following sections:

- [Running the Registration Script and Providing Registration Information](#)
- [Recording the Names of the System Types](#)

27.10.1.5.1 Running the Registration Script and Providing Registration Information The registration script (registration.sh and registration.bat) drives the registration process. When you run this script, it prompts you for the required information. The initial set of prompts displayed by the script are read from the registration.xml file. The registration script is in the *OIM_HOME*/bin directory. The registration.xml file is in the MDS.

Note: You can run the registration script multiple times, at any time during the lifecycle of the Oracle Identity Manager installation. For example, you might want to register a new SoD engine. When you run the script, use the prompts to guide you to the section (set of prompts) in which you want provide input. You can skip the remaining sections.

See [Example 27–1](#) for a sample run of the registration script. In that example, it is assumed that an IT resource has been created to provide information about the SoD engine.

To run the script and provide registration information for the Oracle Identity Manager installation, SoD engine, and target system:

1. Export the SILConfig.xml file from MDS. The SILConfig.xml file is present in MDS with namespace /metadata/iam-features-sil/db/SILConfig.xml.
2. Open the SILConfig.xml file in a text editor and provide values for the DOMBuilderFactoryImpl element.

The value of the DOMBuilderFactoryImpl element depends on the JRE that you are using:

- If you are using the Sun JRE or Oracle JRockit JRE, then uncomment the DOMBuilderFactoryImpl element containing the following value:

```
com.sun.org.apache.xerces.internal.jaxp.DocumentBuilderFactoryImpl
```
- If you are using the IBM JRE, then uncomment the DOMBuilderFactoryImpl element containing the following value:

```
org.apache.xerces.jaxp.DocumentBuilderFactoryImpl
```

3. In a command window, switch to the *OIM_HOME*/bin directory and run the registration script.

Enter login information for Oracle Identity Manager. You are prompted to provide the values for Username, Password, and URL. The sample run segment is given below:

```
[Enter the admin username:]OIM_ADMINISTRATOR_LOGIN
[Enter the admin password:]OIM_ADMINISTRATOR_PASSWORD
[Enter the service url:]t3://OIM_HOST_NAME:OIM_PORT_NO
```

Specify valid values for:

- *OIM_ADMINISTRATOR_LOGIN*
- *OIM_ADMINISTRATOR_PASSWORD*
- *OIM_HOST_NAME*
- *OIM_PORT_NO*

An example of the T3 URL is:

```
t3://localhost:14000
```

You are prompted to specify whether or not you want to proceed with registration:

```
Do you want to proceed with registration? (y/n)
```

Note: From this point onward, an explanation of each prompt displayed by the script is followed by the actual message of the prompt. The actual message is shown in monospace font in this document.

4. Enter *y* to proceed with the registration. You are prompted to specify whether or not you want to register an Oracle Identity Manager installation:

```
Register System Instance for type OIM? (y/n)
```

5. Enter *n*.

Note: From this point onward, the flow is specific to the registration of an Oracle e-Business Suite and Oracle Application Access Controls Governor installation. The flow is almost the same for the SAP CUA or SAP R/3 and SAP GRC installation.

6. You are prompted to specify whether or not you want to register an Oracle e-Business Suite installation:

```
Register System Instance for type EBS? (y/n)
```

7. Enter *n* if you want to use the existing Oracle e-Business Suite, which is registered by default. Enter *y* if you want to register a new EBS instance with another IT resource in Oracle Identity Manager.

8. If you enter *y*, then you are prompted to enter an instance name for the Oracle e-Business Suite installation:

```
Provide instance name
```

Enter a name for the Oracle e-Business Suite installation. For example:

ebs2

9. You are prompted to specify whether or not you want to register an Oracle Application Access Controls Governor installation:

Register System Instance for type OAACG? (y/n)

Enter n if you want to use the existing OAACG, which is registered by default.
Enter y if you want to register a new OAACG instance with another IT resource in Oracle Identity Manager.

10. If you enter y, then you are prompted to enter an instance name for the Oracle Application Access Controls Governor installation:

Provide instance name

Enter a name for the Oracle Application Access Controls Governor installation.
For example:

oaacg01

11. You are prompted to enter the name of the IT resource that you have created:

OIM ITResource Instance Name:

Enter the name of the IT resource that you created: OAACG ITR2

12. If there are no more SoD components (system instances) to register, then enter n in response to the remaining prompts. Otherwise, similar steps to be followed for SAP and GRC instances. After this, you are prompted for custom System Type that you added in Registration.xml, say NEW.

Register System Instance for type NEW? (y/n)

13. Enter y. You are prompted to enter an instance name for the custom type, as shown:

Provide instance name

14. Enter a name for the installation, for example, new1. If the added system type is SoD Engine, then you are prompted to enter the name of the IT resource that you have created:

OIM ITResource Instance Name:

15. Enter the name of the IT resource that you created: ITR_NEW.

16. Open the SILConfig.xml file in a text editor and provide values for the Topologies element. For information about topology values, refer to "[Recording the Names of the System Types](#)" on page 27-41.

The following block of XML shows the Topologies element and its child elements:

Note: If you have multiple target system and SoD engine combinations, then you can add multiple Topology elements inside the Topologies element.

```
<Topologies>
  <Topology>
    <name>@topologyName</name>
    <IdmId>@Idm RegistrationId</IdmId>
```

```

    <SodId>@Sod RegistrationId</SodId>
    <SDSID>@Sds RegistrationId</SDSID>
  </Topology>
</Topologies>

```

Enter values for the following child elements of the Topologies element:

- @topologyName: Enter a name for the topology.

Note: Set the same name for the Topology element as the value of the TopologyName IT resource parameter.

- @Idm RegistrationId: Enter the registration ID of the Oracle Identity Manager installation.
- @Sod RegistrationId: Enter the registration ID of the SoD engine.
- @Sds RegistrationId: Enter the registration ID of the target system.

See Also: Step 2 in "[Recording the Names of the System Types](#)" on page 27-41 for information about the child elements of the Topologies element.

17. Export SILConfig.xml back to MDS.

[Example 27-1](#) shows the output of a sample run of the registration script. Here, it is assumed that an IT resource has been created to provide information about the SoD engine.

Example 27-1 Sample Run of the Registration Script

```

sh registration.sh
Enter data related to login to OIM Server
[Enter the admin username:]OIM_ADMINISTRATOR_LOGIN
[Enter the admin password:]OIM_ADMINISTRATOR_PASSWORD
[Enter the service url:]t3://localhost:14000
Do you want to proceed with registration? (y/n)
Y
Register System Instance for type OIM?(y/n)
n
Register System Instance for type EBS?(y/n)
n
Register System Instance for type OAACG?(y/n)
n
Register System Instance for type SAP?(y/n)
n
Register System Instance for type GRC?(y/n)
n
Register System Instance for type NEW?(y/n)
Y
Provide instance name
new1
OIM ITResource Instance Name:
ITR_NEW

```

27.10.1.5.2 Recording the Names of the System Types At the end of the registration process, the names of the system types are set in the Oracle Identity Manager database. You can retrieve these names from the database by using the registration script. After you retrieve these names, you must enter them in the SILConfig.xml file.

To retrieve and record the names of the service components:

1. In a command window, switch to the following directory:

`OIM_HOME/bin/`

2. Run one the following commands:

For Microsoft Windows:

```
registration.bat printRegistrationIDs
```

For UNIX:

```
registration.sh printRegistrationIDs
```

The following is sample output of this command:

```
-----
System Type      Instance Name  Registration ID
-----
OIM              oim           1
EBS              Ebs           2
OAAACG          oaacg         3
-----
```

3. Copy these instance names for your reference.

27.10.2 Adding Custom SoD Engine

Note:

Perform the procedure described in this section only if you want to use an SoD engine other than Oracle Application Access Controls Governor and SAP GRC. You must also perform the procedures given in ["Using a Custom Target System"](#) on page 27-35 if you are using a target system other than Oracle e-Business Suite, SAP CUA, and SAP R3.

You must install the SoD engine before you begin creating the SIL provider.

You can perform this procedure either before or at any time after first-time implementation of SoD in Oracle Identity Manager.

The following is a summary of the procedure to create a SIL provider:

1. Follow instructions given in the section ["Addressing Prerequisites"](#) on page 27-43.
2. Create an IT resource to hold information about the SoD engine. See ["Creating an IT Resource to Hold Information about the SoD Engine"](#) on page 27-43.
3. Create Java class implementations of the interfaces for the SIL provider. See ["Implementing the Service Components for the Provider"](#) on page 27-44 for instructions.
4. Deploy the service components. See ["Deploying the Service Components"](#) on page 27-44.
5. Add entries in the registration XML file for the new SoD engine. See ["Modifying the Registration XML File for the New SoD Engine"](#) on page 27-44 for instructions.

6. Register the new SoD engine. See ["Registering the New SIL Provider"](#) on page 27-46 for instructions.

27.10.2.1 Addressing Prerequisites

Ensure that the following prerequisites are addressed:

1. Load entitlement data from the target system to the SOD engine. You can use any ETL utility to perform this step. For details, see vendor documentation for the SoD engine.
2. On the SoD engine, create policy definitions or risk definitions by using the data loaded from the target system.
3. Deploy the Oracle Identity Manager connector for the target system. See the connector documentation for more information.

27.10.2.2 Creating an IT Resource to Hold Information about the SoD Engine

You must create an IT resource to hold information about the SoD engine.

See [Chapter 11, "Developing Resource Objects"](#) for detailed information about creating an IT resource type (if it does not already exist) and IT resource. You can specify any name for the IT resource type and IT resource. The following table specifies the names of the parameters that the IT resource must contain:

Parameter	Description	Sample Value
Source Datastore Name	Enter the name of the source data store (the target system) that you defined in the SoD engine. You specify a source data store name while performing the procedure described in the "Configuring Oracle Application Access Controls Governor" on page 27-4 section.	EBS STMD122
dbuser	Enter the user name of the schema owner on the database used by the SoD engine. This account is used to access the Application Access Controls Governor database during SoD operations. Note: This parameter is specific to Oracle Application Access Controls Governor.	databaseusr1
dbpassword	Enter the password of the schema owner on the database used by the SoD engine. Note: This parameter is specific to Oracle Application Access Controls Governor.	Cryp100ne
jdbcURL	Enter the JDBC URL for connecting to the database used by the SoD engine. Note: This parameter is specific to Oracle Application Access Controls Governor.	jdbc:oracle:thin:@10.123.123.123
password	Enter the password of the account created on the SoD engine for API calls.	K1rb1r0s
port	Enter the number of the port at which the SoD engine is listening.	8090
server	Enter the IP address of the host computer on which the SoD engine is running.	10.231.231.231
sslEnable	Enter <code>true</code> if the SoD engine accepts only HTTPS communication requests. Otherwise, enter <code>false</code> .	false

Parameter	Description	Sample Value
username	Enter the user name of an account created on the SoD engine. This account is used to call the SoD engine APIs that are used during SoD validation.	jdoe
sodServerURL	Enter the URL of the SoD server, in the following format: http(s)://HOST_NAME:PORT_NUMBER/URL	http://10.231.231.231:8090/grcc/services/GrccService

Note: If you want to use multiple SoD engines, then create multiple IT resources with the same IT resource type.

27.10.2.3 Implementing the Service Components for the Provider

Create Java implementations of the following service components:

See Also: *Oracle Fusion Middleware Java API Reference for Oracle Identity Manager*

- **SoDAnalysisExecutionOper:** The SoD analysis layer must be implemented for any custom SoD engine, which is not provided by default.
- **IdMvsSoDDataTransformationOper:** Used to transform target system attribute values into values that can be used by the SoD engine. The transformation layer is required to be created for any new SoD engine or target system type.
- **CallBackIdMOper (optional):** To be implemented if any callback is required from SoD Analysis Layer to access Oracle Identity Manager.
- **SoDDataValidationOper (optional):** To be implemented to provide any validation on the attributes given in SoD Analysis layer.

27.10.2.4 Deploying the Service Components

Service components created in "[Implementing the Service Components for the Provider](#)" on page 27-44 are deployed as follows:

1. Create a JAR file for the Java classes that you created for Service Component implementation.
2. Use the UploadJar utility to upload the JAR file as ThirdParty.

Note: The UploadJar.sh or UploadJar.bat utility is in `OIM_HOME/bin`. Run the utility from this location to upload the created JAR file to MDS.

27.10.2.5 Modifying the Registration XML File for the New SoD Engine

Enter the details of the transformation layer in the Registration.xml file as follows:

1. Import the Registration.xml file from the MDS. The Registration.xml file is present with namespace `\metadata\iam-features-sil\db\Registration.xml` in MDS.
2. Open the Registration.xml file in a text editor.
3. Add the SystemType element for the SoD engine, as shown:

```
<SystemType name="SYSTEM_TYPE_NAME" type="SYSTEM_TYPE" isSynch="IS_SYNCH">
```

```

<!-- The Parameters which are required to connect the Sod Engine. -->
<Parameter name="PARAM_NAME1" required="true" />
<Parameter name="PARAM_NAME2" required="true" />
...
</SystemType>

```

Here, replace:

- *SYSTEM_TYPE_NAME* with a name for the system type.
- *SYSTEM_TYPE* with SoD Engine.
- *IS_SYNCH* with `true` or `false`, depending on whether the SoD engine is synchronous or asynchronous.
- *PARAM_NAME* with the name of the parameter used to connect the SoD engine. These parameter values must be provided while registering the SoD engine. These are read in service component implementation classes to connect to the SoD engine.

4. Add all implemented service components, as shown:

```

<ServiceComponent type="SERVICECOMPONENT_TYPE" name="NAME_FOR_IMPLEMENTATION"
  <Impl-Class>NAME_OF_IPMLEMENTATION_CLASS</Impl-Class>
  <IdMSystemType>SYSTEM_TYPE_NAME_FOR_IDM</IdMSystemType>
  <SoDEngineType>SYSTEM_TYPE_NAME_FOR_SOD_ENGINE</SoDEngineType>
  <srcSystemType>SYSTEM_TYPE_NAME_FOR_TARGET_SYSTEM</srcSystemType>

<!-- AttrSoD tag is only required for Sod Analysis Service Component-->
<AttrSoD type="user" isKey="true" name="NAME_OF_ATTRIBUTE_ON_SOD_ENGINE">
<!-- "name" attribute of the "Validation" element should be same as the "name"
of one of the registered "ServiceComponent" of type "SoDDataValidationOper" -->
<Validation name="NAME_FOR_VALIDATION_ON_ATTRIBUTE" />
</AttrSoD>

<AttrSoD type="duty" isKey="true" dutyType="ENTITLEMENT_TYPE"
name="NAME_OF_ENTITLEMENT_ON_SOD_ENGINE"><Validation name="isNotNullOACG"/>
</AttrSoD>

<AttrSoD...>
...
</AttrSoD>

<!-- DataTransformation tag is only required for transformation Service
component-->
  <DataTransformation>
    <AttrSoD type="user" name="NAME_OF_ATTRIBUTE_ON_TARGET_SYSTEM"
sourceIdMAttrName="NAME_OF_ATTRIBUTE_ON_SOD_ENGINE" isSourceKey="true"/>
    <AttrSoD type="user" name="firstname" sourceIdMAttrName="firstname"
isSourceKey="false"/>
    <AttrSoD type="user" name="lastname" sourceIdMAttrName="lastname"
isSourceKey="false"/>
    <AttrSoD type="duty" dutyType="ENTITLEMENT_TYPE" name="accessorigid"
sourceIdMAttrName="ENTITLEMENT_NAME" isSourceKey="true"/>
  </DataTransformation>
</ServiceComponent>

```

Here, replace:

- *SERVICECOMPONENT_TYPE*: Can have values such as `CallBackIdMOper`, `SoDAnalysisExecutionOper`, `SoDDataValidationOper`, or

IdMvsSoDDataTransformationOper depending upon the type of service component.

- *NAME_FOR_IMPLEMENTATION*: Specify a name for the service component, for example, DBToOAACG.
- *NAME_OF_IPMLEMENTATION_CLASS*: Specify the name that you have set for the class that you create by performing the procedure described in ["Creating the Transformation Layer"](#) on page 27-36. For example:
`oracle.iam.grc.sod.scomp.impl.oaacg.transformation.IdMvsSoDDataTransformationOperDBvsOAACG.`
- *SOD_ENGINE*: Enter OAACG if you are using Oracle Application Access Controls Governor as the SoD engine. Enter GRC if you are using SAP GRC as the SoD engine. If you are using a custom SIL provider, then enter the name that you set for that SoD engine.

See Also: ["Adding Custom SoD Engine"](#) on page 27-42

- *SYSTEM_TYPE_NAME*: Specify the system type name that you entered earlier.
 - *NAME_OF_ATTRIBUTE_ON_TARGET_SYSTEM*: Specify the name of the attribute on the target system.
 - *NAME_OF_ATTRIBUTE_ON_SOD_ENGINE*: Specify the name of the corresponding attribute on the SoD engine.
 - *ENTITLEMENT_TYPE*: Enter the type of entitlement, for example, ROLE.
 - *ENTITLEMENT_NAME*: Enter the name of one instance of the entitlement, for example, Resource Manager.
5. Save and close the Registration.xml file.
 6. Export the Registration.xml file back to MDS.

27.10.2.6 Registering the New SIL Provider

To register the new SIL provider, perform the procedure described in the following sections:

1. See ["Running the Registration Script and Providing Registration Information"](#) on page 27-38 for information on rerunning the registration script. In this run of the script, do not enter values for service components that have already been registered.
2. See ["Recording the Names of the System Types"](#) on page 27-41 for information on entering data about the new target system in the SILConfig.xml file.

27.11 Performing Role SoD Check with Oracle Identity Analytics

Role SoD Check is performed when a request to assign roles to, or revoke roles from, a user is raised. Role SoD Check with Oracle Identity Analytics is performed only when the request is raised; when roles are directly assigned or revoked, an SoD Check is not performed.

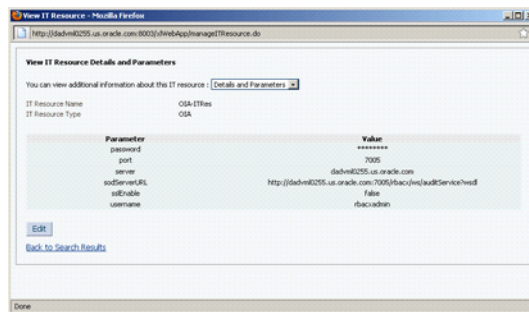
- [Enabling Role SoD Check](#)
- [Using Role SoD Check](#)

Note: Integration between Oracle Identity Manager and Oracle Identity Analytics is a pre-requisite for performing an SoD Check with Oracle Identity Analytics.

27.11.1 Enabling Role SoD Check

To enable Role SoD Check with Oracle Identity Analytics, you need to do the following.

1. Set the value of the XL.SoDCheckSystemProperty system property to TRUE. See "Administering System Properties" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager* for information about system properties.
2. Set the value of the RoleSoDCheckTopologyName system property to `sodoia`. This topology is predefined and registered.
3. Set OIA Connection Details in the 'OIA-ITRes' IT resource, as shown in the following figure:



27.11.2 Using Role SoD Check

The following sections describe how to implement the Role SoD functionality:

- [SoD Check When A User Requests a Role](#)
- [SoD Check When A User Revokes a Role](#)
- [SoD Check When an Administrator Requests To Assign Roles](#)
- [SoD Check When an Administrator Requests To Revoke Roles](#)
- [SoD Check for Assigning/Revoking Roles with Callback Policy Request](#)

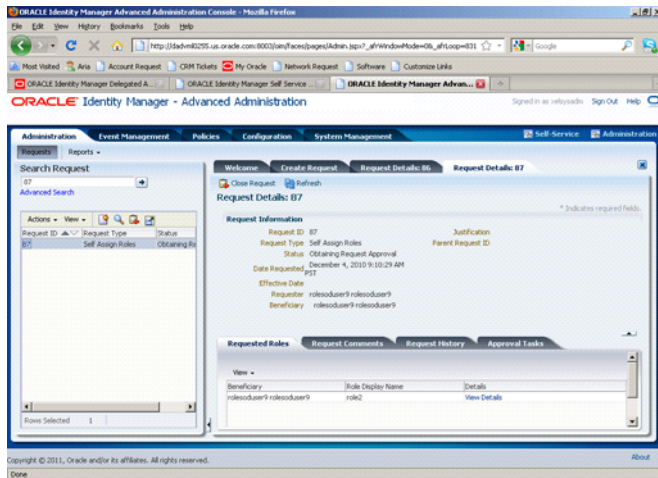
27.11.2.1 SoD Check When A User Requests a Role

The following steps are performed when a user raises a request for roles. SoD Check will be done if it has been enabled. This example procedure assumes that the user has already been assigned role1.

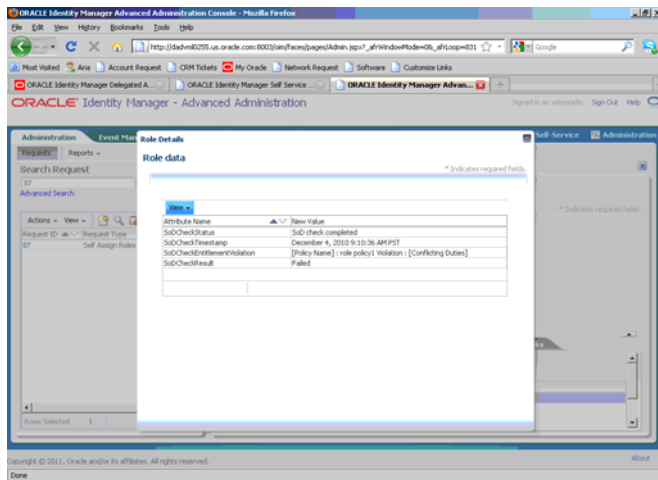
Perform the following steps:

1. Login to the Administrative and User Console as the user.
2. In the Self Service, click the **Create Request** tab, select the **Request for Me** option, and click **Next**.
3. From the Request Template list, select **Self Assign Roles**, and then click **Next**.
4. Select a role that conflicts with the role already assigned to the user, and click **Next**.

5. Add a justification and click **Finish** to submit the request.
6. Logout from the Administrative and User Console.
7. Login to the Administrative and User Console as the administrator to see the following:
 - View the request details, as shown in the following figure:



- View the SoD check results in the request details, as shown in the following figure:



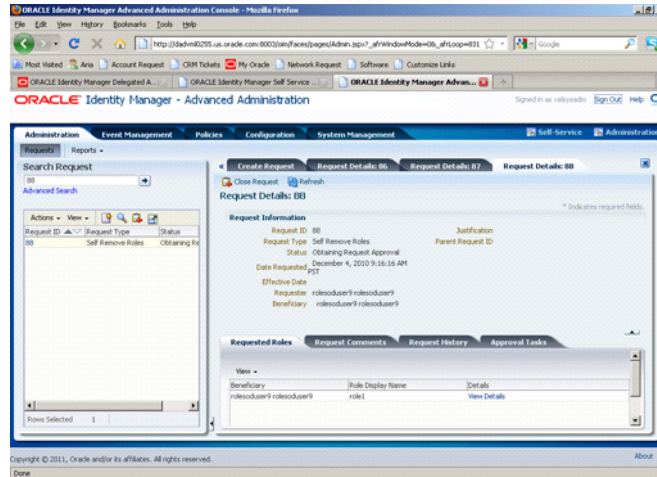
27.11.2.2 SoD Check When A User Revokes a Role

This procedure assumes that the user has already been assigned the role being revoked.

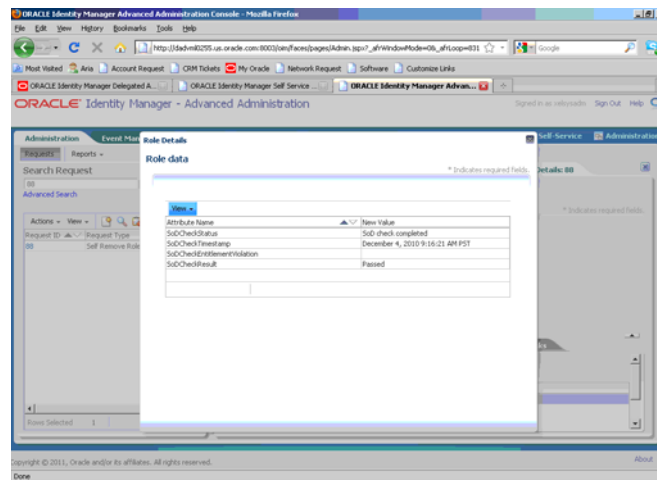
Perform the following steps:

1. Login to the Administrative and User Console as the user.
2. In the Self Service, click the **Create Request** tab, select the **Request for Me** option, and click **Next**.
3. From the Request Template list, select **Self Remove Roles**, and then click **Next**.
4. Select the role to be revoked, and click **Next**.
5. Click **Finish** to submit the request.

6. Logout from the Administrative and User Console.
7. Login to the Administrative and User Console as the administrator to see the following:
 - View the request details, as shown in the following figure:



- View the SoD check results in the request details. SoD Check passes because one of the conflicting roles has been removed. The following figure shows the SoD check results:

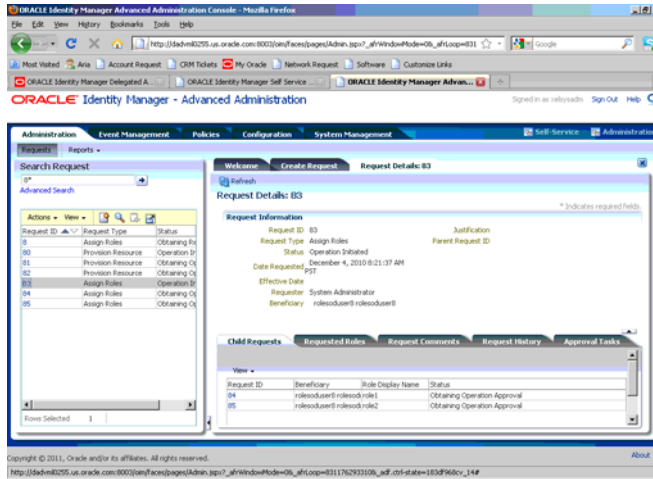


27.11.2.3 SoD Check When an Administrator Requests To Assign Roles

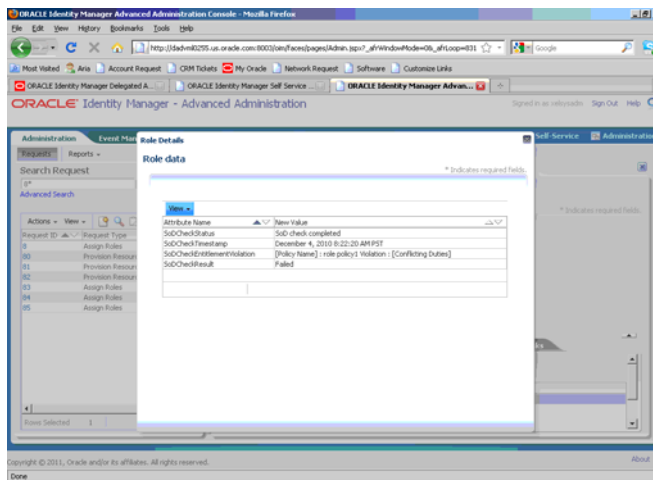
This procedure allows the system administrator to assign a role to the user.

1. Login to the Administrative and User Console as the system administrator.
2. In the Advanced Administration, click the **Create Request** tab, select **Assign Roles** from the Request Template list, and click **Next**.
3. Select the user for whom you want to raise the request, and click **Next**.
4. Select the roles to be assigned to the user, and click **Next**.
5. Click **Finish** to submit the request.

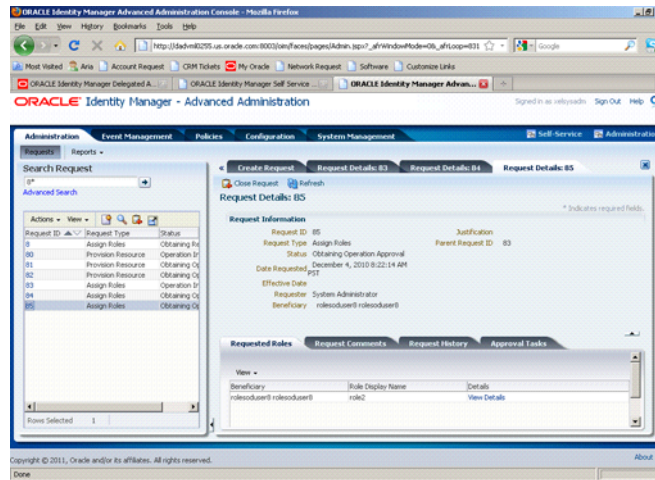
6. The next step is dependent on how many roles are involved with the request.
 - If a request is raised for more than two roles, a bulk request will be created. In this case, SoD Check will be performed for individual child requests. First, a request level approval must be done by logging in to the Self Service, selecting the request to be approved, and clicking **Approve Task**.
 - If the request is raised for two roles, then two child requests will be created and an SoD Check will be done for each. The following figure shows the two child requests:



- Click the first child request to view the SoD Check results.
- The results show that the SoD Check has failed because the request has been raised for two conflicting roles, as shown:



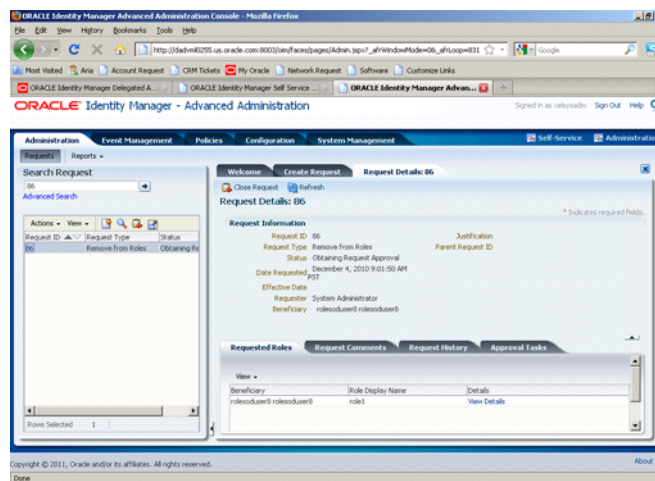
- Click the second request to view its details, as shown:



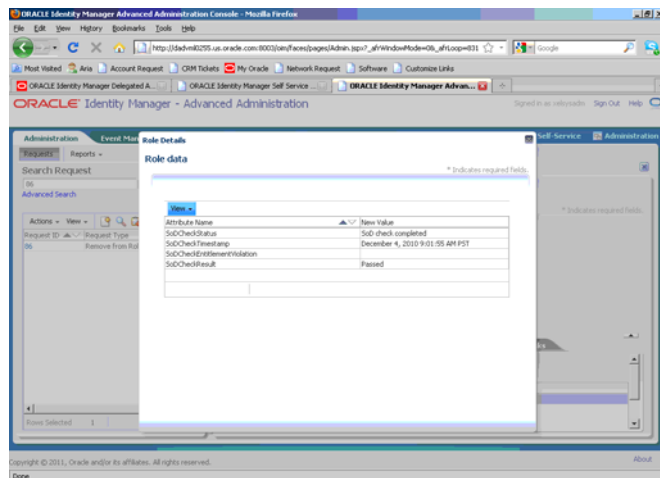
27.11.2.4 SoD Check When an Administrator Requests To Revoke Roles

This procedure allows the system administrator to revoke a role from the user. In the

1. Log in to the Administrative and User Console as the system administrator.
2. In the Advanced Administration, click the **Create Request** tab, select **Remove from Roles** from the Request Template list, and click **Next**.
3. Select the user for whom you want to raise the request, and click **Next**.
4. Select the roles to be revoked, and click **Next**.
5. Click **Finish** to submit the request.
6. View the request details, as shown:



7. View the SoD check results in the request details. SoD Check now passes as one of the conflicting roles has been removed, as shown:



27.11.2.5 SoD Check for Assigning/Revoking Roles with Callback Policy Request

This type of request is automatically approved. The request can be raised as detailed in the previous use cases. The Request Template selected is Assign Roles with callback policy or Revoke Roles with callback policy. SoD Check is performed as detailed in the previous use cases.

27.12 Using SoD in Provisioning Workflow

This section describes various use cases related to SoD:

Note: The procedures in this section are for Asynchronous SoD Engine, for example OAACG, for which you must run the scheduled job to complete the SoD check.

- [Direct Provisioning](#)
- [Updating Entitlements](#)
- [Request Provisioning](#)
- [Creating a Request to Modify Provisioned Resource](#)
- [Request Provisioning With the DefaultSODApproval Workflow](#)
- [Request Provisioning with Approver-Only Field and With the DefaultSODApproval Workflow](#)
- [Requesting for Self](#)
- [Provisioning Based on Access Policies](#)
- [Updating Entitlements By Using Provisioning Based on Access Policies](#)

27.12.1 Direct Provisioning

SoD check can be enabled in direct provisioning by performing the following steps:

1. Enable the SoD check system property. See "[Enabling SoD](#)" on page 27-8 for information about the SoD check system property.

2. Set the Topology Name parameter in the connector IT resource. See step 2 of ["Enabling SoD"](#) on page 27-8 for information about setting the Topology Name parameter.
3. Modify the connector provisioning workflow as described in ["Modifying the Provisioning Workflow for SoD"](#) on page 27-30.

To directly provision a SoD-enabled resource:

1. Create a user whose account is to be created on the target system.
2. In the user details page of Oracle Identity Administration, go to the Resources tab, and click **Add**.
3. Select the SoD-enabled resource from the available options.
4. Enter the process form details. Parent process form must contain SoD check fields, such as SoDCheckStatus, SoDCheckTrackingID, SoDCheckResult, SoDCheckTimestamp, and SoDCheckEntitlementViolation. These fields are populated with values for SoD check.
5. Make sure that you provide entitlement in the child forms. Otherwise, SoD check will not be performed because SoD is required only to check for conflicting entitlements.

6. After provisioning has been initiated, you can see the account created on target system but no entitlement is provisioned. This is because entitlement tasks are kept in the Waiting status until the SoD check completes.

In the resource profile, the Holder and SoDChecker tasks are in the Pending status and the SoDCheckStatus field in parent process form displays `SoD Check Result Pending`. The SoDCheckTrackingID field displays the tracking ID for the simulation started on the SoD engine.

7. Run the Get SOD Check Results Provisioning schedule job to get back results from the SoD engine and complete the SoD check.
8. If SoDCheckResult in the parent process form is in the Passed status, then there is no conflict and the entitlements are provisioned on the target system. The SoDCheckStatus field in the form displays `SoD Check Completed`, and the Holder and SoDChecker tasks are completed.
9. If SoDCheckResult in the parent process form is in the Failed status, then there is a conflict and the entitlements are not provisioned on the target system. The SoDCheckEntitlementViolation field in the form displays the conflicting policy on the SoD Engine, and the Holder task is canceled.

27.12.2 Updating Entitlements

Whenever you open the resource profile and try to add, update, or delete an entitlement in the child form, SoD check is triggered and the new Holder and SoDChecker tasks are generated. To complete SoD check, you must run the Get SOD Check Results Provisioning schedule job. If the new entitlement conflicts with the old ones, the new entitlement is not provisioned. Otherwise, the new entitlement is provisioned on the target system.

The addition, updation, and deletion of entitlements is done on a row by row basis. Therefore, for each modification, separate SoD check is triggered. If you make two modifications, then you can see two new SoDChecker tasks, one for each modification.

27.12.3 Request Provisioning

Default SoD check can be enabled in request provisioning by:

- Enabling the SoD Check system property
- Setting the Topology name parameter in the connector IT resource
- Request dataset must have an attribute of type IT Resource whose value is to be provided during request creation. A valid value must exist in its Topology Name parameter.

To create a request to provision a SoD-enabled resource:

1. In Oracle Identity Manager Advanced Administration, go to the Requests section, and click **Create Request** on the toolbar of the left pane. Select the **Provision Resource** request model.
2. Specify values for the attributes defined in the request dataset. The SoD check fields are part of the common dataset and is not displayed while creating the request. The fields can be viewed in the Request Details page after the request has been created.
3. Provide entitlement data as attribute values in the Create Request wizard, which displays the attribute names based on the request dataset. If no entitlements are provided, then SoD check will not start.
4. If SoD is successfully initiated, then the request is in `Sod check result pending` status. Check the SoD check field values in the request dataset. If SoD is not initiated successfully because of an error, then the request moves forward directly to the Obtaining Request Approval status. The `SoDCheckStatus` field in the dataset displays that Sod check is not initiated.
5. If the request is in `Sod check result pending` status, then run the Get SOD Check Results Approval schedule job to complete the SoD check, and move the request for approvals. If there is a conflict, then the `SoDCheckResult` field displays Failed status, and the `SoDCheckEntitlementViolation` field displays the violating duty.
6. Approve or reject the request-level approval from Oracle Identity Manager Self Service. For request-level approval, the default workflow is `DefaultRequestApproval`. Per this workflow, the approval task is assigned to the System Administrator role. If the administrator approves the request, then it goes for operational-level approval. By default, this level of approval is also assigned to the System Administrator role.

The approver can approve or reject the request based on the SoD check result, as displayed in the request detail fields.

After all approvals are obtained, the resource is provisioned on the target system.

27.12.4 Creating a Request to Modify Provisioned Resource

To request for modification of account on the target system:

1. In Oracle Identity Manager Advanced Administration, go to the Requests section, and click **Create Request** on the toolbar of the left pane. Select the **Modify Provisioned Resource** request model.
2. Select the resource to be modified and the user for which modification is required.

The values for the attributes predefined in the request dataset are displayed. The SoD check fields are part of the common dataset and are not displayed while

creating the request. The fields can be viewed in the Request Details page after the request has been created.

3. Add, update, or delete entitlement data as attribute values in the Create Request wizard, which displays the attribute names based on the request dataset.
4. If SoD is successfully initiated, then the request is in the `SoD check result pending` status. Check the SoD check field values in the request dataset. If SoD is not initiated successfully because of an error, then the request moves forward directly to the Obtaining Request Approval status. The `SoDCheckStatus` field in displays that SoD check is not initiated.
5. If the request is in the `SoD check result pending` status, then run the Get SOD Check Results Approval schedule job to complete the SoD check, and move the request for approvals. If there is a conflict, then the `SoDCheckResult` field displays Failed status, and the `SoDCheckEntitlementViolation` field displays the violating duty.
6. Approve or reject the request-level approval from the Self Service. For request-level approval, `DefaultRequestApproval` is the default workflow. Per this workflow, the approval task is assigned to the System Administrator role. If the administrator approves the request, then it goes for operational-level approval. By default, this level of approval is also assigned to System Administrator. The approver can approve or reject based on the SoD check result as displayed in the request detail fields.

After all approvals are obtained, the new entitlements are provisioned, updated, or deleted on the target system.

27.12.5 Request Provisioning With the DefaultSODApproval Workflow

When the `DefaultSODApproval` workflow has been specified by using an approval policy, perform the following steps to request for provisioning:

1. Specify the `DefaultSODApproval` workflow at the operational level. Therefore, the steps before the operational level of approval remain the same.
2. When the request moves to operational level of approval, per the `DefaultSODApproval` workflow, the approval task is assigned to the System Administrators role. If the administrator approves this task, then the SoD check Web service is loaded, and SoD check is initiated.

This can be confirmed by checking the request status, which must be `SoD check result pending`. Run the Get SOD Check Results Approval scheduled job to complete the SoD check.

3. An approval task is generated that is assigned to the SOD Administrators role. Any user having this role can claim the task and approve it.
4. Before approving the task, verify the SoD check results in the request details. If the task is approved, then the account and/or entitlement provisioning continues.

In this use case, SoD check is performed two times. First is the default SoD check before any level of approval, and the second one is initiated by the `DefaultSODApproval` workflow.

27.12.6 Request Provisioning with Approver-Only Field and With the DefaultSODApproval Workflow

If the IT resource field in the request dataset is made approver-only, then its value is not set when the request is created and can only be set by an approver. SoD check takes the SIL topology information from the connector IT resource. Therefore, if no resource has been selected, then SoD check is not triggered. Here, SoD check is not performed before any level of approval and only performed by using the DefaultSODApproval workflow after the approver has provided the IT resource information.

To request provisioning with approver-only field and with the DefaultSODApproval workflow:

1. When the request is submitted, it directly goes for request-level approval. There is no SoD check before this. Approver approves the request after providing the IT resource information.
2. When the request moves to the operational level of approval, per the DefaultSODApproval workflow, the approval task is assigned to the System Administrators role. If the administrator approves this task, then the SoD check Web service is invoked and SoD check is initiated. This can be confirmed by checking the request status, which must be `Sod check result pending`. You must run the Get SOD Check Results Approval schedule job to complete the SoD check.
3. An approval task is generated that is assigned to the SoD Administrators role. Any user having this role can claim the task and approve it.
4. Before approving the task, verify the SoD results in the request details. If task is approved, then the account and/or entitlement provisioning continues.

27.12.7 Requesting for Self

The user can raise a request for a resource or for modifying a resource from the Self Service. The steps for SoD check are similar for other resource provisioning use cases.

27.12.8 Provisioning Based on Access Policies

To perform provisioning based on access policies:

1. Create a new role.
2. Create an access policy to provision SoD-enabled resource to the new role. Make sure that you provide entitlements in the child form.
3. Assign this role to a newly created user. This provisions the account on the target system, but entitlement provisioning waits for SoD check.
4. Check the process form to verify the `SoDCheckStatus` field value. If the SoD check is successfully initiated, then the value of the `SoDCheckStatus` field is `SoD Check Result Pending`, and the Holder and SoDChecker tasks are generated, which are in the Pending status.
5. Run the Get SOD Check Results Provisioning scheduled job to complete the SoD check.
6. If the SoD check passes, then the Holder and SoDChecker tasks are completed and entitlements get provisioned. Otherwise, the Holder task is canceled and no entitlement provisioning takes place.

27.12.9 Updating Entitlements By Using Provisioning Based on Access Policies

To update entitlements by using provisioning based on access policies:

1. Create a new role.
2. Create an access policy to provision SoD-enabled resource to the new role. Make sure that you provide entitlements in the child form.
3. Assign this role to a user who already have SoD-enabled resource provisioned. Access policy only updates the entitlements for this account.
4. Check the new entitlement rows in the child process form. Check the parent process form to verify the SoDCheckStatus field value. If the SoD check is successfully initiated, then the value of the SoDCheckStatus field is `SoD Check Result Pending`, and the Holder and SoDChecker tasks are generated, which are in the Pending status.
5. Run the Get SOD Check Results Provisioning scheduled job to complete the SoD check.
6. If the SoD check passes, then the Holder and SoDChecker tasks are completed and entitlements get provisioned. Otherwise, the Holder task is canceled and no entitlement provisioning takes place.

27.13 Enabling Logging for SoD-Related Events

If you want to enable logging for all SoD-related events

1. In a text editor, open the `DOMAIN_HOME/config/fmwconfig/servers/oim_server1/logging.xml` file.
2. Search for the `<loggers>` element. The following is a sample `<loggers>` element:

```
<loggers>
<logger name="" level="WARNING:1">
<handler name="odl-handler"/>
<handler name="wls-domain"/>
<handler name="console-handler"/>
</logger>
```

You can change the logging level to `INCIDENT_ERROR:1`, `ERROR:1`, `NOTIFICATION:1`, `NOTIFICATION:16`, `TRACE:1`, `TRACE:16`, or `TRACE:32`. The default logging level prints the error and warning messages.

27.14 Troubleshooting SoD Check

[Table 27–2](#) lists the troubleshooting steps that you can perform if you encounter errors while performing SoD check.

Table 27–2 Troubleshooting SoD Check

Problem	Solution
<p>The SoDCheckStatus field in the process form displays no value or default value. For the field in EBS connector, SoD Check not initiated is the default value. Also, the SoDCheckResult field is not populated.</p>	<p>This means that SoD configuration is incorrect. Check if the Segregation of Duties (SOD) Check Required system property is set to <code>true</code>. If yes, then check the value of <code>topologyName</code> in connector IT resource field.</p> <p>If default registration is used, then the value of the <code>topologyName</code> parameter is <code>sodoaacg</code> for OAACG SoD engine and <code>sodgrc</code> for SAP GRC. If registration is done manually, then check if the corresponding topology is defined in the <code>SILConfig.xml</code> file and this file is seeded into MDS after the change.</p>
<p>The SoDCheckStatus field in the process form or request dataset displays <code>Sod check completed with error</code>. The SoDCheckResult field displays <code>Error from SoD Engine</code>.</p>	<p>SoD configuration is correct but SoD engine connection information might be incorrect, or there is an error from the SoD engine. Errors from the SoD engine can occur because of the following reasons:</p> <ul style="list-style-type: none"> ■ SoD engine or its corresponding database is down. ■ SoD engine is not completely synchronized with the target system. Therefore, specific entitlements for which SoD check is initiated may not be present on the SoD engine. <p>Check the SoD engine log for further errors. If no tracking ID is returned by the SoD engine, then simulation is not started successfully.</p>
<p>The SoDCheckStatus field in the process form is in the <code>SoD Result Pending</code> status, and does not move to the <code>SoD Check Completed</code> status even on running the scheduled job.</p>	<p>Make sure that you run the <code>Get SoD Check Results Provisioning</code> scheduled job and not the scheduled job for approval. Make sure that the scheduled job is triggered. You may enable logging at <code>DEBUG</code> level to confirm this.</p> <p>If the scheduled job is run and the SoD check is still not completing, then there must be an error from the SoD engine. Check the SoD engine log for details.</p>
<p>When requesting for SoD-enabled resource, no SoD fields are displayed in the dataset after creating the request, and the request directly moves to the request-level approval.</p>	<p>This error means that SoD configuration is incorrect. Check if the Segregation of Duties (SOD) Check Required system property is set to <code>true</code>. If yes, then check the value of <code>topologyName</code> in the connector IT resource field.</p> <p>If default registration is used, then the value of the <code>topologyName</code> parameter must be <code>sodoaacg</code> for the OAACG SoD engine and <code>sodgrc</code> for SAP GRC.</p> <p>If registration is performed manually, then check if the corresponding topology is defined in the <code>SILConfig.xml</code> file and this file is seeded into MDS after the change.</p>
<p>The SoDCheckStatus field in the request dataset stays in the <code>SoD Result Pending</code> status and does not move to the <code>SoD Check Completed</code> status even on running the scheduled job.</p>	<p>Make sure that you run the <code>Get SoD Check Results Approval</code> scheduled job and not the scheduled job for approval. Make sure that the scheduled job is triggered. You may enable logging at <code>DEBUG</code> level to confirm this.</p> <p>If the scheduled job is run and the SoD check is still not completing, then there must be an error from the SoD engine. Check the SoD engine log for details.</p>
<p>The SoD check is successfully performed during request provisioning, but the resource state in the user profile does not display as <code>Provisioned</code>. Therefore, the request is in the <code>Operation Initiated</code> status.</p>	<p>Check the process tasks in the resource history. If only the <code>System Validation</code> task is displayed, then the required data might not have been saved in the form. You can try saving the form manually by opening the form in edit mode and clicking Save. Enable the Auto-save option in the process definition for future requests.</p> <p>If other tasks, such as the task to create an account on the target system, are displayed in the resource history, then check the task details to verify if there is an error from the target system. For example, the account being created already exists on the target system.</p>

Table 27–2 (Cont.) Troubleshooting SoD Check

Problem	Solution
<p>The SoD check is successfully performed during direct provisioning, but the resource state in the user profile is not Provisioned.</p>	<p>Check the process tasks in the resource history. If only the System Validation task is displayed, then the required data might not have been saved in the form. This can happen if the Auto-Save option is on in the process definition, and therefore, the form is not displayed during direct provisioning. You can try saving the form manually by opening the form in edit mode and clicking Save. Disable the Auto-save option in the process definition for future requests.</p> <p>If other tasks, such as the task to create an account on the target system, are displayed in the resource history, then check the task details to verify if there is an error from the target system. For example, the account being created already exists on the target system.</p>
<p>Request provisioning has been successfully done and appropriate values are displayed in the request dataset, but the SoD status and result are not reflected to the process form.</p>	<p>Check the SoD field labels in the process form. They must be SoDCheckStatus, SoDCheckTrackingID, SoDCheckResult, SoDCheckTimestamp, and SoDCheckEntitlementViolation. If you change these field labels, then SoD field will not be mapped from the request dataset to the process form.</p>
<p>A particular SoD request is being tried several times and generating error.</p>	<p>There may be a problem with the SoD configuration or error in data submitted in a particular request. If you see that the traces of error for a request though SoD configuration is correct and you want to ignore the particular request, then you can prevent the JMS message related to the request from being tried multiple times by changing the Redelivery Limit for the OIMSODQueue from the WebLogic Administrative Console. To do so:</p> <ol style="list-style-type: none"> 1. Login to the WebLogic Administrative Console. 2. Go to Services, Messaging, JMS Modules, and OIMJMSModule. The list of all the queues are displayed. 3. Click oimSODQueue, and then click Delivery Failure. 4. Change the value of Redelivery Limit from -1 to a positive value. This determines how many times a SoD JMS message will be retried.
<p>Error in task assignment rules evaluation. Error in task assignment rules evaluation for user null. The error is Error in getting owners for "{0}" in configuration "{1}". Error occurred in getting owners for "SOD ADMINISTRATORS" in configuration "jazn.com". Ensure that the group name is valid and has associated owners. Contact Oracle Support if error is not fixable. Make sure that the rules specified for user null are valid.</p>	<p>Ignore this error. The reason for this error is that the OIMDBProvider does not support getting owners for a role. Therefore, SOA logs this error.</p>

Table 27-2 (Cont.) Troubleshooting SoD Check

Problem	Solution
<p>When trying to perform SoD check by using the DefaultSODApproval workflow, the following error message is displayed:</p> <p>Unknown Credential type to find the password for the given map : oim key : sodcheck.credentials</p>	<p>Add sodcheck.credentials as described in step 5 of "Enabling SoD" on page 27-8.</p>
<p>The following error is displayed:</p> <pre>[exec] Caused By: Thor.API.Exceptions.tcITResourceNotFoundException [exec] at com.thortech.xl.ejb.beansimpl.tcITResourceInstanceOperationsBean.getITResourceInstanceParametersData</pre>	<p>The SoD Engine IT resource has not been created. Therefore, according to the SoD Engine that is to be used, the corresponding IT resource must be created. For example, for OAACG, create OAACG-ITRes.</p>
<p>If SoD is enabled for more than one SoD Engine, for example OAACG and OIA, and you try to start SoD check with OIA, then errors might be logged from OAACG files.</p>	<p>This problem occurs if the topology entries in the SILConfig.xml file are incorrect. To see these entries, export the SILConfig.xml file from MDS. For default providers, the SILConfig.xml file has the SIL registration IDs corresponding to the topology names. The IDs in SILConfig.xml and the IDs that SIL registration script returns must be same. For example, the IDs for sodoia topology in SILConfig.xml are:</p> <pre><IdmId>1</IdmId> <SodId>7</SodId> <SDSID>6</SDSID></pre> <p>Then the IDs returned by the registration script are:</p> <pre>1 oimInstance 6 oimSDSInstance 7 oiaInstance</pre> <p>If these are different, then change the IDs in the SILConfig.xml file and reimport it by using the MDS utility.</p>

Part VII

Customization

This part contains chapters to help you customize the user interfaces available with Oracle Identity Manager.

It contains the following chapters:

- [Chapter 28, "Customizing Oracle Identity Manager Interfaces"](#)
- [Chapter 29, "Adding Custom ADF Tabs to Self Service"](#)
- [Chapter 30, "General Customization Concepts"](#)

Customizing Oracle Identity Manager Interfaces

This chapter explains how to customize various aspects of the user interfaces available in Oracle Identity Manager.

Note: Oracle Identity Manager 11g Release 1 (11.1.1) includes a number of UI pages based on earlier UI technologies known as transitional UIs. Due to technical differences, the transitional UIs are displayed in pop-up windows and have a different look and feel. These UIs are discussed in the relevant sections in this chapter.

This chapter contains these sections:

- [Branding Customization](#)
- [Style Sheet Modifications](#)
- [Renaming Button Labels](#)
- [Working with Menus and Tabs](#)
- [Disabling Features](#)
- [Adding or Deleting Columns in Console Tables](#)
- [Data Customization](#)
- [Injecting Custom URLs](#)
- [Changing Popup Properties](#)
- [Customizing the Workflow Designer](#)

28.1 Branding Customization

There are three aspects to customizing the branding information: logo, logo mouser over text, and branding text. These can be customized in two ways. This section describes one way to achieve this. The other method is changing skins by using appropriate styleclasses, which is explained in "[Style Sheet Modifications](#)" on page 28-10.

Branding customization can be done in any one of the following ways:

- Editing files in each console as described in this section
- Using stylesheets as described in "[Style Sheet Modifications](#)" on page 28-10

Skins and stylesheets are mechanisms to allow customization of the look and feel of the UI. The advantage of using stylesheet changes and custom skins is that they are centralized changes and easier to manage. For detailed information about skins and stylesheets, see "Customizing the Appearance Using Styles and Skins" in the *Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework* in the following URL:

http://download.oracle.com/docs/cd/E15523_01/web.1111/b31973/toc.htm

Note that certain customizations, such as branding logo mouseover text, are easier to perform as described in this section.

Branding customizations can be a combination of both ways, which is using stylesheet changes and custom skins for most of the changes, and using specific file changes as described in this section for particular items such as branding logo mouseover text.

You can customize branding in the following sections:

- [Login Page](#)
- [Identity Administration](#)
- [Unauthenticated Self-Service](#)
- [Authenticated Self Service](#)
- [Advanced Administration](#)

28.1.1 Login Page

Branding changes for Login page are configured in the following:

- `iam-consoles-faces.war/pages/Login.jspx`
- `admin.war/pages/Login.jspx`

Branding Text

Branding text can be changed by modifying the value of the resource bundle property 'branding_text'. Login.jspx refers to it in the following location:

```
<f:facet name="branding">
  <af:outputText value="#{admin.model.resources.common.branding_text}"
  id="ot1"/>
</f:facet>
```

The following resource bundle is used to refer to the property:

```
iam-consoles-faces.war/WEB-INF/lib/iam-consoles-faces.jar/oracle/iam/consoles/faces/resources/Common_LOCALE.properties
```

Here, use the appropriate file by replacing *LOCALE* with the value of the locale, for example, `Common_en.properties`.

Tip:

- Before modifying any file, create a backup of the file. The file you edit corresponds to a type of customization to perform with a console. For example, by editing the IdentityUIBundle_en.properties file, you change the branding text for Oracle Identity Administration.
- See [Chapter 35, "Upload JAR and Resource Bundle Utilities"](#) for instructions on how to upload and download JAR and WAR files.
- If you cannot extract the JAR file, then make sure you set the PATH environment variable to point to the directory in which the Java JDK is available.

Logo Image

To change the logo image:

1. Copy the new image, for example new-logo.png, in the oim.ear/iam-consoles-faces.war/images/ directory.
2. Add a stylesheet class after the following line in iam-consoles-faces.war/pages/Login.jspx:

```
<af:document title="#{admin.model.resources.common.title_login_page}" id="d1"
             theme="dark" maximized="true">
```

For example:

```
<af:document title="#{admin.model.resources.common.title_login_page}" id="d1"
             theme="dark" maximized="true">
<af:resource type="css">
    .MyCustomBrandingLogo {
        background-image:url('/oim/images/new-logo.png');
        background-position:center;
        background-repeat:repeat-n; display:block;
        height:3.5em; width:119px;
    }
</af:resource>
...
```

3. Add the attribute with the name "brandingLogoCls" and value as the CSS class name inside the line:

```
<af:pageTemplate viewId="/templates/IdmSignIn.jspx"
                 value="#{bindings.pageTemplateBinding}" id="pt1">
```

For example:

```
<af:pageTemplate viewId="/templates/IdmSignIn.jspx"
                 value="#{bindings.pageTemplateBinding}" id="pt1">
<f:attribute name="brandingLogoCls" value="MyCustomBrandingLogo"/>
...
```

4. Although you restart Oracle Identity Manager, the old files from the tmp/ directory in the WebLogic Managed Servers are loaded, and as a result, the new logo image is not displayed. To rectify this problem:
 - a. Shutdown Oracle WebLogic Server.
 - b. Delete all files in the \$DOMAIN_ROOT/servers/oim_server1/tmp/ directory.
 - c. Start Oracle WebLogic Server.

Logo Mouse Over Text

To change the logo mouse over text:

1. Extract the contents of the archive from the following location to a temporary folder.

```
ORACLE_HOME/modules/oracle.idm.uishell_11.1.1/oracle.idm.uishell.war
```

2. Create a backup of the original file.
3. Extract the contents of the /WEB-INF/lib/oracle-idm-uishell.jar file to a temporary location.
4. In the templates/IdmShell.jspx and templates/IdmSignIn.jspx files, search for shortDesc='Oracle', and put the required logo mouse over text (instead of the default 'Oracle') as the value of shortDesc at the following:

```
<af:panelBorderLayout styleClass="AFBrandingBar" id="ptpb11">
  <f:facet name="start">
    <af:panelGroupLayout styleClass="AFBrandingBarItem"
      inlineStyle="background-color: white;"
      id="ptpg12">
      <af:spacer shortDesc="Oracle"
        styleClass="#{attrs.brandingLogoCls}"
        id="pts1"/>
    </af:panelGroupLayout>
  </f:facet>
```

The code example is to help identify the location to replace. The exact code might differ a bit from this. The text to replace is inside the panel that uses the AFBrandingBarItem styleclass.

5. Repackage the JAR file and place it in the /WEB-INF/lib/ directory of the extracted Web Archive (WAR) file.
6. Repackage the oracle.idm.uishell.war file and place it in the original location.

When you repackage the WAR file, you must not overwrite the MANIFEST.MF file. For example, use the following command:

```
jar -cmf META-INF\MANIFEST.MF oracle.idm.uishell.war *
```

Note: The MANIFEST.MF file is created whenever a WAR file is created. This file contains information about the WAR file, such as the file build number and version number.

By entering m as an argument, you can force Java to retain the details that exist in the original MANIFEST.MF file. In other words, you are not overwriting the file. Oracle Identity Manager requires this information to use the contents in the oracle.idm.uishell.war file.

28.1.2 Identity Administration

Branding changes for Oracle Identity Administration are configured in admin.war/pages/Admin.jspx.

Branding Text

Branding text can be changed by modifying the value of the resource bundle property 'branding_text'. Admin.jspx refers to it in the following location:

```
<f:facet name="branding">
  <af:outputText value="#{resUI.branding_text}" id="ot1"/>
</f:facet>
```

Here, `resUI` refers to the following resource bundle:

```
admin.war/WEB-INF/lib/IdentityTaskFlow.jar/oracle/iam/identitytaskflow/resources/IdentityUIBundle_LOCALE.properties.
```

The branding facet takes any ADF tags, providing flexibility for your branding needs.

Logo Image

To change the logo image:

1. Copy the new image, for example `new-logo.png`, in the `oim.ear/iam-consoles-faces.war/images/` directory.
2. Add a stylesheet class after the following line in `admin.war/pages/Admin.jspx`:

```
<af:document title="#{resUI.window_title_text}" theme="dark" id="d1">
```

For example:

```
<af:document title="#{resUI.window_title_text}" theme="dark" id="d1">
<af:resource type="css">
  .MyCustomBrandingLogo {
    background-image:url('/oim/images/new-logo.png');
    background-position:center;
    background-repeat:repeat-n; display:block;
    height:3.5em; width:119px;
  }
</af:resource>
```

...

3. Add the attribute with the name "brandingLogoCls" and value as the CSS class name inside the line:

```
<af:pageTemplate viewId="/templates/IdmShell.jspx"
value="#{bindings.pageTemplateBinding}" id="pt1">
```

For example:

```
<af:pageTemplate viewId="/templates/IdmShell.jspx"
value="#{bindings.pageTemplateBinding}" id="pt1">
<f:attribute name="brandingLogoCls" value="MyCustomBrandingLogo"/>
```

...

4. Although you restart Oracle Identity Manager, the old files from the `tmp/` directory in the WebLogic Managed Servers are loaded, and as a result, the new logo image is not displayed. To rectify this problem:
 - a. Shutdown Oracle WebLogic Server.
 - b. Delete all files in the `$DOMAIN_ROOT/servers/oim_server1/tmp/` directory.
 - c. Start Oracle WebLogic Server.

Logo Mouse Over Text

If you perform the steps to change the logo mouseover text for the login page as described in "Login Page" on page 28-2, then you do not need to perform the steps in this section.

To change the logo mouseover text:

1. Extract the archive from the following location to a temporary folder:
ORACLE_HOME/modules/oracle.idm.uishell_11.1.1/oracle.idm.uishell.war
2. Create a backup of the original file.
3. Extract the *WEB-INF/lib/oracle-idm-uishell.jar* file to a temporary location.
4. In the *templates/IdmShell.jspx* and *templates/IdmSignIn.jspx* files, search for `shortDesc='Oracle'`, and put the required logo mouse over text (instead of the default 'Oracle') as the value of `shortDesc`, as shown:

```
<af:panelBorderLayout styleClass="AFBrandingBar"
  id="ptpb11">
  <f:facet name="start">
  <af:panelGroupLayout styleClass="AFBrandingBarItem"
    inlineStyle="background-color: white;"
    id="ptpg12">
    <af:spacer shortDesc="Oracle"
      styleClass="#{attrs.brandingLogoCls}"
      id="pts1"/>
    </af:panelGroupLayout>
  </f:facet>
```

5. Repackage the JAR file and put it in the *WEB-INF/lib* of the extracted WAR file.
6. Repackage the *oracle.idm.uishell.war* file and put it back in the original location.

28.1.3 Unauthenticated Self-Service

Branding changes for the Unauthenticated Self Service UI are configured in *iam-consoles-faces.war/pages/USelf.jspx*.

Branding Facet

Branding text can be changed by modifying the value of the resource bundle property 'header_branding'. *USelf.jspx* refers to it in the following location:

```
<f:facet name="branding">
  <af:outputText id="brandingTitle"
    value="#{uself.model.resources.uself.header_branding}"/>
</f:facet>
```

The *uself* resource bundle is located in:

iam-consoles-faces.war/WEB-INF/lib/iam-consoles-faces.jar/oracle/iam/consoles/faces/resources/USelf_LOCALE.properties

Logo Image

To change the logo image:

1. Copy the new image, for example *new-logo.png*, in the *oim.ear/iam-consoles-faces.war/images/* directory.
2. Add a stylesheet class after the following line in *iam-consoles-faces.war/pages/USelf.jspx*:

```
<f:facet name="metaContainer">
```

For example:

```
<f:facet name="metaContainer">
<af:resource type="css">
```

```

        .MyCustomBrandingLogo {
            background-image:url('/oim/images/new-logo.png');
            background-position:center;
            background-repeat:repeat-n; display:block;
            height:3.5em; width:119px;
        }
    </af:resource>
    ...

```

3. In the same file, add the attribute with the name "brandingLogoCls" and value as the CSS class name inside the following line:

```
<af:pageTemplate id="PANEL_PAGE" viewId="/templates/useself.jspx">
```

For example:

```

<af:pageTemplate id="PANEL_PAGE" viewId="/templates/useself.jspx">
<f:attribute name="brandingLogoCls" value="MyCustomBrandingLogo"/>
...

```

4. Although you restart Oracle Identity Manager, the old files from the tmp/ directory in the WebLogic Managed Servers are loaded, and as a result, the new logo image is not displayed. To rectify this problem:
 - a. Shutdown Oracle WebLogic Server.
 - b. Delete all files in the \$DOMAIN_ROOT/servers/oim_server1/tmp/ directory.
 - c. Start Oracle WebLogic Server.

Logo Mouse Over Text

To change the mouse over text for the branding logo image, perform one of the following steps in the iam-consoles-faces.war/pages/USelf.jspx file:

- Inside the line <af:pageTemplate id="PANEL_PAGE" viewId="/templates/useself.jspx">, add as static text value:

```
<f:attribute name="brandingLogoText" value="My Company"/>
```
- From the resource bundle, add the property, such as 'text_logo_mouseover', in the corresponding bundle (iam-consoles-faces.war/WEB-INF/lib/iam-consoles-faces.jar/oracle/iam/consol es/faces/resources/USelf_LOCALE.properties) and use it as shown:

```

<f:attribute name="brandingLogoText"
value="#{useself.model.resources.useself.text_logo_mouseover}"/>

```

28.1.4 Authenticated Self Service

Branding changes for the Oracle Identity Manager Self Service are configured in iam-consoles-faces.war/pages/Self.jspx.

Branding Facet

Branding text can be changed by modifying the value of the resource bundle property 'header_branding'. Self.jspx refers to it in the following location:

```

<f:facet name="branding">
    <af:outputText id="brandingTitle"
value="#{self.model.resources.self.header_branding}"/>
</f:facet>

```

The self resource bundle is located in the `iam-consoles-faces.war/WEB-INF/lib/iam-consoles-faces.jar/oracle/iam/consoles/faces/resources/Self_LOCALE.properties`.

Logo Image

To change the logo image:

1. Copy the new image, for example `new-logo.png`, in the `oim.ear/iam-consoles-faces.war/images/` directory.
2. Add a stylesheet class after the following line in `iam-consoles-faces.war/pages/Self.jspx`:

```
<f:facet name="metaContainer">
```

For example:

```
<f:facet name="metaContainer">
<af:resource type="css">
    .MyCustomBrandingLogo {
        background-image:url('/oim/images/new-logo.png');
        background-position:center;
        background-repeat:repeat-n; display:block;
        height:3.5em; width:119px;
    }
</af:resource>
```

...

3. In the same file, add the attribute with the name `"brandingLogoCls"` and value as the CSS class name inside the following line:

```
<af:pageTemplate id="PANEL_PAGE" viewId="/templates/self.jspx">
```

For example:

```
<af:pageTemplate id="PANEL_PAGE" viewId="/templates/self.jspx">
<f:attribute name="brandingLogoCls" value="MyCustomBrandingLogo"/>
```

...

4. Although you restart Oracle Identity Manager, the old files from the `tmp/` directory in the WebLogic Managed Servers are loaded, and as a result, the new logo image is not displayed. To rectify this problem:
 - a. Shutdown Oracle WebLogic Server.
 - b. Delete all files in the `$DOMAIN_ROOT/servers/oim_server1/tmp/` directory.
 - c. Start Oracle WebLogic Server.

Logo Mouse Over Text

To change the mouse over text for the branding logo image, perform one of the following steps in the `iam-consoles-faces.war/pages/Self.jspx` file:

- Inside the line `<af:pageTemplate id="PANEL_PAGE" viewId="/templates/self.jspx">`, add as static text value:

```
<f:attribute name="brandingLogoText" value="My Company"/>
```
- From the resource bundle, add the property, such as `'text_logo_mouseover'`, in the corresponding bundle

(iam-consoles-faces.war/WEB-INF/lib/iam-consoles-faces.jar/oracle/iam/consol
es/faces/resources/Self_LOCALE.properties) and use it as shown:

```
<f:attribute name="brandingLogoText"
value="#{self.model.resources.self.text_logo_mouseover}"/>
```

28.1.5 Advanced Administration

Branding changes for the Advanced Administration are configured in
iam-consoles-faces.war/pages/Admin.jspx.

Branding Facet

Branding text can be changed by modifying the value of the resource bundle property
'header_branding_adv'. Admin.jspx refers to it in the following location:

```
<f:facet name="branding">
  <af:outputText id="brandingTitle"
value="#{admin.model.resources.admin.header_branding_adv}"/>
</f:facet>
```

The admin resource bundle is located in
iam-consoles-faces.war/WEB-INF/lib/iam-consoles-faces.jar/oracle/iam/consol
es/faces/resources/Admin_LOCALE.properties.

Logo Image

To change the logo image:

1. Copy the new image, for example new-logo.png, in the
oim.ear/iam-consoles-faces.war/images/ directory.
2. Add a stylesheet class after the following line in
iam-consoles-faces.war/pages/Admin.jspx:

```
<f:facet name="metaContainer">
```

For example:

```
<f:facet name="metaContainer">
<af:resource type="css">
  .MyCustomBrandingLogo {
    background-image:url('/oim/images/new-logo.png');
    background-position:center;
    background-repeat:repeat-n; display:block;
    height:3.5em; width:119px;
  }
</af:resource>
...
```

3. In the same file, add the attribute with the name "brandingLogoCls" and value as
the CSS class name inside the following line:

```
<af:pageTemplate id="PANEL_PAGE" viewId="/templates/admin.jspx">
```

For example:

```
<af:pageTemplate id="PANEL_PAGE" viewId="/templates/admin.jspx">
<f:attribute name="brandingLogoCls" value="MyCustomBrandingLogo"/>
...
```

4. Although you restart Oracle Identity Manager, the old files from the tmp/ directory in the WebLogic Managed Servers are loaded, and as a result, the new logo image is not displayed. To rectify this problem:
 - a. Shutdown Oracle WebLogic Server.
 - b. Delete all files in the \$DOMAIN_ROOT/servers/oim_server1/tmp/ directory.
 - c. Start Oracle WebLogic Server.

Logo Mouse Over Text

To change the mouse over text for the branding logo image, perform one of the following steps in the iam-consoles-faces.war/pages/Admin.jspx file:

- Inside the line `<af:pageTemplate id="PANEL_PAGE" viewId="/templates/admin.jspx">`, add as static text value:


```
<f:attribute name="brandingLogoText" value="My Company"/>
```
- From the resource bundle, add the property, such as 'text_logo_mouseover', in the corresponding bundle (iam-consoles-faces.war/WEB-INF/lib/iam-consoles-faces.jar/oracle/iam/consol es/faces/resources/Admin_LOCALE.properties) and use it as shown:


```
<f:attribute name="brandingLogoText" value="#{admin.model.resources.admin.text_logo_mouseover}"/>
```

28.2 Style Sheet Modifications

This section provides instructions on style sheet updates. Topics include:

- [Introduction to the Style Sheets](#)
- [Creating Custom Skins and Overriding Style Sheets](#)
- [Style Sheets in Transitional UI](#)

28.2.1 Introduction to the Style Sheets

Oracle ADF uses skins along with styles to customize the appearance of an application. These concepts apply to all the Oracle Identity Manager interfaces, with the exception of the Transitional UI popups.

See Also: Before customizing style sheets, see Customizing the Appearance Using Styles and Skins in the *Fusion Middleware Web User Interface Developer's Guide* in the following URL:

http://download.oracle.com/docs/cd/E15523_01/web.1111/b31973/toc.htm

Following URL gives a list of all the CSS style selectors that can be used to customize the style sheets:

http://download.oracle.com/docs/cd/E15523_01/apirefs.1111/e15862/toc.htm

You configure new skins in trinidad-config.xml. The default skin for Oracle Identity Manager is "fusion":

```
<?xml version="1.0" encoding='utf-8'?>
<trinidad-config xmlns="http://myfaces.apache.org/trinidad/config">
```

```
<skin-family>fusion</skin-family>
</trinidad-config>
```

There are two console-specific trinidad-config.xml files:

- For Identity Administration: admin.war/WEB-INF/trinidad-config.xml
- For Self Service, Unauthenticated, and Advanced Administration: iam-consoles-faces.war/WEB-INF/trinidad-config.xml

28.2.2 Creating Custom Skins and Overriding Style Sheets

To keep the defaults coming from the "fusion" skin and override certain style sheet elements:

1. Create the skin in the trinidad-skins.xml file. Make a copy of the trinidad-skins.xml from the /iam-consoles-faces.war/WEB-INF/ directory to the admin.war/WEB-INF/ directory. For both the /admin.war/WEB-INF/trinidad-skins.xml file and the /iam-consoles-faces.war/WEB-INF/trinidad-skins.xml file, make following changes:

```
<?xml version="1.0" encoding='utf-8'?>
<skins xmlns="http://myfaces.apache.org/trinidad/skin">
  <skin>
    <id>myskin.desktop</id>
    <family>myskin</family>
    <extends>fusion.desktop</extends>
    <render-kit-id>org.apache.myfaces.trinidad.desktop</render-kit-id>
    <style-sheet-name>skins/myskin/myskin.css</style-sheet-name>

    <bundle-name>oracle.iam.consoles.faces.resources.AdfComponentsMessageBundle</bundle-name>
  </skin>
</skins>
```

2. Register the new "myskin" in both the admin.war/WEB-INF/trinidad-config.xml file and the iam-consoles-faces.war/WEB-INF/trinidad-config.xml file, as shown:

```
<?xml version="1.0" encoding='utf-8'?>
<trinidad-config xmlns="http://myfaces.apache.org/trinidad/config">
  <skin-family>myskin</skin-family>
</trinidad-config>
```

3. Create myskin.css at both the admin.war/skins/myskin/myskin.css and iam-consoles-faces.war/skins/myskin/myskin.css files.
4. Put stylesheet elements that are required to be overridden from the defaults. For example, to change the branding text color, add the following:

```
.AFBrandingBarTitle, .xdj
{
    color:#800080;
}
```

5. Redeploy (or update) the Oracle Identity Manager deployment through the Oracle WebLogic Server Administration Console.

To change the branding information as an alternative to the method in "[Branding Customization](#)" on page 28-1, create a custom skin and use the appropriate style classes given in the URL in "[Introduction to the Style Sheets](#)" on page 28-10.

28.2.3 Style Sheets in Transitional UI

This section describes how to customize the look and feel of the transitional UI in the Administrative and User Console. You do this by editing the cascading style sheets.

This section contains the following topics:

- [Files to Modify](#)
- [Customizing the Appearance of the Transitional UI](#)

28.2.3.1 Files to Modify

To customize colors, fonts, and alignments in the transitional UI, edit the `/xlWebApp/css/Xellerate.css` file or the locale-specific cascading style sheets.

28.2.3.2 Customizing the Appearance of the Transitional UI

The `Xellerate.css` style sheet defines the color, font, point size, and alignment of the transitional UI pages. After you determine how you want to edit the appearance of the transitional UI, perform the following steps:

1. View the source for the page.
2. Determine the style sheet class associated with the element on the page that you want to change.
3. Lookup the style sheet class name within the `Xellerate.css` file.

The `Xellerate.css` file contains context labels. Use these context labels to locate the class to edit when customizing a particular aspect of the transitional UI appearance. In addition, the classes within this file are organized according to the region of the screen, such as header, body, and footer, and the HTML elements they affect, such as links, tables, and check boxes.

28.3 Renaming Button Labels

This section describes how you can rename button labels in different consoles:

- [Identity Administration](#)
- [Other Consoles](#)
- [Transitional UI Pop-ups](#)

28.3.1 Identity Administration

The Identity Administration is made up of four main feature areas, each with their own resource bundle:

- **User Management:**

```
admin.war/WEB-INF/lib/IdentityTaskFlow.jar/oracle/iam/identitytaskflow/resources/IdentityUIBundle_LOCALE.properties
```

- **Organization Management:**

```
admin.war/WEB-INF/lib/orgmgmtTF.jar/oracle/iam/consoles/orgmgmt/tf/resources/orgmgmt-ui_LOCALE.properties
```

- **Role Management:**

```
admin.war/WEB-INF/lib/rolemgmtTF.jar/oracle/iam/consoles/rolemgmt/tf/resources/rolemgmt-ui_LOCALE.properties
```

- Policy Management:

```
admin.war/WEB-INF/lib/OESOIMTaskFlows.jar/oracle/iam/consoles/oesoim/tf/resources/oes-oim-ui_LOCALE.properties
```

For example, for the Policy Management UI, the following properties are set for button labels in the corresponding properties file:

```
button.advanced=Advanced
button.apply=Apply
button.apply.accessKey=A
button.back=Back
button.back.accessKey=B
button.cancel=Cancel
button.cancel.accessKey=C
button.edit_attribute=Edit Attributes
button.edit_attribute.accessKey=E
button.finish=Finish
button.finish.accessKey=F
button.next=Next
button.next.accessKey=N
button.ok=Ok
button.ok.accessKey=O
button.revert=Revert
button.revert.accessKey=R
button.save=Save
button.save.accessKey=S
```

28.3.2 Other Consoles

The common button labels for unauthenticated self service, authenticated Self Service, and Advanced Administration are set in the resource bundle

```
iam-consoles-faces.war/WEB-INF/lib/iam-consoles-faces.jar/oracle/iam/consoles/faces/resources/Common_LOCALE.properties
```

For example:

```
text_cancel = Cancel
description_cancel = Cancel

text_submit = Submit
description_submit = Submit.

text_reset = Reset
description_reset = Reset

text_previous = Back
description_previous = Back

text_next = Next
description_next = Next

text_refresh = Refresh
description_refresh = Refresh

text_perform = Perform
description_perform = Perform
```

```
text_confirm = OK
description_confirm = Confirm
```

Some button labels on the Self Service are also used from the `iam-consoles-faces.war/WEB-INF/lib/OIMUI.jar/oracle/iam/selfservice/self/agency/resources/Agent_LOCALE.properties` file.

In Advanced Administration, many button labels are used from the `Agent_LOCALE.properties` file of the respective feature. For example, for the buttons in request, the file to use is `Agent_LOCALE.properties` in the `iam-consoles-faces.war/WEB-INF/lib/OIMUI.jar/oracle/iam/request/agency/resources/` directory.

Some buttons in this file are as follows:

```
Intent[BACK].name =Back
Intent[BACK].description =Back
Intent[NEXT].name =Next
Intent[NEXT].description =Next
Intent[ADD].name=Add
Intent[CANCEL].name=Cancel
Intent[CLEAR].name=Clear
Intent[SAVE].name=Save
Intent[APPLY].name=Apply
Intent[REVERT].name=Revert
Intent[PREV_RES].name=Previous Resource
Intent[NEXT_RES].name=Next Resource
Intent[CLOSE_REQ].name=Close Request
Intent[SEARCH_LKQ].name=Search
Intent[REFRESH].name=Refresh
```

Following is a list of the Agent properties file locations within the OIMUI.jar for some of the main features:

- **System Configuration:** `oracle/iam/conf/agency/resources`
- **User Configuration:** `oracle/iam/configservice/agency/resources`
- **Scheduler:** `oracle/iam/features/scheduler/agency/resources`
- **Notification Templates:** `oracle/iam/features/notification/agency/resources`
- **Reconciliation:** `oracle/iam/reconciliation/agency/resources`
- **Request:** `oracle/iam/request/agency/resources`
- **Request Templates:** `oracle/iam/requesttemplate/agency/resources`
- **Authenticated Self Service:** `oracle/iam/selfservice/self/agency/resources`
- **Unauthenticated Self Service:** `oracle/iam/selfservice/used/agency/resources`
- **Self Service Task List:** `oracle/iam/tasklist/agency/resources`

28.3.3 Transitional UI Pop-ups

This section describes how to customize the descriptive text and labels that appear on the transitional UI pages of the Administrative and User Console. It also describes how to edit the error messages that appear under the specific conditions that generate them.

You can customize the following text:

- Descriptions of procedures, such as instructional text

- Labels for pages
- Labels for fields within pages
- Labels that appear on buttons
- Labels for links to other pages
- Copyright dates and information

This section contains the following topics:

- [Files to Modify](#)
- [Customizing Descriptive Text and Labels](#)

28.3.3.1 Files to Modify

To customize the text in the transitional UI pages of the Administrative and User Console, edit the `xIWebApp\WEB-INF\classes\xIWebAdmin_LOCALE.properties` file or the locale-specific properties file.

There are other properties files, for example `xIRichClient.properties` for Workflow Visualizer and Workflow Designer, and `xIDefaultAdmin.properties` that is used to include generic (not specific to locale) properties such as image references.

28.3.3.2 Customizing Descriptive Text and Labels

The text displayed in the console is stored in the `xIWebAdmin_LOCALE.properties` file. For any label, procedural instruction, or error message that you want to edit, perform the following steps:

1. Access the JSP page in which the text is displayed.
2. Reference the property associated with the text.
3. Lookup the property name in the `xIWebAdmin_LOCALE.properties` file.

Properties that control the text on more than one page, or that are associated with multiple product functionalities, are listed in one of the GLOBAL sections of the file, such as GLOBAL messages. These properties are also divided by type-specific labels, such as messages, buttons, and labels, to organize the groups of the properties.

The properties that control the text that is displayed in more specific contexts of the product functionality, such as create user and self-registration, are listed in sections labeled in a function-specific manner, such as MANAGE USER labels.

The width of the labels in the Workflow Visualizer are calculated by converting pixel lengths to character lengths. An incorrect conversion can result in truncated text or extra white space surrounding the label. The `xIRichClient.properties` file, or locale-specific file such as `xIRichClient_jp.properties` for Japanese, contains a property named `global.workflowRenderer.labelWidthFactor`. This property is used in the conversion of label text from pixel length to character length. To modify the width of labels in the Workflow Visualizer, modify the integer value that is assigned to this property. A higher integer will increase label widths, and a lower integer will decrease them.

28.4 Working with Menus and Tabs

This section explains how to add and update menus and tabs in the various sections of the Administrative and User Console.

- [Oracle Identity Administration](#)

- [Other Consoles](#)

28.4.1 Oracle Identity Administration

The top level tabs, Administration and Policy, are configured in the IDM Shell configuration file: `admin.war/WEB-INF/idmshell-config.xml`. For information about IDM Shell, refer to the following URL:

<http://www.oracle.com/technetwork/developer-tools/adf/uishell-093084.html>

The configuration file looks like this:

```
<taskflows>
  <taskflow id="_oes_oim_lhs" closeable="false" indialog="false"
    taskFlowId="/taskflows/brsr/BrowseSearch-TF.xml#BrowseSearch-TF">
    <name>Browse And Search</name>
    <description>Browse And Search</description>
  </taskflow>
  <taskflow id="_oes_oim_rhs" closeable="false" indialog="false"
    taskFlowId="/taskflows/welcome/Welcome-TF.xml#Welcome-TF">
  </taskflow>
</taskflows>
<modules>
  <module id="oes_oim_mgr" helpTopicId="oim_ia_policy">
    <lhs-area>
      <taskflow refId="_oes_oim_lhs"/>
    </lhs-area>
    <default-taskflow-list>
      <taskflow refId="_oes_oim_rhs"/>
    </default-taskflow-list>
  </mLoading...odule>
</modules>
```

The configuration inside the `<taskflow>` tabs represents ADF task flows. For additional information on ADF menus, see "Using Menus, Toolbars, and Toolboxes" in the *Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework* in the following URL:

http://download.oracle.com/docs/cd/E15523_01/web.1111/b31973/toc.htm

This example shows the steps needed to configure a new tab for the Identity Administration:

1. Open the `OIM_INSTALL/middleware/Oracle_IDM1/server/apps/oim.ear/admin.war/WEB-INF/idmshell-config.xml` file in a text editor.
2. In the `<taskflows>` tag, add a new taskflow to the list of configured task flows, giving it a unique ID. For illustration, this "new_tab" taskflow configuration will mimic that of the "Welcome" tab (`admin_welcome`):

```
<taskflow id="new_tab" closeable="false" indialog="false"

taskFlowId="/taskflows/adminwelcome/AdminWelcome-taskflow.xml#AdminWelcome-task
flow">
  </taskflow>
```

3. Scrolling down in the same file, locate the `<modules>` tag, and within that the `<module>` with ID of "admin".

- To the `<default-taskflow-list>` add a new `<taskflow>`, setting its `refId` to the taskflow ID we added earlier, namely "new_tab".

```
<module id="admin">
  <lhs-area>
    <taskflow refId="usr_lhs"/>
  </lhs-area>
  <default-taskflow-list>
    <taskflow refId="admin_welcome"/>
    <taskflow refId="new_tab"/>
  </default-taskflow-list>
</module>
```

Note: Modules refer to top-level tabs, and Default-Taskflow-List entries refer to second-level tabs.

- Assign text for the new tab must in a properties file. Extract the file `OIM_INSTALL/middleware/Oracle_IDM1/server/apps/oim.ear/admin.war/WEB-INF/lib/IdentityTaskFlow.jar` to a temporary directory [`WORK_DIR`] using a tool such as unzip.
- To change the header of the Identity Administration, edit the file `WORK_DIR/oracle/iam/identitytaskflow/resources/IdentityUIBundle_en.properties` (or other language-specific translation files as appropriate).
- Add new property named "TaskFlow[new_tab].name" and assign it an appropriate name, such as "New Tab".
- Add new property named "TaskFlow[new_tab].description" and assign it an appropriate description, such as "New Tab Description".

```
TaskFlow[new_tab].name=New Tab
TaskFlow[new_tab].description=New Tab Description
```

Note: The value within the square brackets must match the taskflow id. This is how these property values are associated with the given tab and taskflow.

- Change directory to `WORK_DIR` and repackage the files and directories under `WORK_DIR` by using the jar syntax:

```
jar -cf IdentityTaskFlow.jar *
```

- Place this new `IdentityTaskFlow.jar` file back in `OIM_INSTALL/middleware/Oracle_IDM1/server/apps/oim.ear/admin.war/WEB-INF/lib/`. Be sure to back up the original file before overwriting it.
- Redeploy (or update) the Oracle Identity Manager deployment through the Oracle WebLogic Server Administration Console.

28.4.2 Other Consoles

Self Service provides the ability to hide top-level tabs. In addition, Self Service UI also allows adding new custom tabs. The custom tabs can be associated with custom jsff page implementation. Hiding of the top level menus and tabs on Self Service is

explained in ["Disabling Features"](#) on page 28-18. Adding new tabs and associating custom jsff on Self Service is described in ["Adding Custom ADF Tabs to Self Service"](#) on page 29-1.

28.5 Disabling Features

This section explains the techniques by which features can be enabled or disabled in the different consoles and user interfaces:

- [Disabling Access to Features Through the Authorization Policies](#)
- [Other Administration Features](#)
- [Other Consoles](#)

28.5.1 Disabling Access to Features Through the Authorization Policies

Access to many of the Identity Administration, authenticated Self Service, and Advanced Administration features is controlled by leveraging the integration between Oracle Identity Manager and Oracle Entitlements Server (OES). For features and actions managed within OES, the recommended way to disable a feature/action is to remove access to the feature by controlling the policies available to a user.

See Also: "Managing Authorization Policies" in the *Oracle Fusion Middleware User's Guide for Oracle Identity Manager* for more details on creating and modifying authorization policies

For example, the users with the IT Executive - Outbound Services role in the Customer Support organization must be allowed only to search for all users and view the contact information for each user. Therefore, the buttons for creating and modifying users or roles in the left pane of the Oracle Identity Manager Self Service must be hidden for these users.

You can modify the authorization policies to hide or disable a few action buttons on the left pane of the Administrative or User Console, for which the user does not have permissions. To do so:

1. Create an authorization policy for User Management. For details about creating an authorization policy, see "Creating an Authorization Policy for Role Management" in the *Oracle Fusion Middleware User's Guide for Oracle Identity Manager*.
2. Click the **Permissions** tab and set the Permissions as Search User and View User Detail.
3. For the View User Detail permission, select the attributes for contact information.
4. Click the **Data Constraints** tab and set the data constraints as member of the organization, for example, Customer Support.
5. Click the **Assignment** tab. This tab displays the roles that are assigned to this policy. Add the role, for example IT Executive - Outbound Services. For details about adding the role, see the step 10 of "Creating an Authorization Policy for Role Management" in the *Oracle Fusion Middleware User's Guide for Oracle Identity Manager*.
6. Click **Apply** to save changes.

Enforcement by using the authorization service API must be placed around the User Search and View Detail action menu, buttons, or links. If access for the operation is denied to the user, then the action is not displayed or is displayed as disabled.

28.5.2 Other Administration Features

If the feature cannot be disabled by using authorization policies, then another option is to disable the corresponding ADF action tag. Most ADF action tags can be disabled or rendered invisible.

For details, see:

- The part titled Using ADF Faces Components in the *Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework* in the following URL:

http://download.oracle.com/docs/cd/E15523_01/web.1111/b31973/toc.htm

- The ADF Tag Guide, which is available from the Oracle ADF Components Home Page.

28.5.3 Other Consoles

Features from the Self Service can be removed by editing the `iam-consoles-faces.war/WEB-INF/faces-config-self.xml` file.

For example, to disable the **Profile** tab by hiding it in the Self Service, comment the following in the file:

- The managed bean definition corresponding to this functionality:

```
<!--managed-bean>
<managed-bean-name>myProfilePrimaryOperation</managed-bean-name>
....
<managed-bean-->
```

- The map entry corresponding to this functionality:

```
<!--map-entry>
<key>#{myProfilePrimaryOperation.entityTypeId}</key>
<value>#{myProfilePrimaryOperation}</value>
</map-entry-->
```

28.6 Adding or Deleting Columns in Console Tables

This section explains how you can add or remove searchable columns in the UI. It contains these topics:

- [Identity Administration](#)
- [Transitional UI](#)

28.6.1 Identity Administration

Tables in the Identity Administration use standard ADF features. Many Advanced Search pages in the UI have an Add Field button so that the user can select from additional searchable attributes. To allow searching for users, roles, and organizations, the attribute needs to be marked as searchable in the schema.

See Also: The *ADF Tag Guide*, which is available from the Oracle ADF Components Home Page.

[Table 28-1](#) lists the jsff files with corresponding code fragments that you can modify for customizing the tables in Identity Administration:

Table 28–1 Files to Modify For Customizing Tables in Identity Administration

jsff File	Code Fragment
admin.war/WEB-INF/lib/IdentityTaskflow.jar/taskflows/advsearch/AdvancedSearchview.jsff	<pre> <af:table var="row" rowBandingInterval="0" id="t1" binding="#{pageFlowScope.userAdvancedSearchBean.table}" value="#{pageFlowScope.userAdvancedSearchBean.searchResult}" emptyText="#{resUI.empty_table_string}" columnStretching="none" rowSelection="multiple" fetchSize="#{pageFlowScope.userAdvancedSearchBean.fetchSize}" selectionListener="#{pageFlowScope.userAdvancedSearchBean.tableLi stener.processSelection}" sortListener="#{pageFlowScope.userAdvancedSearchBean.sortListener .processSort}" contextMenuId="::contextPopup2"> <af:forEach items="#{pageFlowScope.userAdvancedSearchBean.columns}" var="def" varStatus="defIndex"> <c:choose> <c:when test="\${defIndex.index == 0}"> <af:column headerText="#{pageFlowScope.userAdvancedSearchBean.advancedSearch AttrsTranslation[def.name]}" sortable="true" id="c1" headerNoWrap="true" sortProperty="#{def.name}"> <af:commandLink text="#{row[def.name]}" id="ot2" actionListener="#{pageFlowScope.userAdvancedSearchBean.userLinkLi stener.processAction}"/> </af:column> </c:when> <c:otherwise> <af:column headerText="#{pageFlowScope.userAdvancedSearchBean.advancedSearch AttrsTranslation[def.name]}" sortable="true" id="c2" headerNoWrap="true" sortProperty="#{def.name}"> <af:outputText value="#{row[def.name]}" id="ot3"/> </af:column> </c:otherwise> </c:choose> </af:forEach> </af:table> </pre>

The `#{pageFlowScope.userAdvancedSearchBean.columns}` returns the list of attributes that are configured for the advanced search results attributes. If you want to change the attributes in the advanced search table for users, then change the attributes from the Advanced Console, User Configuration, Search Configuration, Search Results Table Configuration. The first attribute in the Search Results Table Configuration will be displayed as a link.

Table 28–1 (Cont.) Files to Modify For Customizing Tables in Identity Administration

jsff File	Code Fragment
admin.war/WEB-INF/lib/IdentityTaskflow.jar/taskflows/modifyuser/ModifyUserview.jsff	<p>The Roles tab:</p> <pre data-bbox="634 302 1448 1461"> <af:treeTable binding="#{pageFlowScope.modifyUserBean.roleTreeTable}" value="#{pageFlowScope.modifyUserBean.treeModel}" var="role" rowBandingInterval="1" id="ROLES_DATA" rowSelection="single" columnStretching="last" fetchSize="9" contentDelivery="immediate" selectionListener="#{pageFlowScope.modifyUserBean.roleSelected}" immediate="true" width="100%"> <f:facet name="nodeStamp"> <af:column sortable="true" width="350" sortProperty="roleDisplayName" headerText="#{resUI.role_display_name_text}" id="c1"> <!--<af:group>--> <!--<af:image source="/images/group_ena.png"/> --> <af:outputText value="#{role.roleDisplayName}" id="ot52"/> <!--</af:group>--> </af:column> </f:facet> <af:column sortable="false" width="250" headerText="#{resUI.role_name_text}" align="start" id="c72"> <af:outputText value="#{role.roleName}" id="ot12"/> </af:column> <af:column sortable="false" width="250" headerText="#{resUI.role_descripetion_text}" align="start" id="c7"> <af:outputText value="#{role.description}" id="ot11"/> </af:column> <af:column sortable="false" headerText="#{resUI.role_namespace_text}" align="start" id="c71"> <af:outputText value="#{role.roleNameSpace}" id="ot111"/> </af:column> <!--af:column sortable="false" headerText="#{resUI.role_assigned_on_text}" align="start" id="c6"> <af:outputText value="#{role.provisionedOn}" id="ot1"/> </af:column--> </af:treeTable> </pre> <p>The Roles tab contains static columns that can be rearranged or removed based on your requirement. The role objects are instances of <code>oracle.iam.identitytaskflow.common.model.vo.RoleDetailVO</code> with the following methods:</p> <pre data-bbox="634 1598 1448 1822"> public String getRoleName() public String getProvisionedOn() public String getMembershipType() public String getDescription() public List<RoleDetailVO> getChildren() public String getRoleKey() public String getRoleDisplayName() public String getRoleNameSpace() </pre>

Table 28–1 (Cont.) Files to Modify For Customizing Tables in Identity Administration

jsff File	Code Fragment
admin.war/WEB-INF/lib/IdentityTaskflow.jar/taskflows/modifyuser/ModifyUserview.jsff	<pre> The Resources tab: <af:table var="row" columnStretching="last" summary="#{resUI.resources_table_summary_text}" contentDelivery="immediate" binding="#{pageFlowScope.modifyUserBean.resourceTable}" fetchSize="60" id="t3" emptyText="#{resUI.empty_table_string}" value="#{pageFlowScope.modifyUserBean.resources}" rowSelection="single" selectionListener="#{pageFlowScope.modifyUserBean.showSelectedResourceDetailsAction}"> <af:column sortable="true" headerText="#{resUI.resource_name_text}" align="start" id="c6"> <af:commandLink id="com1" text="#{row.resourceName}" partialSubmit="true" actionListener="#{pageFlowScope.modifyUserBean.showResourceDetailsReadOnlyAction}" disabled="#{row.viewActionDisable}"/> </af:column> <af:column sortable="false" headerText="#{resUI.resource_type_text}" align="start" id="c3"> <af:outputText value="#{row.type}" id="ot6"/> </af:column> <af:column sortable="false" headerText="#{resUI.resource_status_text}" align="start" id="c4"> <af:outputText value="#{row.status}" id="ot7"/> </af:column> <af:column sortable="false" headerText="#{resUI.resource_provisioned_text}" align="start" id="c5"> <af:outputText value="#{row.provisionedOn}" id="ot8"/> </af:column> <af:column sortable="false" headerText="#{resUI.resource_request_key_text}" align="start" id="c52"> <af:outputText value="#{row.requestId}" id="ot82"/> </af:column> <af:column sortable="false" headerText="#{resUI.resource_service_account}" align="start" id="c53"> <af:outputText value="#{row.serviceAccount}" id="ot84"/> </af:column> <af:column sortable="false" headerText="#{resUI.resource_description_text}" align="start" id="c51"> <af:outputText value="#{row.descriptiveData}" id="ot81"/> </af:column> </af:table> </pre>

The Resources tab contains static columns that can be rearranged or removed based on your requirement. The row objects are instances of `oracle.iam.identitytaskflow.common.model.vo.ResourceDetailVO` and the following methods:

```

public String getResourceName()
public String getStatus()
public String getProvisionedOn()
public String getType()
public String getProcessInstanceKey()
public String getResourceUserKey()

```


Table 28–1 (Cont.) Files to Modify For Customizing Tables in Identity Administration

jsff File	Code Fragment
admin.war/WEB-INF/lib/IdentityTaskflow.jar/taskflows/modifyuser/ModifyUserview.jsff	<p>The Proxy tab:</p> <pre> <af:table var="row" rowBandingInterval="0" id="t5" contentDelivery="lazy" binding="#{pageFlowScope.modifyUserBean.proxyTable}" emptyText="#{resUI.empty_table_string}" value="#{pageFlowScope.modifyUserBean.proxyList}" selectionListener="#{pageFlowScope.modifyUserBean.proxySelectionA ction}" columnStretching="last" rowSelection="single"> <af:column sortable="true" headerText="#{resUI.proxy_name_text}" id="c9" sortProperty="proxyName"> <af:commandLink text="#{row.proxyName}" id="ot4" actionListener="#{pageFlowScope.modifyUserBean.openProxyDetailPag eAction}"/> </af:column> <af:column sortable="false" headerText="#{resUI.proxy_start_date_text}" id="c13"> <af:outputText value="#{row.startDate}" id="ot2"/> </af:column> <af:column sortable="false" headerText="#{resUI.proxy_end_date_text}" id="c10"> <af:outputText value="#{row.endDate}" id="ot14"/> </af:column> <af:column sortable="false" headerText="#{resUI.proxy_status_text}" id="c11"> <af:outputText value="#{row.status}" id="ot13"/> </af:column> <af:column sortable="false" headerText="#{resUI.proxy_relationship_text}" id="c12"> <af:outputText value="#{row.relationship}" id="ot15"/> </af:column> <af:column sortable="false" headerText="#{resUI.proxy_last_updated_text}" id="c8"> <af:outputText value="#{row.lastUpdated}" id="ot10"/> </af:column> </af:table> </pre>

The Proxy tab contains static columns that can be rearranged or removed based on your requirement. The row objects are instances of `oracle.iam.identitytaskflow.common.model.vo.ProxyDetailVO` and the following methods:

```

public String getStartDate()
public String getEndDate()
public String getStatus()
public String getRelationship()
public String getLastUpdated()
public String getProxyName()
public String getProxyKey()
public String getProxyNameUserKey()
public String getLoginUserKey()
public String getLoginUserName()
public String getRequestId()
public String getServiceAccount()

```

Table 28–1 (Cont.) Files to Modify For Customizing Tables in Identity Administration

jsff File	Code Fragment
admin.war/WEB-INF/lib/IdentityTaskflow.jar/taskflows/modifyuser/ModifyUserview.jsff	<p>The Direct Reports tab:</p> <pre> <af:table value="#{pageFlowScope.modifyUserBean.reportList}" var="row" fetchSize="60" binding="#{pageFlowScope.modifyUserBean.directReportiesTable}" emptyText="#{resUI.empty_table_string}" rowSelection="single" id="t6" selectionListener="#{pageFlowScope.modifyUserBean.directReportSel ectedAction}" columnStretching="last"> <af:column sortProperty="Name" sortable="false" headerText="#{resUI.username_header_text}" id="column1"> <af:outputText value="#{row.userDisplayName}" id="outputText1"/> </af:column> <af:column sortProperty="Name" sortable="false" headerText="#{resUI.userlogin_header_text}" id="column2"> <af:outputText value="#{row.username}" id="outputText2"/> </af:column> <af:column sortProperty="Status" sortable="false" headerText="#{resUI.status_header_text}" id="column4"> <af:outputText value="#{row.identityStatus}" id="outputText3"/> </af:column> <af:column sortProperty="Org" sortable="false" headerText="#{resUI.organization_header_text}" id="column5"> <af:outputText value="#{row.organization}" id="outputText4"/> </af:column> </af:table> </pre>

The Direct Reports tab contains static columns that can be rearranged or removed based on your requirement. The row objects are instances of `oracle.iam.identitytaskflow.common.model.vo.ReportDetailVO` and the following methods:

```

public String getUsername()
public String getFirstName()
public String getLastName()
public String getStatus()
public String getOrganization()
public String getEmail()
public String getPhone()
public String getGlobalID()
public String getIdentityStatus()
public String getAccountStatus()
public String getUserDisplayName()

```

Table 28–1 (Cont.) Files to Modify For Customizing Tables in Identity Administration

jsff File	Code Fragment
admin.war/WEB-INF/lib/IdentityTaskflow.jar/taskflows/modifyuser/ModifyUserview.jsff	<p>The Requests tab:</p> <pre data-bbox="634 302 1448 1346"> <af:table var="row" rowBandingInterval="1" id="t8" columnStretching="column:column7" rowSelection="single" emptyText="#{resUI.empty_table_string}" value="#{pageFlowScope.modifyUserBean.requestList}" binding="#{pageFlowScope.modifyUserBean.requestsTable}"> <af:column sortable="true" sortProperty="requestID" headerText="#{resUI.request_id_header_text}" id="c14"> <af:commandLink text="#{row.requestID}" id="c11" actionListener="#{pageFlowScope.modifyUserBean.showSelectedReques tDetailsAction}"/> </af:column> <af:column sortable="true" sortProperty="modelName" headerText="#{resUI.request_model_name_header_text}" id="c17" noWrap="false"> <af:outputText value="#{row.modelName}" id="ot18"/> </af:column> <af:column id="c18" headerText="#{resUI.request_status_text}" noWrap="false"> <af:outputText value="#{row.status}" id="ot23"/> </af:column> <af:column sortable="true" sortProperty="requestedBy" headerText="#{resUI.request_requested_by_header_text}" id="c15"> <af:outputText value="#{row.requestedBy}" id="ot19"/> </af:column> <af:column sortable="false" headerText="#{resUI.request_parent_id_header_text}" id="c16"> <af:outputText value="#{row.parentID}" id="ot20"/> </af:column> <af:column sortable="false" headerText="#{resUI.request_date_requested_header_text}" id="column6"> <af:outputText value="#{row.dateRequested}" id="ot21"/> </af:column> </af:table> </pre> <p>The Requests tab contains static columns that can be rearranged or removed based on your requirement. The row objects are instances of <code>oracle.iam.identitytaskflow.common.model.vo.RequestDetailVO</code> and the following methods:</p> <pre data-bbox="634 1507 1448 1682"> public String getRequestID() public String getRequestedBy() public String getParentID() public String getDateRequested() public String getModelName() public String getStatus() </pre>

Table 28–1 (Cont.) Files to Modify For Customizing Tables in Identity Administration

jsff File	Code Fragment
admin.war/WEB-INF/lib/IdentityTaskflow.jar/taskflows/rolepicker/RolePicker.jsff	<pre> <af:table binding="#{pageFlowScope.rolePickerBean.table}" value="#{pageFlowScope.rolePickerBean.results}" var="row" emptyText="#{resUI.empty_table_string}" id="t1" inlineStyle="width:inherit;" rowSelection="multiple" columnBandingInterval="1" columnStretching="first" fetchSize="40" selectionListener="#{pageFlowScope.rolePickerBean.roleSelectAction}"> <af:column headerText="" id="c0" rowHeader="true" width="18px"/> <af:column headerText="#{resUI.role_display_name_text}" id="c3" headerNoWrap="true"> <af:outputText value="#{row.roleDisplayName}" id="ot5"/> </af:column> <af:column headerText="#{resUI.role_namespace_text}" id="c1" headerNoWrap="true"> <af:outputText value="#{row.roleNameSpace}" id="ot3"/> </af:column> <af:column headerText="#{resUI.role_name_text}" id="c2" headerNoWrap="true"> <af:outputText value="#{row.roleName}" id="ot4"/> </af:column> </af:table> </pre>

The columns of the role picker popup can be rearranged or removed based on your requirement. The role objects are instances of `oracle.iam.identitytaskflow.common.model.vo.RoleDetailVO` with the following methods:

```

public String getRoleName()
public String getProvisionedOn()
public String getMembershipType()
public String getDescription()
public List<RoleDetailVO> getChildren()
public String getRoleKey()
public String getRoleDisplayName()
public String getRoleNameSpace()

```

Table 28–1 (Cont.) Files to Modify For Customizing Tables in Identity Administration

jsff File	Code Fragment
admin.war/WEB-INF/lib/IdentityTaskflow.jar/taskflows/search/SimpleSearchview.jsff	<pre> <af:table var="row" rowBandingInterval="0" id="t4" binding="#{pageFlowScope.simpleSearchBean.table}" value="#{pageFlowScope.simpleSearchBean.searchResult}" emptyText="#{resUI.empty_table_string}" columnStretching="last" rowSelection="multiple" fetchSize="#{pageFlowScope.simpleSearchBean.tableFetchSize}" selectionListener="#{pageFlowScope.simpleSearchBean.processSelection}" sortListener="#{pageFlowScope.simpleSearchBean.tableSortListener.processSort}"> <af:column headerText="#{pageFlowScope.simpleSearchBean.headerText}" sortable="true" id="c1" headerNoWrap="true" sortProperty="#{pageFlowScope.simpleSearchBean.sortProperty}"> <af:commandLink text="#{row[pageFlowScope.simpleSearchBean.sortProperty]}" id="ot2" actionListener="#{pageFlowScope.simpleSearchBean.userLinkListener.processAction}" /> </af:column> </af:table> </pre>

The

oracle.iam.identitytaskflow.backing.taskflows.search.SimpleSearchView getSearchResult() method returns a list of strings to display in the simple search table for users. The attribute whose value has to be displayed is determined in the following order from the attributes defined in the Advanced Administration, User Configuration, Search Configuration, Simple Search Attributes:

1. Display name
2. User Login
3. User Key
4. The first attribute entry in the Advanced Administration, User Configuration, Search Configuration, Simple Search Attributes

Table 28–1 (Cont.) Files to Modify For Customizing Tables in Identity Administration

jsff File	Code Fragment
admin.war/WEB-INF/lib/OESOIMTaskFlows.jar/taskflows/advsrc/AdvancedSearch.jsff	<p>On the Authorization Policy tab, the advanced search feature is managed through the admin.war/WEB-INF/lib/OESOIMTaskFlows.jar/taskflows/advsrc/AdvancedSearch.jsff file. The contents of this file is as shown:</p> <pre> <af:table var="row" columnStretching="last" id="shTable" rowSelection="single" columnSelection="single" contentDelivery="immediate" summary="#{props['padvsrc.tabs.browse.columns.summary']}" binding="#{pageFlowScope.advancedSearchBean.searchTableBinding}" selectionListener="#{pageFlowScope.advancedSearchBean.selectSearchRow}" emptyText="#{props['brsr.tabs.advsrc.emptyText']}" shortDesc="#{props['brsr.tabs.advsrc.shortDesc']}" value="#{pageFlowScope.advancedSearchBean.searchResultsModel}" partialTriggers=":q2" columnBandingInterval="1"> ... <af:column sortable="true" sortProperty="authzPolicy.displayName" headerText="#{props['padvsrc.columns.policy_name.headerText']}" width="200" noWrap="false" align="start" id="c1"> <af:outputText value="#{row.authzPolicy.displayName}" id="ot2"/> </af:column> The row objects are instances of oracle.iam.consoles.oesoim.tf.base.PolicyTableModel.SearchWrapper with getter methods: public AuthzPolicy getAuthzPolicy() public boolean getHasAssignment() public boolean getHasAssignment() public String getAssignment() public boolean getHasPermissions() public String getFirstPermission() public String getPermissions() </pre>

28.6.2 Transitional UI

In the transitional UI pages of the Administrative and User Console, when you click a menu item to perform tasks, such as managing access policies, a search page is displayed. For example, when you click the Manage link under the Access Policies menu item, the Manage Access Policies page is displayed with two drop-down menus for searching access policies. You can customize the number of drop-down menus, and what the items in the drop-down menus are.

When the search results display, you can determine the maximum number of rows in the results table displayed on each page. After a user selects an item from the results table, a detail page is displayed such as the Resource Detail page. The detail page contains an additional details menu. You can customize the items in these menus.

This section contains the following topics:

- [Customizing Search Drop-Down Item](#)
- [Customizing Number of Search Drop-Down Items and Search Results](#)

28.6.2.1 Customizing Search Drop-Down Item

Use the Design Console to change the lookup codes for search pages and additional details. To customize drop-downs:

1. Log in to the Design Console.
2. Open the Lookup Definition form by navigating to Administration, then to Lookup Definition.
3. Search to locate the desired lookup definition.

Tip: For your search criteria, use `lookup.webclient*` search to find the search pages, or `*additional_details` to find the additional details.

4. Make the desired changes to the lookup codes to set the options displayed in the drop-down menu for each search page.
 - The Code Key is the metadata for each column.
 - The Decode value is what is displayed in the Administrative and User Console.
 - The order the items appear in the Code Key list are the order they appear in the Administrative and User Console drop-down list. If you delete an entry and add it back, it is displayed last in the list.
5. Save your changes.

28.6.2.2 Customizing Number of Search Drop-Down Items and Search Results

To change the number of drop-down menus, and the maximum number of search results on each page, edit the `xlDefaultAdmin.properties` file.

To set the number of drop-down menus:

1. Open the `xlDefaultAdmin.properties` file.
2. Locate the property from [Table 28-2](#), and edit it as required.

Table 28-2 *Properties that Determine the Number of Menus on a Search Page*

Property Name	Default	Page
<code>global.property.numsearchaccesspolicyfields</code>	2	Access Policies
<code>global.property.numsearchresourcefields</code>	2	Search Resources
<code>global.property.numsearchattestationprocessfields</code>	3	Attestation Process

3. To change the maximum number of search results on each page, change the value of the property `global.displayrecordNum.value` to the desired value. The default value is 10.
4. Save the file.
5. Restart Oracle Identity Manager.

28.7 Data Customization

This section explains how you can customize form templates in various consoles. It contains these topics:

- [Advanced Administration](#)
- [Unauthenticated Self Service](#)
- [Authenticated Self Service](#)

28.7.1 Advanced Administration

You can modify request templates to customize the request attributes for self-registration and request forms.

For details, see "Managing Request Templates" in the *Oracle Fusion Middleware User's Guide for Oracle Identity Manager*.

28.7.2 Unauthenticated Self Service

You can customize the template used for self-registration by specifying template attributes in the SelfCreateUserDataSet.xml file. This both renders the attributes and also specifies the restrictions.

28.7.3 Authenticated Self Service

You can configure the user form to show or hide attributes from the My Profile page by means of the User Configuration UI.

The steps needed to display an attribute on the My Profile page are as follows:

1. Review the My Profile page to determine the desired customization.
2. Go to the Advanced Administration, and under Configuration, click **User Configuration**.
3. From the Actions menu, select **User Attributes**.
4. Select the attribute you want to show on the My Profile page, and from the Actions menu, select **Modify Attribute**.
5. Change the visibility to **Yes**, and click **Save**.

The My Profile page will now render this attribute.

Similar steps can be used to hide attributes on the page.

28.8 Injecting Custom URLs

This section explains how to inject customized URLs into the consoles. Topics include:

- [Custom URLs for the Identity Administration](#)
- [Custom URLs for Other Consoles](#)

28.8.1 Custom URLs for the Identity Administration

The Identity Administration has been developed as ADF Task Flows. The .jsff files in the task flows can be modified to inject custom URLs. See the chapter titled Working with Navigation Components in the *Oracle Fusion Middleware Web User Interface*

Developer's Guide for Oracle Application Development Framework in the following Web site:

http://download.oracle.com/docs/cd/E15523_01/web.1111/b31973/toc.htm

28.8.2 Custom URLs for Other Consoles

In the unauthenticated self service, authenticated Self Service, and Advanced Administration, there are two options for injection of custom URLs.

The pages in the unauthenticated Self Service, authenticated Self Service, and Advanced Administration allow adding any custom URLs by editing the corresponding .jspx files. This gives an option to put custom URLs by using ADF tags in places, such as the header.

The files are:

For unauthenticated Self Service:

`iam-consoles-faces.war/pages/USelf.jsx`

For authenticated Self Service:

`iam-consoles-faces.war/pages/Self.jsx`

For Advanced Administration:

`iam-consoles-faces.war/pages/Admin.jsx`

Note: The same files can be used to add additional static text at different locations as permitted by the .jspx files. For example, to add a static text in the header region, identify the appropriate adf facet in the file and use standard adf tags to put such information.

28.9 Changing Popup Properties

The popups in unauthenticated Self Service, authenticated Self Service, and Advanced Administration can be modified to change the properties, such as turning resizing on or off. This can be done by modifying the corresponding .jspx file and making changes to the popup definition.

The files are:

For unauthenticated Self Service:

`iam-consoles-faces.war/pages/USelf.jsx`

For authenticated Self Service:

`iam-consoles-faces.war/pages/Self.jsx`

For Advanced Administration:

`iam-consoles-faces.war/pages/Admin.jsx`

For example, the operational popups, which means popups having some operation unlike confirmation or message popups, search for the definition of POPUP_FORM_0. Inside this definition, add `resize="on"` to the contained `<af:dialog />` definition, as shown:

```
<af:popup id="POPOP_FORM_0" clientComponent="true"
contentDelivery="lazyUncached">
  <af:dialog binding="#{admin.view.popupForms[0].dialog}"
resize="on" clientComponent="true" cancelVisible="false"
...

```

28.10 Customizing the Workflow Designer

There are a number of properties that can be useful in customizing the Workflow Designer user interface based on what a particular locale demands. These properties files are in `xlWebApp\WEB-INF\classes\xlRichClient_LOCALE.properties`.

This section discusses some of the examples of Workflow Designer customization.

The label widths for the task names can be customized by using:

```
global.workflowRenderer.labelWidthFactor=7
```

If in a particular locale the text is seen to be truncated, then this property can be changed accordingly to modify the width of the labels.

Similarly, the width and height of the icons representing the tasks and responses can be controlled by:

```
workflowRenderer.referenceMarker.defaultHeight=22
workflowRenderer.referenceMarker.defaultWidth=22
workflowRenderer.referenceMarker.defaultDistance=6
workflowRenderer.referenceMarker.maxLabelLength=5
```

```
workflowRenderer.response.defaultHeight=20
workflowRenderer.response.defaultWidth=250
workflowRenderer.response.maxLabelLength=25
```

```
workflowRenderer.task.defaultHeight=38
workflowRenderer.task.defaultWidth=38
workflowRenderer.task.maxLabelLength=200
```

In the search functionality at different places in the designer implementation, the number of results could be high in cases where users and groups are searched. To control the number of results to be displayed, the following properties can be used.

```
workflowDesigner.label.limitUserSearchResults=200
workflowDesigner.label.limitGroupSearchResults=200
workflowDesigner.label.limitAdapterSearchResults=200
workflowDesigner.label.limitEmailTemplateSearchResults=200
workflowDesigner.label.limitRulesSearchResults=200
workflowDesigner.label.limitAssignTypeSearchResults=200
workflowDesigner.label.limitDependentDataTaskSearchResults=200
workflowDesigner.label.limitExistingTasksSearchResults=200
```

When the number of results returned are more than these numbers, a message is shown to narrow the search criteria.

Adding Custom ADF Tabs to Self Service

This chapter describes the functionality that allows adding ADF page fragments to the Oracle Identity Manager Self Service.

To enable complex customizations in the Self Service, you can add additional top-level tabs to the right of the Tasks, Requests, and My Profile sections of the Self Service. To do so:

1. Create new .jsff fragment(s) for UI customization.

ADF provides the declarativeComponent tag to embed a separate file in a .jspx file. This technique is used to embed custom page fragments in Self.jsx. The custom file(s) must reside in iam-consoles-faces.war under the /web/pageFragments/ directory. For example, /web/pageFragments/custom/Custom.jsff.

The following is a simple starter example with a page header:

```
<?xml version='1.0' encoding='UTF-8'?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.1"
          xmlns:af="http://xmlns.oracle.com/adf/faces/rich"
          xmlns:f="http://java.sun.com/jsf/core">
  <jsp:directive.page contentType="text/html;charset=utf-8"/>
  <af:componentDef>
    <af:panelStretchLayout id="ps11a" topHeight="auto" startWidth="0px"
                          endWidth="0px" bottomHeight="0px">
      <f:facet name="bottom"/>
      <f:facet name="start"/>
      <f:facet name="end"/>
      <f:facet name="top">
        <af:panelHeader id="ph1a" size="-1" text="">
          <f:facet name="context">
            <af:outputText value="Customize Fragment" id="ot2a"
                          inlineStyle="font-size:15px"/>
          </f:facet>
          <f:facet name="menuBar"/>
          <f:facet name="toolbar"/>
          <f:facet name="legend">
            <af:separator id="s2a"/>
          </f:facet>
          <f:facet name="info"/>
        </af:panelHeader>
      </f:facet>
    </af:panelStretchLayout>
  </af:componentDef>
</jsp:root>
```

-
2. Create managed bean(s) for UI customizations, compile bean in separate .jar file, and add it to iam-consoles-faces.war.

Note: A typical JavaServer Faces application includes one or more managed or backing beans, which are JavaBeans components associated with UI components used in a page. A backing bean defines the UI component properties, each of which is bound to either a component's value or a component instance. A backing bean can also define methods that perform functions associated with a component, including validation, event handling, and navigation processing.

To provide custom business logic, a new managed bean must be created for each custom page. The new managed beans must be compiled into a new .jar file that you can add to iam-consoles-faces.war

3. Add customization page(s) to faces-config-self.xml.

Define the custom pages, as shown:

```
<managed-bean>
  <managed-bean-name>customPage</managed-bean-name>
  <managed-bean-class>
    oracle.iam.consoles.faces.backing.Self$OperationAction
  </managed-bean-class>
  <managed-bean-scope>application</managed-bean-scope>
  <managed-property>
    <property-name>id</property-name>
    <property-class>java.lang.String</property-class>
    <value>customization_page</value>
  </managed-property>
  <managed-property>
    <property-name>pageUrl</property-name>
    <property-class>java.lang.String</property-class>
    <value>../pageFragments/custom/Custom.jsff</value>
  </managed-property>
</managed-bean>
```

Add the custom pages to the primaryOperationsMap, as shown:

```
<managed-bean>
  <managed-bean-name>primaryOperationsMap</managed-bean-name>
  <managed-bean-class>java.util.LinkedHashMap</managed-bean-class>
  <managed-bean-scope>application</managed-bean-scope>
  <map-entries>
    <key-class>java.lang.String</key-class>
<value-class>oracle.iam.consoles.faces.backing.Self$OperationAction</value-class>
  <map-entry>
    <key>#{myTaskPrimaryOperation.entityTypeId}</key>
    <value>#{myTaskPrimaryOperation}</value>
  </map-entry>
  <map-entry>
    <key>#{myRequestPrimaryOperation.entityTypeId}</key>
    <value>#{myRequestPrimaryOperation}</value>
  </map-entry>
  <map-entry>
    <key>#{myProfilePrimaryOperation.entityTypeId}</key>
    <value>#{myProfilePrimaryOperation}</value>
```

```
        </map-entry>
        <!-- New custom fragments -->
        <map-entry>
            <key>#{customPage.id}</key>
            <value>#{customPage}</value>
        </map-entry>
    </map-entries>
</managed-bean>
```

4. Add new managed bean(s) to faces-config-self.xml.

The new managed bean class must be configured in the `oim.ear/iam-consoles-faces.war/WEB-INF/faces-config-self.xml` file, as shown:

```
<managed-bean>
    <managed-bean-name>customBean</managed-bean-name>
    <managed-bean-class>oracle.iam.consoles.faces.custom.CustomBean</managed-bean-c
lass>
    <managed-bean-scope>session</managed-bean-scope>
</managed-bean>
```

5. Add custom properties for tab names to Self.properties, as shown:

Note: The first part of the resource name must match the managed-bean id parameter in step 3.

```
customization_page.text=Custom
customization_page.shortDesc=Custom Description
```

General Customization Concepts

Oracle Identity Manager customization is enabled by the Design Console that lets you deal with configuration and design functions, such as designing forms and workflows and creating and managing adapters. Using the Design Console, you can grant user privileges to work on particular areas of the application configuration.

This chapter discusses the following topics:

- [Rule Elements, Variables, Data Types, and System Properties](#)
- [Service Accounts](#)
- [Design Console Actions](#)

30.1 Rule Elements, Variables, Data Types, and System Properties

The Design Console lets you perform Oracle Identity Manager customization tasks such as adding and modifying rule elements for a rule, creating or editing e-mail definitions, and creating forms. For these customization tasks, you must set parameters, variables, and data types. This section describes these parameters, variables, and data types.

In the Rule Elements tab of the Rule Designer form, you can create and manage elements and nested rules for a rule. [Table 30–1](#) lists the rule elements that can be used to create Oracle Identity Manager rules, by using the Rule Designer form.

Table 30–1 *Rule Elements to Create Oracle Identity Manager Rules*

Type	Sub-Type	Attribute Source	Variable
General	NA	User Profile Data	Email
			End Date
			First Name
			Identity
			Last Name
			Display Name
			Manager
			Middle Name
			Organization Name
			User Role Name
Start Date			

Table 30–1 (Cont.) Rule Elements to Create Oracle Identity Manager Rules

Type	Sub-Type	Attribute Source	Variable
General	NA	User Profile Data	User Type Identity Status User Login Design Console Access Any fields that are displayed in the User Defined Fields region of the User Profile tab of the Users form.
Process Determination	Organization Provisioning	Requester Information	Display Name Email End Date First Name Identity Last Name Manager Full Name Manager Middle Name Organization Name Start Date Identity Status User Role Name User Login Design Console Access Any fields that are displayed in the User Defined Fields region of the User Profile tab of the Users form.
Process Determination	Organization Provisioning	Object Information Request Target Information	Object Name Object Type Organization Customer Type Organization Name Organization Status Parent Organization Any fields that are displayed in the User Defined Fields tab of the Organizations form.

Table 30–1 (Cont.) Rule Elements to Create Oracle Identity Manager Rules

Type	Sub-Type	Attribute Source	Variable
		Object Data Information	Any fields that are displayed in the Additional Columns tab of the Form Designer form for the custom form associated with the resource object.
		Process Data Information	Any fields that are displayed in the Additional Columns tab of the Form Designer form for the custom form associated with the process.
	User Provisioning	Requester Information	Display Name Email End Date First Name Identity Last Name Manager Full Name Manager Middle Name Organization Name User Type Start Date Identity Status User Role Name User Login Design Console Access Any field defined on the FormMetadata.xml user self-registration and user profile modification section with property useInRule set to true
		Object Information	Object Name Object Type
		Request Target Information	Display Name Email End Date First Name Identity

Table 30–1 (Cont.) Rule Elements to Create Oracle Identity Manager Rules

Type	Sub-Type	Attribute Source	Variable
			Last Name Manager Full Name Manager Middle Name Organization Name User Type Start Date Identity Status User Role Name User Login Design Console Access Any field defined on the FormMetadata.xml user self-registration and user profile modification section with property useInRule set to true
Process Determination	User Provisioning	Requester Information; Request Target Information	Any fields that are displayed in the User Defined Fields region of the User Profile tab of the Users form.
		Object Information	Object Name Object Type
		Object Data Information	Any fields that are displayed in the Additional Columns tab of the Form Designer form for the custom form associated with the resource object.
		Process Data Information	Any fields that are displayed in the Additional Columns tab of the Form Designer form for the custom form associated with the process.
Task Assignment	Organization Provisioning; User Provisioning	Task Information	Allow Cancellation while Pending Allow Multiple Instances Assign Task to Manager Disable Manual Insert Task Conditional Task Data Label

Table 30-1 (Cont.) Rule Elements to Create Oracle Identity Manager Rules

Type	Sub-Type	Attribute Source	Variable
			Task Default Assignee
			Task Name
			Task Required for Completion
			Task Sequence
		Process Information	Object Name
			Process Name
			Process Type
		Object Information	Object Name
			Object Type
		Requester Information	Email
			End Date
			First Name
			Identity
Task Assignment	Organization Provisioning; User Provisioning	Requester Information	Display Name
			Email
			End Date
			First Name
			Identity
			Last Name
			Manager Full Name
			Manager
			Middle Name
			Organization Name
			User Type
			Start Date
			Identity Status
			User Role Name
			User Login
			Design Console Access
			Any field that is displayed in the User Defined Fields region of the User Profile tab of the Users form

Table 30–1 (Cont.) Rule Elements to Create Oracle Identity Manager Rules

Type	Sub-Type	Attribute Source	Variable
		Object Data Information	Any field that is displayed in the Additional Columns tab of the Form Designer form for the custom form associated with the resource object
		Process Data Information	Any field that is displayed in the Additional Columns tab of the Form Designer form for the custom form associated with the process
Pre-Populate	Organization Provisioning; User Provisioning	Requester Information	Display Name
			Email
			End Date
			First Name
			Identity
			Last Name
			Manager Full Name
			Manager
			Middle Name
			Organization Name
			User Type
			Start Date
			Identity Status
			User Role Name
			User Login
			Design Console Access
			Any field that is displayed in the User Defined Fields region of the User Profile tab of the Users form
		Request Information	Request Creation Date
			Request ID
			Request Object Action
			Request Priority
			Requestor
		Object Information	Object Name
			Object Type

Table 30–1 (Cont.) Rule Elements to Create Oracle Identity Manager Rules

Type	Sub-Type	Attribute Source	Variable
		Object Data Information	Any field that is displayed in the Additional Columns tab of the Form Designer form for the custom form associated with the resource object
		Process Data Information	Any field that is displayed in the Additional Columns tab of the Form Designer form for the custom form associated with the process
	Organization Provisioning	Request Target Information	Organization Customer Type Organization Name Organization Status Parent Organization Any field that is displayed in the User Defined Fields tab of the Organizations form
	User Provisioning	Request Target Information	Email End Date First Name Identity Last Name Manager Full Name Manager Login
Pre-Populate	User Provisioning	Request Target Information	Display Name Email End Date First Name Identity Last Name Manager Full Name Manager Middle Name Organization Name User Type Start Date Identity Status

Table 30–1 (Cont.) Rule Elements to Create Oracle Identity Manager Rules

Type	Sub-Type	Attribute Source	Variable
			User Role Name
			User Login
			Design Console Access
			Any field that is displayed in the User Defined Fields region of the User Profile tab of the Users form

You can use the Email Definition form to create templates for e-mail notifications to be sent to the users. [Table 30–2](#) lists the variables that can be used to create e-mail templates by using the Email Definition form.

Table 30–2 Variables to Create Templates

Type	Target	Location Type	Contact Type	Variable
Provisioning Related	User Profile Information; Assignee Profile Information	NA	NA	First Name
				Identity
				Last Name
				Manager Login
				Middle Name
				Role
				Status
				End Date
				User Group Name
				User Login
				User Manager
				Start Date
				Oracle Identity Manager Type
Manager Full Name				
Organization Name				
Email				
Provisioning Related	User Profile Information; Assignee Profile Information	NA	NA	Any field that is displayed in the User Defined Fields region of the User Profile tab of the Users form
				Object Information
				Object Target Type

Table 30–2 (Cont.) Variables to Create Templates

Type	Target	Location Type	Contact Type	Variable
				Object Type
	Process Information	NA	NA	Object Name
				Process Name
				Process Type
	Object Data Information	NA	NA	Any field that is displayed in the Additional Columns tab of the Form Designer form for the custom form associated with the resource object
	Process Data Information	NA	NA	Any field that is displayed in the Additional Columns tab of the Form Designer form for the custom form associated with the process
General	User Profile Information	NA	NA	First Name
				Identity
				Last Name
				Email Address
				Manager Login
				Middle Name
				Role
				Status
				User End Date
				User Group Name
				User Login
				User Manager
				User Start Date
				Oracle Identity Manager Type
				Any field that is displayed in the User Defined Fields region of the User Profile tab of the Users form

[Table 30–3](#) describes the properties that can be associated with different data types used to create Oracle Identity Manager forms, by using the Form Designer form.

Table 30–3 Properties Associated with Data Types for Creating Oracle Identity Manager Forms

Data Type	Data Property	Description
Text Field	Required	<p>If this text field must be populated for the form to be saved, then enter "true" into the corresponding Property Value field. Otherwise, type "false" into this field.</p> <p>Note: The default value for this data property is false.</p>
	Is Visible	<p>If you want this text field to be displayed when Oracle Identity Manager generates the form, then enter "true" into the corresponding Property Value field. Otherwise, type "false" into this field.</p> <p>Note: The default value for this data property is true.</p>
Lookup Field	Auto Complete	<p>By entering "true" in the corresponding Property Value field, Oracle Identity Manager filters the lookup field. A user can then add characters to the lookup field before double-clicking it. By doing so, only those Lookup values which match these characters are displayed in the Lookup window.</p> <p>As an example, for a State lookup field, a user can enter "new" into the field. Then, once the user double-clicks the lookup field, only those states that begins with the letters "new" (for example, New Hampshire, New Jersey, New Mexico, and New York) are displayed in the Lookup window. If you do not want Oracle Identity Manager to filter the lookup field, then enter "false" into the associated Property Value field.</p> <p>The default property value for the Auto Complete property is false.</p>
	Column Captions	<p>In the corresponding Property Value field, enter the name of the column heading that is displayed in the Lookup window when a user double-clicks the lookup field. If the Lookup window has multiple columns, then enter each column heading into the Property Value field, separating them with commas, for example, Organization Name, Organization Status.</p>
Lookup Field	Column Names	<p>In the corresponding Property Value field, enter the name of the database column that represents the column caption that you want to be displayed in the Lookup window.</p> <p>If the Lookup window has multiple columns, then enter each database column into the Property Value field, separating them with commas.</p>
	Column Widths	<p>In the corresponding Property Value field, enter the width of the column that is displayed in the Lookup window.</p> <p>If the Lookup window has multiple columns, then enter each column width into the Property Value field, separating them with commas, for example, 20,20.</p>
	Lookup Column Name	<p>In the corresponding Property Value field, enter the name of the Lookup column as it is displayed in the database, which must be saved to the database.</p>

Table 30-3 (Cont.) Properties Associated with Data Types for Creating Oracle Identity Manager Forms

Data Type	Data Property	Description
	Lookup Query	<p>In the corresponding Property Value field, enter the name of the SQL query that runs when a user double-clicks the lookup field. As a result, the appropriate Lookup columns are displayed in the Lookup window.</p> <p>To correctly display the data returned from a query, you must add a <code>lookupfield.header</code> property to the <code>xlWebAdmin_locale.properties</code> file. For example, consider the following SQL query: <code>select usr_status from usr</code>. To view the data returned from the query, you must add the following entry to the <code>xlWebAdmin_locale.properties</code> files:</p> <pre>lookupfield.header.users.status=User Status</pre> <p>If the <code>xlWebAdmin_locale.properties</code> file does not contain a <code>lookupfield.header</code> property for your specified query, then the Administrative and User Console displays a lookup window after you click the corresponding lookup icon.</p> <p>The syntax for a <code>lookupfield.header</code> property is as follows:</p> <pre>lookupfield.header.column_code=display value</pre> <p>The <code>column_code</code> portion of the entry must be lowercase and any space must be replaced by the underscore character (<code>_</code>).</p> <p>By default, the following entries for lookup field column headers are already available in the system resource bundle:</p> <pre>lookupfield.header.lookup_definition.lookup_code_information .code_key=Value lookupfield.header.lookup_definition.lookup_code_information .decode=Description lookupfield.header.users.manager_login=User ID lookupfield.header.organizations.organization_name=Name lookupfield.header.it_resources.key=Key lookupfield.header.it_resources.name=Instance Name lookupfield.header.users.user_id=User ID lookupfield.header.users.last_name=Last Name lookupfield.header.users.first_name=First Name lookupfield.header.groups.group_name=Group Name lookupfield.header.objects.name=Resource Name lookupfield.header.access_policies.name=Access Policy Name</pre>
Lookup Field	Lookup Code	<p>In the corresponding Property Value field, enter the lookup definition code. This code contains all information pertaining to the lookup field, including lookup values and the text that are displayed with the lookup field once a lookup value is selected.</p> <p>Important: The Lookup Code data property can be used in lieu of the Column Captions, Column Names, Column Widths, Lookup Column Name, and Lookup Query properties. In addition, the information contained in the Lookup Code property supersedes any values set in these five data properties.</p> <p>Tip: An easy way to enter a lookup code is by starting the Lookup Definition form, querying for the desired code, copying this code to the Clipboard, and pasting it into the Lookup Code field.</p> <p>Note: The classification type of the lookup definition code must be of Lookup Type (the Lookup Type radio button on the Lookup Definition form must be selected).</p>
	Required	<p>If this Lookup field must be populated for the form to be saved, then enter "true" into the corresponding Property Value field. Otherwise, type "false" into this field.</p> <p>Note: The default value for this data property is false.</p>

Table 30–3 (Cont.) Properties Associated with Data Types for Creating Oracle Identity Manager Forms

Data Type	Data Property	Description
	Visible Field	<p>If you want this lookup field to be displayed when Oracle Identity Manager generates the form, then enter "true" into the corresponding Property Value field. Otherwise, type "false" into this field.</p> <p>Note: The default value for this data property is true.</p>
Text Area	Number of Rows	<p>In the corresponding Property Value field, enter the row length of the text area. So, if you want the text area to be five rows in length, then type "5" into the Property Value field.</p>
	Required	<p>If this text area must be populated for the form to be saved, then enter "true" into the corresponding Property Value field. Otherwise, type "false" into this field.</p> <p>Note: The default value for this data property is false.</p>
	Visible Field	<p>If you want this text area to be displayed when Oracle Identity Manager generates the form, then enter "true" into the corresponding Property Value field. Otherwise, type "false" in this field.</p> <p>Note: The default value for this data property is true.</p>
IT Resource Lookup Field	Type	<p>If you select this data property, then a box is displayed in the Property Value field. From this box, select the type of Server for the IT Resource.</p> <p>Important: This property is required.</p>
	Required	<p>If this lookup field must be populated for the form to be saved, then enter "true" into the corresponding Property Value field. Otherwise, type "false" in this field.</p> <p>Note: The default value for this data property is false.</p>
	Visible Field	<p>If you want this lookup field to be displayed when Oracle Identity Manager generates the form, then enter "true" into the corresponding Property Value field. Otherwise, type "false" into this field.</p> <p>Note: The default value for this data property is true.</p>
Date and Time Window	Required	<p>If this text field must be populated for the form to be saved, enter "true" into the corresponding Property Value field. Otherwise, type "false" into this field.</p> <p>Note: To populate this text field, double-click it, and select a date and time from the Date & Time window that is displayed.</p> <p>Note: The default value for this data property is false.</p>
	Visible Field	<p>If you want this text field to be displayed when Oracle Identity Manager generates the form, then enter "true" into the corresponding Property Value field. Otherwise, type "false" in this field.</p> <p>Note: The default value for this data property is true.</p>
Password Field	Required	<p>If this text field must be populated for the form to be saved, enter "true" into the corresponding Property Value field. Otherwise, type "false" in this field.</p> <p>Note: The default value for this data property is false.</p>
	Visible Field	<p>If you want this text field to be displayed when Oracle Identity Manager generates the form, then enter "true" into the corresponding Property Value field. Otherwise, type "false" in this field.</p> <p>Note: The default value for this data property is true.</p>
Lookup Field	Lookup Code	<p>In the corresponding Property Value field, enter the lookup definition code. This code contains all information pertaining to the lookup field, including lookup values and the text that are displayed with the lookup field once a lookup value is selected.</p>

Table 30-3 (Cont.) Properties Associated with Data Types for Creating Oracle Identity Manager Forms

Data Type	Data Property	Description
	Lookup Query	<p>In the corresponding Property Value field, enter the name of the SQL query that runs when a user double-clicks the lookup field. As a result, the appropriate Lookup columns are displayed in the Lookup window.</p> <p>To correctly display the data returned from a query, you must add a <code>lookupfield.header</code> property to the <code>xlWebAdmin_locale.properties</code> file. For example, consider the following SQL query: <code>select usr_status from usr</code>. To view the data returned from the query, you must add the following entry to the <code>xlWebAdmin_locale.properties</code> files:</p> <pre>lookupfield.header.users.status=User Status</pre> <p>If the <code>xlWebAdmin_locale.properties</code> file does not contain a <code>lookupfield.header</code> property for your specified query, then the Administrative and User Console displays a lookup window after you click the corresponding lookup icon.</p> <p>The syntax for a <code>lookupfield.header</code> property is as follows:</p> <pre>lookupfield.header.column_code=display value</pre> <p>The <code>column_code</code> portion of the entry must be lowercase and any space must be replaced by the underscore character (<code>_</code>).</p> <p>By default, the following entries for lookup field column headers are already available in the system resource bundle:</p> <pre>lookupfield.header.lookup_definition.lookup_code_information .code_key=Value lookupfield.header.lookup_definition.lookup_code_information .decode=Description lookupfield.header.users.manager_login=User ID lookupfield.header.organizations.organization_name=Name lookupfield.header.it_resources.key=Key lookupfield.header.it_resources.name=Instance Name lookupfield.header.users.user_id=User ID lookupfield.header.users.last_name=Last Name lookupfield.header.users.first_name=First Name lookupfield.header.groups.group_name=Group Name lookupfield.header.objects.name=Resource Name lookupfield.header.access_policies.name=Access Policy Name</pre>
	Column Names	<p>In the corresponding Property Value field, enter the name of the database column that represents the column caption that you want to be displayed in the Lookup window.</p> <p>If the Lookup window has multiple columns, then enter each database column into the Property Value field, separating them with commas.</p>
Radio Button	Required	<p>If a radio button must be selected for the form to be saved, then enter "true" into the corresponding Property Value field. Otherwise, type "false" in this field.</p> <p>Note: The default value for this data property is false.</p>
	Visible Field	<p>If you want this radio button (or group of radio buttons) to be displayed when Oracle Identity Manager generates the form, then enter "true" into the corresponding Property Value field. Otherwise, type "false" in this field.</p> <p>Note: The default value for this data property is true.</p>
Check Box	Required	<p>If this check box must be selected for the form to be saved, then enter "true" into the corresponding Property Value field. Otherwise, type "false" in this field.</p> <p>Note: The default value for this data property is false.</p>

Table 30–3 (Cont.) Properties Associated with Data Types for Creating Oracle Identity Manager Forms

Data Type	Data Property	Description
	Visible Field	<p>If you want this check box to be displayed when Oracle Identity Manager generates the form, then enter "true" into the corresponding Property Value field. Otherwise, type "false" in this field.</p> <p>Note: The default value for this data property is true.</p>
Combo Box	Lookup Code	<p>In the corresponding Property Value field, enter the Lookup definition code. This code contains all information pertaining to the box, including box items and the text that is displayed with the box once a lookup value is selected.</p> <p>Important: The Lookup Code data property can be used in lieu of the Column Captions, Column Names, Column Widths, Lookup Column Name, and Lookup Query properties. In addition, the information contained in the Lookup Code property supersedes any values set in these five data properties.</p> <p>Tip: An easy way to enter a lookup code is by starting the Lookup Definition form, querying for the desired code, copying this code to the Clipboard, and pasting it into the Lookup Code field.</p> <p>Note: The classification type of the lookup definition code must be of Lookup Type (the Lookup Type option on the Lookup Definition form must be selected).</p>
	Required	<p>If an item from this box field must be selected for the form to be saved, then enter "true" into the corresponding Property Value field. Otherwise, type "false" in this field.</p> <p>Note: The default value for this data property is false.</p>
	Visible Field	<p>If you want this box to be displayed when Oracle Identity Manager generates the form, then enter "true" into the corresponding Property Value field. Otherwise, type "false" in this field.</p> <p>Note: The default value for this data property is true.</p>
Text Field (Display Only)	Visible Field	<p>If you want this text field to be displayed when Oracle Identity Manager generates the form, then enter "true" into the corresponding Property Value field. Otherwise, type "false" in this field.</p> <p>Note: The default value for this data property is true.</p>

30.2 Service Accounts

Service accounts are general administrator accounts that are used for maintenance purpose. They are typically shared by a set of users. Service accounts are requested, provisioned, and managed in the same manner as regular accounts. A service account is distinguished from a regular account by an internal flag.

When a user is provisioned with a service account, Oracle Identity Manager manages a mapping from the user's identity to the service account. This user is considered the owner of the Service Account. When the user is deleted or the resource is revoked, the provisioning process for the service account does not get canceled, which would cause the undo tasks to fire. Instead, a task is inserted into the provisioning process in the same way Oracle Identity Manager handles Disable and Enable actions. This task removes the mapping from the user to the service account, and returns the service account to the pool of available accounts. This management capability is exposed through APIs.

[Table 30–4](#) describes the service account management tasks and their corresponding APIs.

Table 30–4 Service Account Management Tasks and Corresponding APIs

Tasks	Description	API Methods
Service Account Change	You can change an existing regular account to be a service account or change an existing service account to be a regular account. Either way, the Service Account Change task is inserted into the provisioning process, becoming active in the Tasks tab of the Process Definition. Any adapter that is associated with this provisioning process runs. If there is no adapter, then a predefined response code is attached.	<code>tcUserOperationsIntf.changeFromServiceAccount</code> <code>tcUserOperationsIntf.changeToServiceAccount</code>
Service Account Alert	When a user with a linked service account is deleted or disabled, the Service Account Alert task is inserted into the provisioning process of the service account instance. You can use this task to start the appropriate actions in response to the event that occurred for the user.	NA
Service Account Moved	You can transfer ownership of a service account from one user to another. This translates into the provisioning instance showing up in the resource profile of the new owner, and no longer in the resource profile of the old user. The Service Account Moved task is inserted into the provisioning process of the resource instance after the account is moved. Any adapter associated with this provisioning process runs. If there is no adapter, then a predefined response code is attached.	<code>tcUserOperationsIntf.moveServiceAccount</code>

30.2.1 Service Account Customization: Scenario One

The following scenario describes how to allow a user to request a service account on Active Directory. To create a service account, you first create a regular account, and then use the `changeToServiceAccount` API to change the regular account to a service account. The following is the process to achieve this:

1. The user logs in and requests a service account.
2. The system prompts the Active Directory supervisor for approval.
3. The Active Directory supervisor approves the request.
4. The service account is created.
5. Notification is sent to the employee that the request has been approved.
6. Later, when the service account owner is off-boarded, the owner's supervisor should be assigned as the new owner of the service account and a notification is sent to the owner.

To implement this scenario, perform the following steps:

1. On the Active Directory object form, add a check box field so that the user can select whether the requested account is a service account or a regular account.

2. Modify the Active Directory process form to incorporate the check box field and establish data-flow.

3. Grant the user permissions to update the object form.

The service account request process is the same as the user self-request process. The request is created and approved in the usual manner.

4. Add a conditional task to the provisioning process that will get inserted after the creation of the account and that will check the "is service account" flag on the process form and invoke the `changeToServiceAccount()` API by using the current account's `oiu_key`.

When provisioning starts, the provisioning process checks the flag and loads the `changeToServiceAccount()` API.

Note that tasks can send out e-mail notifications when the tasks are completed.

5. When the user is off-boarded, attach an adapter to the "Service Account Alert" task so that the system can identify the current user, look up that user's manager or supervisor, and load the `tcUserOperationsIntf.moveServiceAccount()` API to reassign ownership of the service account appropriately.

30.2.2 Service Account Customization: Scenario Two

This section describes at a high level how to allow a user to request that service account ownership be transferred away from another user and to the requesting user. The following is the process to achieve this:

1. The user logs in to Oracle Identity Manager and requests a transfer of ownership for a particular Active Directory service account away from the current user and to the requesting user.
2. The request is forwarded to the current service account owner for approval.
3. The service account is transferred to the requesting user upon approval of the current owner.

To implement this scenario, perform the following steps:

Note: This use case requires heavy customization.

1. Because the Oracle Identity Manager user interface does not support account ownership transfer requests, create a dummy resource with custom logic that will query the service accounts present in the system for particular resource objects.
2. The approver in this scenario is the service account owner. Therefore, use a task assignment adapter to first retrieve the service account owner, and then assign the task to that owner.

As noted in the previous scenario, tasks can send out e-mail notifications when tasks are completed.

3. After the approval goes through, load the `moveServiceAccount()` API to transfer ownership of the service account to the requester.

30.3 Design Console Actions

Table 30-5 lists the Oracle Identity Manager actions, and the conditions and results of these actions.

Table 30–5 Oracle identity Manager Actions, Conditions, and Results

Action	Condition	Result
A user is deleted.	Oracle Identity Manager cancels all the existing tasks in process instance and inserts undo tasks for these tasks, if they are defined.	If so, then the condition for this task has been met (the user has been revoked), and Oracle Identity Manager inserts the task into the existing process. If the task has an adapter attached to it, then it will run.
A user is disabled.	Oracle Identity Manager checks each process for any tasks that display the Disable selection in the Task Effect combo box.	If so, then the condition for this task has been met (the user has been disabled), and Oracle Identity Manager inserts the task into the existing process. If the task has an adapter attached to it, then it will run.
A user is enabled.	Oracle Identity Manager checks each process for any tasks that display the Enable selection in the Task Effect combo box.	If so, then the condition for this task has been met (the user has been enabled), and Oracle Identity Manager inserts the task into the existing process. If the task has an adapter attached to it, then it will run.
A user's password has been modified on the Users form	Oracle Identity Manager checks each process to see if it has a Change User Password task.	If so, then the condition for this task has been met (the user's password has been modified), and Oracle Identity Manager inserts the task into all existing processes, which have that task defined. If the task has an adapter attached to it, then it will run.
The data fields of an application process form have been modified.	Oracle Identity Manager checks each process to see if it has a task that starts with the <i>field label Updated</i> naming convention (for example, HomeDirectory Updated).	The condition for this task is met (the process task begins with the <i>field label Updated</i> naming convention). Oracle Identity Manager inserts the task into all existing processes, which have that task defined. If the task has an adapter attached to it, then it will run.
A user's profile information has been moved to a different organization.	Oracle Identity Manager checks each process to see if it has a task that begins with the words Move User.	The condition for this task is met (the user's profile information has been moved to a different organization). Oracle Identity Manager inserts the task into the existing process. If the task has an adapter attached to it, then it will run.

Part VIII

APIs and Web Services

This part describes the APIs and Web services that Oracle Identity Manager supports.

It contains the following chapters:

- [Chapter 31, "Using APIs"](#)
- [Chapter 32, "Using SPML Services"](#)

Oracle provides a network-aware, Java-based application programming interface (API) that exposes Services, called Utility in earlier releases, available in Oracle Identity Manager. This API is based on Plain Old Java Objects (POJO) and takes care of all the plumbing required to interact with Oracle Identity Manager. This API can be used for building clients for Oracle Identity Manager and for integrating third-party products with the Oracle Identity Manager platform.

This chapter contains these sections:

- [Accessing Oracle Identity Manager Services](#)
- [Oracle Identity Manager Services](#)
- [Commonly Used Services](#)
- [Developing Clients for Oracle Identity Manager](#)
- [Working With Legacy Oracle Identity Manager APIs](#)
- [Code Sample](#)

31.1 Accessing Oracle Identity Manager Services

The entry point to Oracle Identity Manager Services is through `oracle.iam.platform.OIMClient` class. `Thor.API.tcUtilityFactory` used in earlier releases is also supported. Oracle recommends using the `oracle.iam.platform.OIMClient` for developing clients to integrate with Oracle Identity Manager.

This section describes the following topics:

- [Using OIMClient](#)
- [Using the tcUtilityFactory](#)

31.1.1 Using OIMClient

OIMClient is the entry point for accessing the services available in Oracle Identity Manager. You use the following sequence of steps when using OIMClient:

1. Create an instance of OIMClient with the environment information required to connect to Oracle Identity Manager application, as shown:

```
Hashtable env = new Hashtable();

env.put(OIMClient.JAVA_NAMING_FACTORY_INITIAL,
"weblogic.jndi.WLInitialContextFactory");
env.put(OIMClient.JAVA_NAMING_PROVIDER_URL, http://OIM_HOSTNAME:OIM_PORT);
OIMClient oimClient = new OIMClient(env);
```

Here, replace *OIM_HOSTNAME* with the host name on which Oracle Identity Manager is deployed and *OIM_PORT* with the port number.

2. Login to the Oracle Identity Manager with the appropriate credentials, as shown:

```
oimClient.login(OIM_USERNAME, OIM_PASSWORD);
```

3. Lookup a service, as shown:

```
UserManager usermgr = oimClient.getService(UserManager.class);
```

4. Call method on a service, as shown:

```
HashMap userAttributes = new HashMap();  
.....  
UserManagerResult result = userMgr.create(new User(null, userAttributes));
```

31.1.2 Using the tcUtilityFactory

Earlier releases of Oracle Identity Manager supports tcUtilityFactory for accessing Oracle Identity Manager Services (or Utilities, as they are called in legacy releases). tcUtilityFactory continues to be supported. However, as mentioned earlier, Oracle recommends using OIMClient for building all client applications for Oracle Identity Manager.

You use the following sequence of steps when using tcUtilityFactory:

1. Create an instance of tcUtilityFactory with the environment information, such as username and password, as shown:

```
tcUtilityFactory ioUtilityFactory = new tcUtilityFactory(env, "OIM_USERNAME",  
"OIM_PASSWORD");
```

2. Look up utility or service by providing the fully qualified name of the utility, as shown:

```
tcUserOperationsIntf moUserUtility =  
(tcUserOperationsIntf)ioUtilityFactory.getUtility("Thor.API.Operations.tcUserOp  
erationsIntf");
```

3. Run operations on the utility, as shown:

```
Hashtable mhSearchCriteria = new Hashtable();  
mhSearchCriteria.put("Users.First Name", psFirstName);  
tcResultSet moResultSet = moUserUtility.findUsers(mhSearchCriteria);
```

31.2 Oracle Identity Manager Services

The Oracle Identity Manager API provides access to services available in Oracle Identity Manager. Because the APIs introduced in 11g Release 1 (11.1.1) and the legacy APIs use different conventions, this section discusses them separately in the following topics:

- [Services Introduced in Oracle Identity Manager 11g Release 1 \(11.1.1\)](#)
- [Legacy Services or Utilities](#)

31.2.1 Services Introduced in Oracle Identity Manager 11g Release 1 (11.1.1)

Services introduced in Oracle Identity Manager 11g Release 1 (11.1.1) follow the following conventions:

- **Package Names:** Services are in packages whose names end with "api", for example:

```
oracle.iam.request.api
oracle.iam.identity.usermgmt.api
```

- **Service Interface Names:** Services introduced in 11g typically use the naming convention of "*Service", for example:

```
oracle.iam.request.api.RequestService
oracle.iam.selfservice.self.selfmgmt.api.AuthenticatedSelfService
```

Some Identity Administration APIs use the "*Manager" naming convention for their APIs, for example:

```
oracle.iam.identity.usermgmt.api.UserManager
```

31.2.2 Legacy Services or Utilities

Legacy services, also called utilities, follow the following naming conventions

- **Package Names:** All legacy APIs are in Thor.API.Operations package.
- **Service Interface Names:** Service names are of the form "*Intf", for example, Thor.API.Operations.tcImportOperationsIntf.

See Also: *Oracle Fusion Middleware Java API Reference for Oracle Identity Manager* for a full list of services available in Oracle Identity Manager. You can use the naming conventions above to find the APIs.

31.3 Commonly Used Services

Table 31–1 lists some commonly used services in Oracle Identity Manager.

Table 31–1 Commonly Used Services

Service Name	Description
UserManager	Provides operations for user management, such as create, search, modify, and delete users
RequestService	Provides operation to submit, withdraw, close, and search requests Note: Using the request template service, when you search for a request template that does not exist, a null value is returned.
RoleManager	Provides operations for role management such as create, search, modify, and delete roles. In addition, this service provides operations for management of role members and relationships between roles.
OrganizationManager	Provides operations for organization management such as create, search, modify, delete, enable, and disable organizations.

31.3.1 Mapping Between Legacy and New Services

In Oracle Identity Manager 11g Release 1 (11.1.1), some of the legacy APIs have been rewritten by using new architecture and the corresponding utility services or interface classes have been changed. [Table 31–2](#) provides a high-level correspondence between the legacy and new interfaces.

Table 31–2 Mapping Between Legacy and New Services

Legacy Service	New Service
Thor.API.Operations.tcUserOperationsIntf	oracle.iam.identity.usermgmt.api.UserManager
Thor.API.Operations.tcGroupOperationsIntf	oracle.iam.identity.rolemgmt.api.RoleManager
Thor.API.Operations.tcOrganizationOperationsIntf	oracle.iam.identity.orgmgmt.api.OrganizationManager
Thor.API.Operations.tcRequestOperationsIntf	oracle.iam.request.api.RequestService
Thor.API.Operations.tcSchedulerOperationsIntf	oracle.iam.scheduler.api.SchedulerService
Thor.API.Operations.tcEmailOperationsIntf	oracle.iam.notification.api.NotificationService

31.4 Developing Clients for Oracle Identity Manager

This section includes the following topics:

- [Prerequisites for Developing Clients](#)
- [Setup and Configuration](#)

31.4.1 Prerequisites for Developing Clients

The following prerequisites must be met for developing clients for Oracle Identity Manager:

- Java Development Kit (JDK) 1.6 installed and set in the path
- ANT 1.7 installed and set in the path

31.4.2 Setup and Configuration

Oracle Identity Manager package contains a ZIP file that contains the required libraries and configuration files for developing clients. It also contains a sample client, which you can use as the starting point for developing your application.

To run an application client for Oracle Identity Manager:

1. Copy `OIM_ORACLE_HOME/server/client/oimclient.zip` to the computer on which you want to develop the client, for example the `oimclient/` directory. This directory is referred to as `OIM_CLIENT_HOME` in this document. Extract the ZIP file. Note that the `oimclient.zip` file consists of the `conf`, `lib`, `sample`, `directories`, `oimclient.jar`, and `README`.
2. Copy the application server-specific client library to the `OIM_CLIENT_HOME/lib/` directory. For Oracle WebLogic Server, `wlfullclient.jar` is the client library. It is created in `MIDDLEWARE_HOME/WL_HOME/server/lib/` directory, for example, `/scratch/beahome/wlserver_10.3/server/lib/`. Check if `wlfullclient.jar` is present. If not, then you must generate one by using the `jarbuilder` tool. See Oracle WebLogic Server documentation on how to generate `wlfullclient.jar`.

3. Edit and run the sample client. To do so:
 - a. Open the `OIM_CLIENT_HOME/sample/src/oracle/iam/samples/SampleOIMClient.java` sample client file.
 - b. Edit the following constants to point to the host on which Oracle Identity Manager is displayed:
 - `OIMURL`: The URL of the Oracle Identity Manager host computer
 - `OIMUserName`: Administrator username for Oracle Identity Manager
 - `OIMPassword`: Administrator password for Oracle Identity Manager
 - c. Run the ant command. This compiles and runs the sample client. The sample generates the following output when it runs successfully:

```
[java] LOGGER >> Creating client....
[java] LOGGER >> Logging in
[java] LOGGER >> Log in successful
[java] LOGGER >> User Created
```

31.5 Working With Legacy Oracle Identity Manager APIs

This section describes the following topics:

- [Using a Result Set Object](#)
- [Handling Oracle Identity Manager Exceptions](#)
- [Cleaning Up](#)

31.5.1 Using a Result Set Object

Legacy Oracle Identity Manager APIs extensively use the `tcResultSet` interface. The `Thor.API.tcResultSet` interface is a data structure that stores records retrieved from the database. Methods in the Oracle Identity Manager API that must return a set of data use a result set. This is a two-dimensional data structure in which the columns correspond to the attributes and rows correspond to the entities. For example, a result set that is returned by the method that searches for users, each row would represent data pertaining to one user, and each column in the row would be an attribute for that user.

You can scroll through the result set and retrieve individual entries corresponding to particular attributes by using the various methods provided. To locate a particular row in the result set, use the `goToRow()` method with the row number as a parameter. To retrieve the values for the columns from a row, use appropriate accessor methods, such as `getStringValue()`. To obtain the value from a specific column, pass the column name as a parameter to the accessor method. The column name is the descriptive code defined in the Oracle Identity Manager Meta-Data system. The following table shows some sample metadata values. This mapping is based on lookup codes and can be looked up in the Design Console by using the Lookup Definition Form.

Column Code	Explanation
IT Resources.Name	The name of an IT resource
Process Definition.Name	The name of a provisioning process

Note: Keep track of the result set objects that are retrieved, because they will be required when updating an existing record.

The following is an example of how to use a result set. This example obtains a result set by calling the `findAllUsers()` method. This method searches for all users matching certain criteria:

```
tcResultSet moResultSet = moUserUtility.findAllUsers(mhAttribs);
```

To check if the `findAllUsers()` method returned any records, use the `isEmpty()` method, for example:

```
boolean mbEmpty = moResultSet.isEmpty();
```

To retrieve the number of records found, use the `getRowCount()` method. If no records are found, then the method returns 0. The following is an example:

```
int mnNumRec = moResultSet.getRowCount();
```

To select a particular record in the system, use the `goToRow()` method:

```
moResultSet.goToRow(5);
```

To retrieve the values of attributes from the current row, use the appropriate accessor method, for example:

```
String msUserLastName = moResultSet.getStringValue("Users.Last Name");
```

31.5.2 Handling Oracle Identity Manager Exceptions

The API methods throw Oracle-defined Java exceptions. Instead of using the `getMessage()` method on the exception object received, you can access the `isMessage` internal variable to retrieve the exception message.

31.5.3 Cleaning Up

The `tcUtilityFactory` class manages all resources used by a utility or factory instance and provides a means to release these resources after they are used.

If you instantiate and use `tcUtilityFactory` to obtain utility class instances, to release the resources that are associated with the utility class, call the `close(utility Object)` method on the factory class. If the session has ended, then call the `close()` method on the factory instance to release all the utility classes, the session objects, and the database objects.

If you obtain a utility class directly by using static calls, after the utility object is no longer needed, call the `close(object)` method on the utility object.

31.6 Code Sample

[Example 31-1](#) illustrates how to retrieve Oracle Identity Manager information. This example creates an instance of the factory class. The instance is then called several times to retrieve individual utility classes and use them to retrieve Oracle Identity Manager information.

Example 31-1 Retrieving Oracle Identity Manager Information

```
/*
```


This class is intended to showcase some of OIM API's. These API's are specific to OIM 11g release. As an example, Legacy API's usage for Organization is also shown.

```

*/

package oracle.iam.samples;

// Role related API's
import oracle.iam.identity.rolemgmt.api.RoleManager;
import oracle.iam.identity.rolemgmt.vo.Role;
import oracle.iam.identity.exception.RoleSearchException;
import oracle.iam.identity.rolemgmt.api.RoleManagerConstants.RoleAttributeName;
import oracle.iam.identity.rolemgmt.api.RoleManagerConstants.RoleCategoryAttributeName;

// User related API's
import oracle.iam.identity.usermgmt.api.UserManager;
import oracle.iam.identity.usermgmt.vo.User;
import oracle.iam.identity.exception.UserSearchException;
import oracle.iam.identity.usermgmt.api.UserManagerConstants.AttributeName;

// Organization Legacy API's
import Thor.API.Operations.tcOrganizationOperationsIntf;
import Thor.API.tcResultSet;
import Thor.API.Exceptions.tcAPIException;
import Thor.API.Exceptions.tcColumnNotFoundException;
import Thor.API.Exceptions.tcOrganizationNotFoundException;

import oracle.iam.platform.OIMClient;
import oracle.iam.platform.authz.exception.AccessDeniedException;
import oracle.iam.platform.entitymgr.vo.SearchCriteria;

import java.util.*;

import javax.naming.NamingException;
import javax.security.auth.login.LoginException;

public class Sample {

    private static OIMClient oimClient;

    /*
     * Initialize the context and login with client supplied environment
     */
    public void init() throws LoginException {
        System.out.println("Creating client...");
        String ctxFactory = "weblogic.jndi.WLInitialContextFactory";
        String serverURL = "t3://OIM_HOSTNAME:OIM_PORT";
        String username = "xelsysadm";
        String password = "xelsysadm";
        Hashtable env = new Hashtable();
        env.put(OIMClient.JAVA_NAMING_FACTORY_INITIAL, ctxFactory);
        env.put(OIMClient.JAVA_NAMING_PROVIDER_URL, serverURL);

        oimClient = new OIMClient(env);
        System.out.println("Logging in");
        oimClient.login(username, password);
        System.out.println("Log in successful");
    }
}

```

```
}

/**
 * Retrieves User login based on the first name using OIM 11g
 * UserManager service API.
 */
public List getUserLogin(String psFirstName) {
    Vector mvUsers = new Vector();
    UserManager userService = oimClient.getService(UserManager.class);
    Set<String> retAttrs = new HashSet<String>();

    // Attributes that should be returned as part of the search.
    // Retrieve "User Login" attribute of the User.
    // Note: Additional attributes can be specified in a
    // similar fashion.
    retAttrs.add(AttributeName.USER_LOGIN.getId());

    // Construct a search criteria. This search criteria states
    // "Find User(s) whose 'First Name' equals 'psFirstName'".
    SearchCriteria criteria;
    criteria = new SearchCriteria(AttributeName.FIRSTNAME.getId(), psFirstName,
SearchCriteria.Operator.EQUAL);
    try {
        // Use 'search' method of UserManager API to retrieve
        // records that match the search criteria. The return
        // object is of type User.
        List<User> users = userService.search(criteria, retAttrs, null);

        for (int i = 0; i < users.size(); i++) {
            //Print User First Name and Login ID
            System.out.println("First Name : " + psFirstName + " -- Login ID : " +
users.get(i).getLogin());
            mvUsers.add(users.get(i).getLogin());
        }
    } catch (AccessDeniedException ade) {
        // handle exception
    } catch (UserSearchException use) {
        // handle exception
    }
    return mvUsers;
}

/**
 * Retrieves the administrators of an Organization based on the
 * Organization name. This is Legacy service API usage.
 */
public List getAdministratorsOfOrganization(String psOrganizationName) {
    Vector mvOrganizations = new Vector();
    tcOrganizationOperationsIntf moOrganizationUtility =
oimClient.getService(tcOrganizationOperationsIntf.class);
    Hashtable mhSearchCriteria = new Hashtable();
    mhSearchCriteria.put("Organizations.Organization Name", psOrganizationName);
    try {
        tcResultSet moResultSet = moOrganizationUtility.findOrganizations(mhSearchCriteria);
        tcResultSet moAdmins;
        for (int i = 0; i < moResultSet.getRowCount(); i++) {
            moResultSet.goToRow(i);
            moAdmins =
moOrganizationUtility.getAdministrators(moResultSet.getLongValue("Organizations.Key"));
            mvOrganizations.add(moAdmins.getStringValue("Groups.Group Name"));
        }
    }
}
```

```

        System.out.println("Organization Admin Name : " +
moAdmins.getStringValue("Groups.Group Name"));
    }
    } catch (tcAPIException tce) {
        // handle exception
    } catch (tcColumnNotFoundException cnfe) {
        // handle exception
    } catch (tcOrganizationNotFoundException onfe) {
        // handle exception
    }
    }
    return mvOrganizations;
}

/**
 * Retrieves Role Display Name based on Role name and Role Category
 * using OIM 11g RoleManager service API. This example shows how
 * to construct compound search criteria.
 */
public List getRoleDisplayName(String roleName, String roleCategory ) {
    Vector mvRoles = new Vector();
    RoleManager roleService = oimClient.getService(RoleManager.class);
    Set<String> retAttrs = new HashSet<String>();

    // Attributes that should be returned as part of the search.
    // Retrieve the "Role Display Name" attribute of a Role.
    // Note: Additional attributes can be specified in a
    // similar fashion.
    retAttrs.add(RoleAttributeName.DISPLAY_NAME.getId());

    // Construct the first search criteria. This search criteria
    // states "Find Role(s) whose 'Name' equals 'roleName'".
    SearchCriteria criterial;
    criterial = new SearchCriteria(RoleAttributeName.NAME.getId(), roleName,
SearchCriteria.Operator.EQUAL);

    // Construct the second search criteria. This search criteria
    // states "Find Role(s) whose 'category' equals 'roleCategory'".
    SearchCriteria criteria2;
    criteria2 = new SearchCriteria(RoleCategoryAttributeName.NAME.getId(), roleCategory,
SearchCriteria.Operator.EQUAL);

    // Construct the compound search criteria using 'criterial1' and
    // 'criteria2' as arguments. This showcases how to construct
    // compound search criterias.
    SearchCriteria criteria = new SearchCriteria(criterial, criteria2,
SearchCriteria.Operator.AND);
    try {
        // Use 'search' method of RoleManager API to retrieve
        // records that match the search criteria. The return
        // object is of type Role.
        List<Role> roles = roleService.search(criteria, retAttrs, null);

        for (int i = 0; i < roles.size(); i++) {
            //Print Role Display Name
            System.out.println("Role Display Name : " +
                roles.get(i).getDisplayName());
            mvRoles.add(roles.get(i).getDisplayName());
        }
    } catch (AccessDeniedException ade) {
        // handle exception
    }
}

```

```
        } catch (RoleSearchException use) {
            // handle exception
        }
        return mvRoles;
    }

    // Main method invocation
    // Following assumptions are made
    //1. A User "Joe Doe" already exists in OIM
    //2. An Organization "Example Organization" already exists in OIM
    //3. A Role "Foobar" already exists in OIM
    public static void main(String args[]) {
        List moList = null;

        try {
            Sample oimSample = new Sample();

            // initialize resources
            oimSample.init();
            // retrieve User logins with first name 'Joe'
            moList=oimSample.getUserLogin("Joe");
            // retrieve User logins with first names starting with 'J'
            moList=oimSample.getUserLogin("J*");
            // retrieve the administrators of an Organization with name
            // 'Example Organization'
            moList=oimSample.getAdministratorsOfOrganization(
                "Example Organization");
            // retrieve Role display name with role name 'FooBar'
            // and role category as 'Default'
            moList=oimSample.getRoleDisplayName("foobar", "Default");
            // release resources
            oimClient.logout();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

The following is the sample output:

```
[java] Creating client....
[java] Logging in
[java] Log in successful
[java] First Name : Joe -- Login ID : JDOE
[java] First Name : J* -- Login ID : JHOND
[java] First Name : J* -- Login ID : JDOE
[java] Organization Admin Name : SYSTEM ADMINISTRATORS
[java] Role Display Name : foobar
```

Using SPML Services

Oracle Identity Manager provides client applications with the Identity Management service, which makes use of the Service Provisioning Markup Language (SPML).

This chapter describes the SPML XSD Web service interfaces supported by Oracle Identity Manager. It contains the following topics:

- [Introduction](#)
- [Create Identity \(SPML Core Service: addRequest\)](#)
- [Modify Users, Roles, Change Attributes and Role Memberships \(SPML Core Service: modifyRequest\)](#)
- [Delete an Identity or Role \(SPML Core Service: deleteRequest\)](#)
- [Check Request Status \(SPML Core Service: statusRequest\)](#)
- [List Available Targets \(SPML Core Service: listTargets\)](#)
- [Disable a User \(SPML Suspend Service: suspendRequest\)](#)
- [Enable a User \(SPML Suspend Service: resumeRequest\)](#)
- [Check if User is Active \(SPML Suspend Service: activeRequest\)](#)
- [Validate a Username \(SPML Username Service: validateUsername\)](#)
- [Obtain a Username \(SPML Username: suggestUsername\)](#)
- [Reset Password \(SPML Core Service: resetPasswordRequest\)](#)
- [Lookup Username Policy \(SPML Username Service: lookupUsernamePolicy\)](#)
- [Cancel/Withdraw Request \(SPML Async Service: cancelRequest\)](#)
- [Batch Request \(SPML Batch Request Service: batchRequest\)](#)
- [Securing SPML Web Services](#)
- [Operations Not Supported](#)

See Also: "SPML Example - Add User" on page C-2 for information about customizing the SPML service with custom attributes

32.1 Introduction

This section introduces the use of SPML services using XSD profile in Oracle Identity Manager.

32.1.1 About SPML Interactions

Oracle Identity Manager provides the identity management service to enable client applications to manage identities (users and roles). The service makes use of the Service Provisioning Markup Language (SPML), which is an XML framework based on specifications from the OASIS committee that provides for exchanging user, resource and service provisioning information.

This document lists and describes the SPML interactions that Oracle Identity Manager supports.

Profile Support

SPML has two profiles: the XSD profile and the DSML profile. This release of Oracle Identity Manager makes use of the XSD profile.

Types of Interactions

The SPML specification allows interactions to be synchronous or asynchronous.

Oracle Identity Manager supports only asynchronous interactions for add, modify, delete, suspend, resume, and reset password requests. For asynchronous interactions, Oracle Identity Manager responds immediately with a pending status, and it is up to the requestor to get the current state by issuing a statusRequest. For add and reset password requests, delayed (conditional) notification is supported. Delayed notification data can be sent as part of payload.

For username services, all services are synchronous.

Search APIs

For search APIs in the Identity Management realm, refer to Oracle Identity Management APIs in the *Oracle Fusion Middleware Java API Reference for Oracle Identity Manager*.

32.1.2 Integration Interface

The integration interface is defined in terms of the Service Provisioning Markup Language (SPML). In Oracle Identity Manager, implementation of SPML supports managing identities and roles, and username reservation capabilities.

Both the asynchronous and synchronous execution modes are supported, although not all services support both modes. If an invalid mode is specified in a request, the service returns an `unsupportedExecutionMode` SPML error code.

To use the SPML services, the application must create a Web service client. The WSDL for this client is available at the following URL:

```
http://OIM_HOST:OIM_PORT/spml-xsd/SPMLService?WSDL
```

As an alternative, you can also navigate to the WSDL and XML schema definitions using a hosted SPML Web service end-point URL.

The XSD (`oracle_common_pso.xsd`) is available at:

```
$OIM_HOME/features/spml-xsd.jar
```

32.2 Create Identity (SPML Core Service: addRequest)

To create an identity with user or role attributes, you implement the `addRequest` operation which supports asynchronous execution mode. Successful request

submission returns a request submission tracking identifier and the request status is listed as pending.

When creating a user, you can also assign role memberships to that user by using the `addRequest` operation. To do this, you must use the SPML reference capability with `typeOfReference` set to `memberOf` and include the role GUID as PSO reference ID.

Note: If the username or password attributes are not provided, those attributes can be auto-generated in Oracle Identity Manager if the appropriate plug-ins are installed.

Table 32–1 lists the features of identity creation with `addRequest` operation.

Table 32–1 Identity Creation with `addRequest`

Item/Feature	Description
SPML Execution Mode	Asynchronous only
Input	<code>addRequest</code> element as defined by [SPMLv2]. Optional, <code>notificationData</code> for controlling user email notifications.
Output	<code>addResponse</code> element as defined by [SPMLv2].
Processing	The add operation allows adding identity. Optionally, existing roles may be assigned to the identity. The runtime errors are reported by using the <code>customError</code> SPML custom error code. Only validation errors are returned in the Response. No request ID is returned. Optionally, notification data can be sent as input as: <ul style="list-style-type: none"> ▪ <code>SentNotification</code>: Boolean flag that determines whether or not to send notification. ▪ <code>SendNotificationTo</code>: Comma separated email address.
Examples	See the Appendix for these examples: <ul style="list-style-type: none"> ▪ "SPML Example - Add User" on page C-2 ▪ "SPML Example – Add User with Role Assignment" on page C-10

32.3 Modify Users, Roles, Change Attributes and Role Memberships (SPML Core Service: modifyRequest)

You implement the SPML `modifyRequest` service for these tasks:

- to assign or revoke role memberships from an existing user (identity)
- to modify an existing role
- to modify user attributes

Table 32–2 lists the features of role membership management with `modifyRequest` operation.

Table 32–2 Role Membership Management with `modifyRequest`

Item/Feature	Description
SPML Execution Mode	Asynchronous

Table 32–2 (Cont.) Role Membership Management with modifyRequest

Item/Feature	Description
Input	<p><code>modifyRequest</code> element as defined by [SPMLv2].</p> <p>Use <code>modificationMode="delete"</code> for deleting role membership and <code>modificationMode="add"</code> for adding role membership.</p> <p>Role memberships declared using Reference capability, with <code>typeOfReference="inheritsFrom"</code> and Role GUID as PSO ID.</p>
Output	<code>modifyResponse</code> element as defined by [SPMLv2].
Processing	<p>The <code>modifyRequest</code> operation allows modifying an existing identity or existing role.</p> <p>This operation checks for SPML execution mode for both identity and role. Invalid execution mode returns an <code>unsupportedExecutionMode</code> SPML error code.</p> <p>If the modify request does not contain identity PSO object, or contains invalid GUIDs the operation returns <code>malformedRequest</code> or <code>invalidIdentifier</code> SPML malformed request error respectively.</p> <p>Other runtime errors are reported using <code>customError</code> SPML custom error code.</p>
Examples	<p>See the Appendix for these examples:</p> <ul style="list-style-type: none"> ▪ Section C.11, "SPML Example - Assign Role Membership" ▪ Section C.13, "SPML Example – Revoke Role Membership"

32.4 Delete an Identity or Role (SPML Core Service: deleteRequest)

You implement the SPML `deleteRequest` service to delete an existing role or user, as described in [Table 32–3](#).

Table 32–3 Role Membership Deletion with deleteRequest

Item/Feature	Description
SPML Execution Mode	Asynchronous
Input	<code>deleteRequest</code> element as defined by [SPMLv2].
Output	<code>deleteResponse</code> element as defined by [SPMLv2].
Processing	<p>The <code>deleteRequest</code> operation allows deletion of an existing identity or existing role.</p> <p>This operation checks for SPML execution mode for both identity and role. Invalid execution mode returns an <code>unsupportedExecutionMode</code> SPML error code.</p> <p>If the delete request does not contain identity PSO object, or contains invalid GUIDs the operation returns <code>malformedRequest</code> or <code>invalidIdentifier</code> SPML malformed request error respectively.</p> <p>Other runtime errors are reported using <code>customError</code> SPML custom error code.</p>
Examples	See the example " SPML Example - Delete Role " on page C-18.

32.5 Check Request Status (SPML Core Service: statusRequest)

The status operation enables a requestor to determine whether an asynchronous operation has:

- failed
- pending
- completed successfully

For any async operation, after the request is submitted, any errors after validation errors cannot be returned in the response. The errors, if any, are returned in the status response. If the statusRequest returns request status as failed, then the statusResponse might have some error message as well.

Table 32–4 lists the features of the statusRequest operation.

Table 32–4 Check Request Status

Item/Feature	Description
SPML Execution Mode	Synchronous
Input	statusRequest element as defined by [SPMLv2].
Output	statusResponse element as defined by [SPMLv2].
Processing	The status operation accepts attribute asyncRequestID which contains the asynchronous operation identifier. If the operation identifier is invalid the noSuchIdentifier error code will be returned. Result of the status operation is provided in the status attribute of statusResponse element.
Example	See the example Section C.20, "SPML Example - Status Request"

32.6 List Available Targets (SPML Core Service: listTargets)

The SPML listTargets service enables a requestor to obtain the set of targets that a provider makes available for provisioning. The service also returns:

- the object types that each target supports
- the set of capabilities that the provider supports for each object in each target

The only target currently supported is Oracle Identity Manager; the object types that we support are all Oracle Identity Manager object types.

Table 32–5 lists the features of obtaining targets with listTargets.

Table 32–5 Obtaining Targets with listTargets

Item/Feature	Description
SPML Execution Mode	Synchronous
Input	listTargetsRequest element as defined by [SPMLv2].
Output	listTargetsResponse element as defined by [SPMLv2].

Table 32–5 (Cont.) Obtaining Targets with listTargets

Item/Feature	Description
Processing	<p>Only the XML Schema profile is supported. Any another profile request results in a failure with the <code>unsupportedProfile</code> error code.</p> <p>A single, static provisioning target named <code>Oracle Identity Manager</code> is supported.</p> <p>The response is generated by inserting the PSO object schemas, the list of supported capabilities for each PSO, and the schema for the operation data capability into a <code>listTargetsResponse</code> element.</p>

32.7 Disable a User (SPML Suspend Service: suspendRequest)

The suspend operation enables the requestor to suspend a user.

[Table 32–6](#) lists the features of the suspendRequest operation.

Table 32–6 Suspending a User with suspendRequest

Item/Feature	Description
SPML Execution Mode	Asynchronous
Input	<code>suspendRequest</code> element as defined by [SPMLv2].
Output	<code>suspendResponse</code> element as defined by [SPMLv2].
Processing	<p>This operation requires a valid user PSO ID and optionally an effective suspension date.</p> <p>If the PSO identifier is invalid, the <code>noSuchIdentifier</code> error code is returned.</p> <p>The suspend operation is applicable for users only. It returns <code>unsupportedOperation</code> error if the PSO object is not an identity.</p>
Examples	See the example " SPML Example - Suspend User " on page C-8.

32.8 Enable a User (SPML Suspend Service: resumeRequest)

The resumeRequest operation enables the requestor to resume/enable a suspended user.

[Table 32–7](#) lists the features of the resumeRequest operation.

Table 32–7 Re-enabling a User with resumeRequest

Item/Feature	Description
SPML Execution Mode	Asynchronous
Input	<code>resumeRequest</code> element as defined by [SPMLv2].
Output	<code>resumeResponse</code> element as defined by [SPMLv2].
Processing	<p>This operation requires a valid user PSO ID and optionally an effective resumption date.</p> <p>If the PSO identifier is invalid, the <code>noSuchIdentifier</code> error code is returned.</p> <p>The resume operation is applicable for users only. It returns <code>unsupportedOperation</code> error if the PSO object is not an identity.</p>

Table 32–7 (Cont.) Re-enabling a User with resumeRequest

Item/Feature	Description
Examples	See the example " SPML Example - Resume User " on page C-7.

32.9 Check if User is Active (SPML Suspend Service: activeRequest)

The activeRequest operation enables a requestor to determine whether a specified user is active or has been suspended.

[Table 32–8](#) lists the features of the activeRequest operation.

Table 32–8 Checking if User Has Been Suspended with activeRequest

Item/Feature	Description
SPML Execution Mode	Synchronous
Input	activeRequest element as defined by [SPMLv2].
Output	activeResponse element as defined by [SPMLv2].
Processing	This operation requires a valid user PSO ID. If the PSO identifier is invalid, the noSuchIdentifier error code is returned. If the request is valid and if the specified user exists, the provider must get the user status. The activeRequest operation is applicable for users only. It returns unsupportedOperation error if the PSO object is not an identity.
Examples	See the example " SPML Example - Check If User is Active " on page C-9.

32.10 Validate a Username (SPML Username Service: validateUsername)

The validateUsername operation enables a requestor to determine whether a username already exists or it is reserved.

[Table 32–9](#) lists the features of the resumeRequest operation.

Table 32–9 Checking Username Validity with resumeRequest

Item/Feature	Description
SPML Execution Mode	Synchronous
Input	validateUsernameRequest element as defined by [SPMLv2]. userName is the only input parameter accepted.
Output	validateUsernameResponse element as defined by [SPMLv2].
Processing	This operation takes a username and checks if the username exists. Processing errors are reported with SPML customError code.
Examples	See the example " SPML Example - Validate User Name " on page C-8.

32.11 Obtain a Username (SPML Username: suggestUsername)

The `suggestUsername` operation enables a requestor to obtain a valid username for a given policy.

[Table 32–10](#) lists the features of the `suggestUsername` operation.

Table 32–10 *Obtaining a Username with suggestUsername*

Item/Feature	Description
SPML Execution Mode	Synchronous
Input	<code>suggestUsernameRequest</code> element as defined by [SPMLv2].
Output	<code>suggestUsernameResponse</code> element as defined by [SPMLv2].
Processing	This operation takes user information and uses it to construct a username based on the applicable username policy. Processing errors are reported with SPML <code>customError</code> code.
Examples	See the example " SPML Example - Suggest User Name " on page C-7.

32.12 Reset Password (SPML Core Service: resetPasswordRequest)

The `resetPasswordRequest` operation enables a requestor to reset the password for a user.

[Table 32–11](#) lists the features of the `resetPasswordRequest` operation.

Table 32–11 *Resetting the user password with resetPasswordRequest*

Item/Feature	Description
SPML Execution Mode	Synchronous
Input	<code>resetPasswordRequest</code> element as defined by [SPMLv2]. Optional notification data for controlling end user email notification.
Output	<code>resetPasswordRequest</code> element as defined by [SPMLv2].
Processing	This operation takes user key or user GUID as an input to reset the password with random generated password. Optionally, notification data can be sent as input as: <ul style="list-style-type: none"> ▪ <code>SentNotification</code>: Boolean flag to determine whether or not to send notification. ▪ <code>SendNotificationTo</code>: Comma separated email address. Processing errors are reported with SPML <code>customError</code> code.
Examples	See the Appendix for these examples: <ul style="list-style-type: none"> ▪ Section C.21, "SPML Example - Reset Password" ▪ Section C.22, "SPML Example - Reset Password with Notification"

32.13 Lookup Username Policy (SPML Username Service: lookupUsernamePolicy)

The `lookupUsernamePolicy` operation enables a requestor to obtain details about the configured username policy in Oracle Identity Manager. You can also provide locale in the request to obtain details in the provided locale.

Table 32–12 lists the features of the lookupUsernamePolicy operation.

Table 32–12 *Lookup Username policy details with lookupUsernamePolicy*

Item/Feature	Description
SPML Execution Mode	Synchronous
Input	lookupUsernamePolicyRequest element as defined by [SPMLv2].
Output	lookupUsernamePolicyResponse element as defined by [SPMLv2].
Processing	This operation returns the information about configured user name policy in Oracle Identity Manager.
Examples	See the example Section C.23, "SPML Example - Lookup User Name Policy" .

32.14 Cancel/Withdraw Request (SPML Async Service: cancelRequest)

The cancel request operation enables the requestor to withdraw the specified request ID. If the request is withdrawn successfully, then all the pending approvals are also withdrawn. Only the requester of the submitted request can withdraw it.

Table 32–13 lists the features of the cancelRequest operation.

Table 32–13 *Cancel a Request with cancelRequest*

Item/Feature	Description
SPML Execution Mode	Synchronous
Input	cancelRequest element as defined by [SPMLv2].
Output	cancelResponse element as defined by [SPMLv2].
Processing	This operation cancels/withdraws the specified request. The runtime errors are reported by using the customError SPML custom error code.
Examples	See the example Section C.24, "SPML Example - Cancel Request" .

32.15 Batch Request (SPML Batch Request Service: batchRequest)

The batch operation combines any number of individual requests into a single request as defined by SPML v2. Examples of individual requests that can be combined into a single request are creating a user Robert Klein, updating a user Terrence Hill, deleting a user John Doe, and reset password for a user Jane Doe in a single request.

Batch request does not support transactional semantics, which means that the failure of a nested request does not undo a nested request that has already been completed. Each individual response occupies the same position within the <batchResponse> that the corresponding individual request occupies within the <batchRequest>.

This operation supports parallel processing only ("processing='parallel'") and runs the nested requests within the <batchRequest> in any order. When error condition occurs, it continues processing the subsequent subrequests, specified by "onError='resume'". If a request fails to be processed, then the next request is processed. If one or more of the nested requests in that batch fails, then operation returns a <batchResponse> with "status='failure'", even if some of the requests in that batch succeed.

lists the features of the batchRequest operation.

Table 32–14 Executing Batch Request with batchRequest

Item/Feature	Description
SPML Execution Mode	Synchronous
Input	batchRequest element as defined by [SPMLv2].
Output	batchResponse element as defined by [SPMLv2].
Processing	This operation supports only four types of sub requests: addRequest for identity, modifyRequest for identity, deleteRequest for identity, resetPasswordRequest.
Examples	See the example Section C.25, "SPML Example - Batch Request" .

32.16 Securing SPML Web Services

This section explains how to secure SPML Web services. It contains these topics:

- [About Web Services Security](#)
- [A Request Example](#)
- [Applying Policies](#)

32.16.1 About Web Services Security

SPML XSD Web service uses Oracle Web Services Security Manager to provide security. SPML Web services is protected by using the following policies:

Note: The SPML XSD profile Web services can be loaded only by users that are a member of the SPML_App_Role. This is done for added security.

See *Oracle Fusion Middleware Security and Administrator's Guide for Web Services* for information about configuring the MBeans for the Web service.

- SAML or username token service policy with message protection:
oracle/wss11_username_token_with_message_protection_client_policy
- In the Fusion Applications environment, with the username token and message protection security:
oracle/wss11_username_token_with_message_protection_client_policy

The default policy can be changed using Oracle Enterprise Manager Fusion Middleware Control.

32.16.2 A Request Example

A sample Request looks like this:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:ns1="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" >
  <soap:Header>
    <ns1:Security>
      <ns1:UsernameToken>
```

```
        <ns1:Username>weblogic</ns1:Username>
        <ns1:Password>weblogic1</ns1:*****>
    </ns1:UsernameToken>
</ns1:Security>
</soap:Header>
<soap:Body xmlns:ns1="urn:oasis:names:tc:SPML:2:0">
    <ns1:listTargetsRequest />
</soap:Body>
</soap:Envelope>
```

32.16.3 Applying Policies

At deployment time, the administrator can use the Oracle Enterprise Manager Fusion Middleware Control Console to apply correct security policy to protect the service. Refer to the following documentation for details about using Fusion Middleware Control:

"Accessing the Security and Administration Tools" in the *Oracle Fusion Middleware Security and Administrator's Guide for Web Services*.

32.17 Operations Not Supported

Oracle Identity Manager 11g Release 1 (11.1.1) does not support the following SPML operations as part of the XSD profile:

- Search user
- Search role
- Any operation, such as create, modify, delete, or search, on organizations

Part IX

Utilities

This part describes how to use the utilities provided by Oracle Identity Manager.

It contains the following chapters:

- [Chapter 33, "MDS Utilities and User Modifiable Metadata Files"](#)
- [Chapter 34, "Using the Bulk Load Utility"](#)
- [Chapter 35, "Upload JAR and Resource Bundle Utilities"](#)

MDS Utilities and User Modifiable Metadata Files

Oracle WebLogic Server provides three utilities to enable you to modify Oracle Identity Manager metadata. They are:

- `weblogicExportMetadata.sh` or `WeblogicExportMetadata.bat`: Export specific metadata files from the MDS database
- `weblogicImportMetadata.sh` or `WeblogicImportMetadata.bat`: Import specific metadata files into the MDS database
- `weblogicDeleteMetadata.sh` or `WeblogicDeleteMetadata.bat`: Delete specific metadata files from the MDS database

This section explains how to use the utilities. Topics include:

- [Setting up the Environment for MDS Utilities](#)
- [Structure of Properties File](#)
- [User Modifiable Metadata Files](#)
- [Example of MDS Utility Usage](#)

33.1 Setting up the Environment for MDS Utilities

There are two steps needed to set up the environment for the MDS utilities:

- set an environment variable
- set up the properties file to specify the parameters needed by the utilities

Set Environment Variable

Set the `OIM_ORACLE_HOME` environment variable to the Oracle Identity Management Oracle home directory inside the Middleware home directory. For example, for Microsoft Windows, set the `OIM_ORACLE_HOME` environment variable to `C:\Oracle\Middleware\Oracle_IDM1\` directory.

Set Up the properties File

Set the necessary properties in the `weblogic.properties` file, which is located in the same folder as the utilities.

Table 33–1 Parameters in the Properties File

Property Name	Description	Notes
wls_servername	Name of the Oracle WebLogic Server on which Oracle Identity Manager is deployed	
application_name	The application name	Value is: <ul style="list-style-type: none"> oim if importing/exporting an out-of-the-box event handler. OIMMetadata for customizable metadata. See "User Modifiable Metadata Files" on page 33-4. If importing or exporting custom data, set application_name to OIMMetadata.
metadata_from_loc	Directory location from which an XML file should be imported. This property is used by weblogicImportMetadata.sh script.	Microsoft Windows paths include // as file or directory separator.
metadata_to_loc	Directory location to which an XML file should be exported. This property is used by the weblogicExportMetadata.sh script.	Microsoft Windows paths include // as file or directory separator.
metadata_files	Full path and name of an XML file. This property is used by weblogicExportMetadata.sh and weblogicDeleteMetadata.sh scripts.	For example, you may specify /file/User.xml to export a user entity definition. You can indicate multiple xml files as comma-separated values.

Using the Import Utility

When you run the `weblogicImportMetadata.sh` utility, all files specified in `metadata_from_loc` will be imported.

For example, you want to import `User.xml` which exists in `/scratch/johnny/temp/oim/file/User.xml`. You must define `metadata_from_loc` as `/scratch/johnny/temp/oim`.

Note: Make sure no other files exist in the directory specified by `metadata_from_loc` or its subdirectories. The import utility tries to recursively import all the files under the directory.

All the Import, Export, and Delete utilities are located in the `OIM_ORACLE_HOME/server/bin/` directory.

Using the Export Utility

When you run the `weblogicExportMetadata.sh` utility, the files specified in `metadata_files` will be exported to the folder specified by `metadata_to_loc`.

Note: If the file name contains spaces, then you must specify the file name as is, without quotes or any escape characters. For example, to export a file named ProvisionResourceeBusiness Suite User.xml, where the path is /db/ProvisionResourceeBusiness Suite User.xml, you must specify the metadata files property as follows:

```
metadata_files=/db/ProvisionResourceeBusiness Suite User.xml
```

Using the Delete Utility

When you run the `weblogicDeleteMetadata.sh` utility, the files specified in `metadata_files` will be deleted from MDS.

Note: If the file name contains spaces, then you must specify the file name as is, without quotes or any escape characters. For example, to delete a file named ProvisionResourceeBusiness Suite User.xml, where the path is /db/ProvisionResourceeBusiness Suite User.xml, you must specify the metadata files property as follows:

```
metadata_files=/db/ProvisionResourceeBusiness Suite User.xml
```

33.2 Structure of Properties File

The properties file looks like this:

```
# Weblogic Server Name on which OIM application is running

wls_servername=@servername

# If you are importing or exporting any out of box event handlers, value is oim.
# For rest of the out of box metadata, value is OIMMetadata.
# If you are importing or exporting any custom data, always use application name
as OIMMetadata.

application_name=@appname

# Directory location from which XML file should be imported.
# Lets say I want to import User.xml and it is in the location
/scratch/johnny/temp/oim/file/User.xml,
# I should give from location value as /scratch/johnny/temp/oim. Make sure no
other files exist
# in this folder or in its sub folders. Import utility tries to recursively import
all the files under the
# from location folder. This property is only used by weblogicImportMetadata.sh

metadata_from_loc=@metadata_from_loc

# Directory location to which XML file should be exported to

metadata_to_loc=@metadata_to_loc

# For example /file/User.xml to export user entity definition. You can specify
multiple xml files as comma separated values.
# This property is only used by weblogicExportMetadata.sh and
weblogicDeleteMetadata.sh scripts

metadata_files=@metadata_files
```

33.3 User Modifiable Metadata Files

The following metadata is used for configuring LDAP Container Rules to determine in which container user and roles should be created in LDAP.

Note: Oracle Identity Manager looks into MDS with file paths starting with /metadata or /db only. Therefore, make sure that starting path or directory name for any XML document is either one of /metadata or /db.

/db/LDAPContainerRules.xml

The following metadata is used for configuring reconciliation profile and reconciliation horizontal table entity definition for LDAP user, role, role hierarchy, and role membership reconciliation:

/db/LDAPUser
/db/LDAPRole
/db/LDAPRoleHierarchy
/db/LDAPRoleMembership
/db/RA_LDAPROLE.xml
/db/RA_LDAPROLEHIERARCHY.xml
/db/RA_LDAPROLEMEMBERSHIP.xml
/db/RA_LDAPUSER.xml
/db/RA_MLS_LDAPROLE.xml
/db/RA_MLS_LDAPUSER.xml
/db/oim-config.xml

The following metadata is used for configuring LDAP user, role entity definitions and membership, and hierarchy relationship definitions:

/metadata/iam-features-ldap-sync/LDAPRole.xml
/metadata/iam-features-ldap-sync/LDAPRoleMembership.xml
/metadata/iam-features-ldap-sync/LDAPUser.xml
/metadata/iam-features-ldap-sync/LDAPUserMembership.xml

The following metadata contain the request model and dataset definitions for default request types:

/metadata/iam-features-requestactions/model-data/AssignRolesDataset.xml
/metadata/iam-features-requestactions/model-data/CreateRoleDataSet.xml
/metadata/iam-features-requestactions/model-data/CreateUserDataSet.xml
/metadata/iam-features-requestactions/model-data/DeleteRoleDataSet.xml
/metadata/iam-features-requestactions/model-data/DeleteUserDataset.xml
/metadata/iam-features-requestactions/model-data/DeleteUserRequest.xml
/metadata/iam-features-requestactions/model-data/DisableUserDataset.xml
/metadata/iam-features-requestactions/model-data/DisableUserRequest.xml
/metadata/iam-features-requestactions/model-data/EnableUserDataset.xml
/metadata/iam-features-requestactions/model-data/EnableUserRequest.xml
/metadata/iam-features-requestactions/model-data/ModifyRoleDataSet.xml
/metadata/iam-features-requestactions/model-data/ModifyUserDataset.xml
/metadata/iam-features-requestactions/model-data/RemoveRolesDataset.xml
/metadata/iam-features-requestactions/model-data/ResourceCommonDataset.xml
/metadata/iam-features-requestactions/model-data/SelfCreateUserDataset.xml

The following metadata contains the predefined event handler definitions for Oracle Identity Manager operations:

Note: These are read only documents. Contact Oracle support if there is a need to modify and delete any of the event handlers that are defined in these metadata file.

```

/db/ldapMetadata/EventHandlers.xml
/metadata/iam-features-OIMMigration/EventHandlers.xml
/metadata/iam-features-Scheduler/EventHandlers.xml
/metadata/iam-features-accesspolicy/event-definition/EventHandlers.xml
/metadata/iam-features-asyncwsclient/EventHandlers.xml
/metadata/iam-features-autoroles/event-definition/EventHandlers.xml
/metadata/iam-features-callbacks/event_configuration/EventHandlers.xml
/metadata/iam-features-configservice/event-definition/EventHandlers.xml
/metadata/iam-features-identity/event-definition/EventHandlers.xml
/metadata/iam-features-notification/EventHandlers.xml
/metadata/iam-features-passwordmgmt/event-definition/EventHandlers.xml
/metadata/iam-features-reconciliation/event-definition/EventHandlers.xml
/metadata/iam-features-request/event-definition/EventHandlers.xml
/metadata/iam-features-requestactions/event-definition/EventHandlers.xml
/metadata/iam-features-selfservice/event-definition/EventHandlers.xml
/metadata/iam-features-sod/EventHandlers.xml
/metadata/iam-features-system-configuration/EventHandlers.xml
/metadata/iam-features-tasklist/EventHandlers.xml
/metadata/iam-features-templatefeature/EventHandlers.xml
/metadata/iam-features-transUI/EventHandlers.xml
/metadata/iam-features-splws/EventHandlers.xml

```

33.4 Example of MDS Utility Usage

To write additional event handlers for any kernel operations, you need to define this event handler in an XML file and seed it into MDS.

For example, suppose you need to write a pre-process event handler on the user create operation to generate the user ID. A sample event handler definition would be like this:

```

<eventhandlers>
  <preprocess-handler
    class="oracle.iam.user.ComputeUserID"
    entity-type="User"
    operation="CREATE"
    name="Compute User ID"
    order="1001"
    stage="preprocess"
    sync="TRUE">
  </preprocess-handler>
</eventhandlers>

```

You would put this content in an XML file called EventHandlers.xml and place it in a directory, such as /scratch/data with a path such as /metadata/user/custom/.

Note: Oracle Identity Manager looks into MDS with file paths starting with /metadata or /db only. Therefore, make sure that starting path or directory name for any XML document is either one of /metadata or /db.

To import the file into MDS, modify the following values in the `weblogic.properties` file and run the `weblogicImportMetadata.sh/weblogicImportMetadata.bat` file:

```
wls_servername=oim server name, for example oim_server1
application_name=oim
metadata_from_loc=/scratch/data
```

The above metadata/XML file is imported into MDS with the full path `/metadata/user/custom/EventHandlers.xml`.

Lets say you want to update the document and change the order in which this event handler is executed. First, export the document by modifying the following values in the `weblogic.properties` file and running the `weblogicExportMetadata.sh/weblogicExportMetadata.bat` file:

```
wls_servername=oim server name, for example oim_server1
application_name=oim
metadata_to_loc=/scratch/data
metadata_files=/metadata/user/custom/EventHandlers.xml
```

The document will be exported to the `/scratch/data/metadata/user/custom` folder. Under `/scratch/data`, if the folder structure `/metadata/user/custom` does not exist, MDS will create it.

You can now edit the file to change the order and run the import command as describe above.

Finally, suppose you decide that instead of being computed, the user ID should be specified during user creation. In that case, this document/XML needs to be deleted. To delete the document, modify the following values in the `weblogic.properties` file and run the `weblogicDeleteMetadata.sh/weblogicDeleteMetadata.bat` file:

```
wls_servername=oim server name, for example oim_server1
application_name=oim
metadata_files=/metadata/user/custom/EventHandlers.xml
```

Using the Bulk Load Utility

Oracle Identity Manager may be one among many repositories of user data in your organization. When you start using Oracle Identity Manager, you might want to load data from the other repositories into Oracle Identity Manager. The Bulk Load utility offers a solution to this requirement.

The Bulk Load utility is aimed at automating the process of loading a large amount of data into Oracle Identity Manager. It helps reduce the downtime involved in loading data. You can use this utility after you install Oracle Identity Manager or at any time during the production lifetime of Oracle Identity Manager.

This document is divided into the following sections:

- [Features of the Bulk Load Utility](#)
- [Installing the Bulk Load Utility](#)
- [Preparing Your Database for a Bulk Load Operation](#)
- [Running the Utility](#)
- [Loading OIM User Data](#)
- [Loading Account Data](#)
- [Loading Role, Role Hierarchy, Role Membership, and Role Category Data](#)
- [Data Recorded During the Operation](#)
- [Gathering Performance Data from the Bulk Load Operation](#)
- [Cleaning Up After a Bulk Load Operation](#)
- [Generating an Audit Snapshot](#)

34.1 Features of the Bulk Load Utility

The following are features of the bulk load utility:

- The utility is compatible with Oracle Identity Manager release 9.1.0 and later.
- Data can be loaded into Oracle Identity Manager as OIM Users, accounts allocated (provisioned) to OIM Users, roles, role hierarchies, role memberships, or role categories.
- Data can be loaded from a single or multiple CSV files or a database table. Data imported into Oracle Identity Manager is automatically converted into OIM Users, accounts provisioned to OIM Users, roles, role hierarchies, role memberships, or role categories.
- Data can be loaded from a single or multiple trusted sources.

- Data can be loaded into either an empty Oracle Identity Manager repository or an Oracle Identity Manager repository that already contains data about OIM Users and resources. In other words, user data can be loaded at any time, either immediately after Oracle Identity Manager installation or when the system is already in production.
- Exceptions generated during user data loading are handled, and records that fail the loading process can be retried.
- Audit snapshots can be generated after a bulk load operation.
- After bulk loading of OIM User data, password change at first login is enforced because a dummy password is used during the operation.

Note: You cannot use the utility to encrypt user attributes. In other words, if a user field in Oracle Identity Manager is encrypted, then the utility cannot be used to encrypt data that is loaded into that field.

34.2 Installing the Bulk Load Utility

To install the utility:

1. Zip and copy the following directory from the installation package into a directory on the Oracle Identity Manager database host computer:

`MIDDLEWARE_HOME/Oracle_IDM1/server/db/oim/oracle/Utilities/oimbulkload`

Note: You can run the utility from a remote host. It is not mandatory to run the utility from a directory in the Oracle Identity Manager database host.

2. Extract the contents of the ZIP file.

The oimbulkload directory is created when you extract the contents of the ZIP file. The following directories are created inside this directory:

- `sqls`: This directory contains SQL scripts used during bulk load operations.
- `scripts`: This directory contains the `.sh` and `.bat` scripts used during bulk load operations.
- `csv_files`: If you are going to use a single or multiple CSV files as the input source, then the CSV files must be stored in this directory.
- `lib`: The directory contains the `oimBulkLoad.jar` file.
- `sample_data`: This directory contains the following sample CSV files:
 - For OIM User load operations:
 - `master.txt`
 - `OIDusers.csv`
 - `HRusers.csv`
 - For account load operations:
 - `parentAD.csv`
 - `childAD.csv`

- For role-related load operations:
 - Role.csv (Role load)
 - Rolec.csv (Role category)
 - Roleh.csv (Role hierarchy)
 - Rolem.csv (Role membership)
- `Logs_YYYYMMDD_hhmi`: The log directory contains the log files that store the summary of the bulk load operation. This directory is created at run time.

The following sections provide additional information about the utility and bulk load operations:

- [Scripts That Constitute the Utility](#)
- [Temporary Tables Used During a Bulk Load Operation](#)
- [Options Offered by the Utility](#)

34.2.1 Scripts That Constitute the Utility

The following are the main scripts that constitute the utility:

- **`oim_blkld.bat` and `oim_blkld.sh`**

This script contains the code to perform bulk load operations. When it is run, this script calls other scripts and stored procedures.
- **`oim_blkld_setup.sql`**

This script is used to add a datafile in the Oracle Identity Manager tablespace. The "Creating a Datafile in the Oracle Identity Manager Tablespace" section of this document provides more information.

34.2.2 Temporary Tables Used During a Bulk Load Operation

The following temporary tables are used during a bulk load operation:

- **`OIM_BLKLD_TMP_SUFFIX`**

If you are using a CSV file as the input source, then the utility automatically creates the `OIM_BLKLD_TMP_SUFFIX` table and first loads data from the CSV file into this table. The suffix for the table name is determined as follows:

 - The first 6 characters of the file name are taken into account.
 - Special characters in the file name and the file extension (`.csv`) are ignored while determining the first 6 characters.
 - A unique number is appended to the first 6 characters.
 - For example, if the name of the file is `acc_Data.csv`, then the table that is created during the bulk load operation is named `oim_blkld_tmp_accDat1`.

If there are multiple CSV files, then one table is created for each file. Because the first six characters of each CSV file name are appended to the table name, you must ensure that the first six characters of each file's name are unique. This guideline is explained later in this document.

Note: if you are using a database table as the input source, then you can specify any name for the table. You provide the name of this table as one of the input parameters of the utility.

- **OIM_BLKLD_EX_SUFFIX**

The OIM_BLKLD_EX_SUFFIX table is used to hold data records that fail (are not loaded into Oracle Identity Manager) during a bulk load operation. One OIM_BLKLD_EX_SUFFIX table is created for each OIM_BLKLD_TMP_SUFFIX table. The EXCEPTION_MSG column of the table stores the reason for failure of each record in the table.

If you are using CSV files as the input source, then the first six characters of the CSV file name are added as a suffix to the table name. For example, if the name of the CSV file is usrdt120508.csv, then the name of the table is OIM_BLKLD_EX_usrdt1. If there are multiple CSV files, then one temporary table is created for each CSV file.

Note: If there are multiple CSV files, then you must ensure that the first six characters of each CSV file name are unique.

- **OIM_BLKLD_LOG**

During a bulk load operation, the utility inserts progress and error messages in the OIM_BLKLD_LOG table. You can query this table to monitor the progress of a bulk load operation. This procedure is described in detail later in this document.

34.2.3 Options Offered by the Utility

When you run the bulk load utility, it prompts you to select one of the following options:

Note: The utility prompts for more input depending on the option you select.

- **Load User Data**

You select this option if you want the utility to load OIM User data. In other words, data is imported into the USR table of Oracle Identity Manager. You can select the input source, CSV files or database tables, for the data that you want to load.

- **Load Account Data**

You select this option if you want the utility to load account data. In other words, data is imported into the relevant UD_ tables of Oracle Identity Manager. You can select the input source, CSV files or database tables, for the data that you want to load.

- **Load Role Data**

You select this option if you want the utility to load role data. In other words, data is imported into the UGP table of Oracle Identity Manager. You can select the input source, CSV files, or database tables, for the data that you want to load.

- **Load Role Membership**

You select this option if you want the utility to load role membership data. In other words, data is imported into the USG table of Oracle Identity Manager. You can select the input source, CSV files or database tables, for the data that you want to load.

- Load Role Hierarchy

You select this option if you want the utility to load role hierarchy data. In other words, data is imported into the GPG table of Oracle Identity Manager. You can select the input source, CSV files, or database tables, for the data that you want to load.

- Load Role Category

You select this option if you want the utility to load role data. In other words, data is imported into the ROLE_CATEGORY tables of Oracle Identity Manager. You can select the input source, CSV files, or database tables, for the data that you want to load.

- Generate Audit Snapshot

You select this option if you want the utility to generate an audit snapshot of data that you have loaded.

34.3 Preparing Your Database for a Bulk Load Operation

Preparing your database for a bulk load operation involves the following:

- [Creating a Tablespace for Temporary Tables](#)
- [Creating a Datafile in the Oracle Identity Manager Tablespace](#)

34.3.1 Creating a Tablespace for Temporary Tables

As mentioned in "[Temporary Tables Used During a Bulk Load Operation](#)", temporary database tables are used during the bulk load operation. It is recommended that you create a tablespace to accommodate these temporary tables instead of using the default tablespace of the Oracle Identity Manager database.

Follow the instructions in the database documentation to create a tablespace.

34.3.2 Creating a Datafile in the Oracle Identity Manager Tablespace

The default size of the datafile in the Oracle Identity Manager tablespace created during Oracle Identity Manager installation is 500 MB. You may need to add space to this datafile to accommodate the data that you are going to load. The alternative is to create a datafile.

To create a datafile in the Oracle Identity Manager tablespace:

1. Start a SQL*Plus session.
2. Connect to the Oracle Identity Manager database as SYSDBA.
3. Run the oim_blkld_setup.sql script. The script will prompt for the following:
 - Name of the Oracle Identity Manager tablespace
 - Full path and name for the datafile to be added in the Oracle Identity Manager tablespace
 - Oracle Identity Manager database user name

34.4 Running the Utility

Note: If there are name conflicts with existing tables, then the utility overwrites existing temporary tables at the start of each run. If required, rename temporary database tables created during an earlier run of the utility.

To run the utility:

1. Stop Oracle Identity Manager.
2. Run one of the following scripts:

Note: To load CSV file with non-ASCII data, before running the oim_blkld.sh or oim_blkld.bat script, set the NLS_LANG environment parameter to the UTF8 character set, in the following format:

```
NLS_LANG = LANGUAGE_TERRITORY.UTF8
```

For example:

```
NLS_LANG = American_America.UTF8
```

- On UNIX computers:
OIMBulkload/script/oim_blkld.sh
 - On Microsoft Windows computers:
OIMBulkload\script\oim_blkld.bat
3. From the main menu, select one of the options depending on the data you want to load, such as user, account, or role-related data, as described in ["Options Offered by the Utility"](#) on page 34-4.
 4. From the second menu:
 - Select **CSV File** if you are using CSV files as the input source.
 - Select **DB Table** if you are using a database table as the input source.
 5. When prompted, provide values for the input parameters described in ["Determining Values for the Input Parameters of the Utility"](#) on page 34-11.

Note: See ["Determining Values for the Input Parameters of the Utility"](#) on page 34-11 for information about the input parameters required for loading OIM User data. See corresponding sections for information about the input parameters required to load account, role, role hierarchy, role membership, and role category data.

6. Monitor the performance of the operation by following the steps given in ["Monitoring the Progress of the Operation"](#).

34.5 Loading OIM User Data

The following is a summary of the steps involved in loading OIM User data:

1. Prepare your database for bulk load if not done already. See "[Preparing Your Database for a Bulk Load Operation](#)" on page 34-5 for details.
2. Create the OIM User whose password will be used as the default password for all OIM Users created during the bulk load operation.
3. Create the input source for the bulk load operation.

If you want to use a database table as the input source, then create the table and copy user data into the table.

If you want to use CSV files as the input source, then create the CSV files and copy user data into the files. In addition, create a master.txt file containing the names of the files in the sequence in which you want to load data from them.
4. Determine values for the input parameters of the utility.
5. Stop Oracle Identity Manager.
6. Run the oim_blkld.sh or oim_blkld.bat script. See "[Running the Utility](#)" on page 34-6 for information about running the oim_blkld.sh or oim_blkld.bat scripts.
7. Monitor the progress of the bulk load operation.
8. Determine the outcome of the bulk load operation.
9. If required, reload data that was not loaded during the first run.
10. Restart Oracle Identity Manager.
11. Verify the outcome of the bulk load operation.
12. Gather performance data from the operation.
13. Remove temporary tables and files created during the operation.
14. Generate an audit snapshot.

The following sections provide detailed information about the steps involved in loading OIM User data:

- [Setting a Default Password for OIM Users Added by the Utility](#)
- [Creating the Input Source for the Bulk Load Operation](#)
- [Determining Values for the Input Parameters of the Utility](#)
- [Monitoring the Progress of the Operation](#)
- [Handling Exceptions Recorded During the Operation](#)
- [Fixing Exceptions and Reloading Data Records](#)
- [Verifying the Outcome of the Bulk Load Operation](#)

34.5.1 Setting a Default Password for OIM Users Added by the Utility

The utility does not encrypt passwords that it assigns to OIM Users created during the bulk load operation. Instead, it assigns the password of an existing OIM User to all OIM Users that are created during the operation.

Note: Each OIM User is required to change the password at first login.

When you run the utility, it prompts for the login name of the existing OIM User whose password you want to use as the default password for the new OIM Users. Before you run the utility, create this OIM User as follows:

Note: You can create a user in Oracle Identity Manager dedicated for the bulk load operation, and later delete the user if it not required any more. Otherwise, any existing OIM User can be used to perform bulk load operations.

1. Log in to the Oracle Identity Manager Administrative and User Console as a user with Create User privileges.
2. Click **Administration**.
3. On the Welcome page, click **Create User**.
4. Specify values for the following fields:
 - User Login
 - First Name (optional)
 - Last Name
 - Organization: Select **Xellerate Users**.
 - Password
 - Confirm Password
5. Click **Save**.

34.5.2 Creating the Input Source for the Bulk Load Operation

Depending on the input source that you want to use, apply the guidelines given in one of the following sections:

- [Using CSV Files As the Input Source](#)
- [Creating Database Tables As the Input Source](#)

34.5.2.1 Using CSV Files As the Input Source

If you want to use CSV files as the input source for the bulk load operation, then apply the following guidelines while creating the CSV files:

- The CSV files must be placed in the oimbulkload/csv_files directory.
- The first line in the CSV file is called the control line. This line must contain a comma-separated list of column names of the USR table in the Oracle Identity Manager database.

Note: Ensure that the Password column or any other encrypted column is not included in the list of columns. As mentioned earlier in this document, the utility assigns the password of an existing OIM User that you specify to all OIM Users that it loads into Oracle Identity Manager.

- From the second line onward, the file must contain values for the columns in the control line. The order of columns in the first line and the values in the rest of the lines must be the same.

The following are sample contents of a CSV file:

```
USR_LOGIN,USR_FIRST_NAME,USR_LAST_NAME,UD_ADUSER_OBJECTGUID
john_doe, John, Doe, jdoe
jane_doe, Jane, Doe, janedoe
richard_roe, Richard, Roe, rroe
```

- If the value in any column contains a comma, then that value must be enclosed in double quotation marks ("").
- The CSV file must contain values for all columns that are designated as mandatory in the USR table. The following table lists the mandatory columns required to load the USR table:

Mandatory Column	Description
USR_FIRST_NAME	The first name of the user
USR_LAST_NAME	The last name of the user

Note:

- USR_LOGIN is not a mandatory column in Oracle Identity Manager 11g Release 1 (11.1.1).
 - There are some key mandatory columns that you can ignore. For example, the ACT_KEY column in the USR table, which is populated by ORG_NAME.
-
-

- Each row in the CSV file must have a unique value for the USR_LOGIN column in the USR table. If there are multiple files, you must ensure that USR_LOGIN values are unique across the CSV files. This check for uniqueness of USR_LOGIN values must also cover existing OIM Users in Oracle Identity Manager.

Ensuring that USR_LOGIN values are unique can be a time-consuming exercise. As an alternative, you can first perform the bulk load operation, fix USR_LOGIN values that are not unique, and then retry the loading operation for the modified user records. This is possible because the utility checks for uniqueness of USR_LOGIN values at run time and copies records that fail this check into the OIM_BLKLD_EX table. Later in this document, there are instructions on retrying the bulk load operation for records that are not loaded during the first run.

- If you want to include an organization name in each user record, then add ORG_NAME in the control line and enter the organization name for each user from the second line onward. If ORG_NAME is not included, then the users must be assigned to the Xellerate Users organization.

Note: All organization names listed under the ORG_NAME column in the CSV file must exist in Oracle Identity Manager.

- If you want to include a manager name in each user record, then add MANAGER_NAME in the control line and enter the USR_LOGIN value of the manager for each user from the second line onward.

The utility looks up the USR_LOGIN values for managers after all user data, from all CSV files, is loaded into Oracle Identity Manager. If a USR_LOGIN value given in the MANAGER_NAME column does not exist in Oracle Identity Manager, then the lookup for that user record fails and the record is copied into the exception table, OIM_BLKLD_EX. At the end of the bulk load operation, you can perform the procedure described in ["Fixing Exceptions and Reloading Data Records"](#) to reload user records that fail the first run.

- Note that the following default values are inserted into Oracle Identity Manager if the CSV file does not contain values for these columns:

ORG_NAME: Xellerate Users

USR_TYPE: End-User

USR_STATUS: Active

USR_EMP_TYPE: Full-Time

- Create a master TXT file containing the names of the CSV files containing user data to be loaded. You can specify any name for the file, for example, master.txt. Save the master file in the oimbulkload/csv_files directory.

If you want to load multiple CSV files, then enter the name of each data CSV file on a separate line in the master file. Order the list of CSV file names in the sequence in which you want the utility to load data from the files. For example, suppose you have created three data CSV files, London_Users.csv, NewYork_Users.csv, and Tokyo_Users.csv. In the master file, you enter the names of the data CSV files in the following order:

```
Tokyo_Users.csv
London_Users.csv
NewYork_Users.csv
```

When you run the utility, data is loaded in this order. This is because the user data in London and New York may have a dependency on the Tokyo users. This is to ensure the manager-user hierarchy.

- If the CSV file is generated on Microsoft Windows and is to be loaded on Linux environment, then remove the special characters, such as '\n\r', to avoid run-time errors.

34.5.2.2 Creating Database Tables As the Input Source

If you want to use a database table as the input source for loading OIM User data, then apply the following guidelines while creating the database table:

- Create the table in the Oracle Identity Manager database.
- The table must contain the following primary key column:

OIM_BLKLD_USRSEQ NUMBER(19)

The utility uses this column as the primary key. If required, you can use a database sequence to populate this column.

- The rest of the columns must be the same as the ones in the USR table that you want to use. In other words, ignore optional USR_ columns that you do not want to include in the table that you create.
- Note that the following default values are inserted into Oracle Identity Manager if the table does not contain values for these columns:

ORG_NAME: Xellerate Users

USR_TYPE: End-User

USR_STATUS: Active

USR_EMP_TYPE: Full-Time

Table 34-1 shows the structure of a sample database table.

Table 34-1 Structure of a Sample Database Table

Name	Null?	Type
USR_LOGIN	NOT NULL	VARCHAR2(256)
USR_FIRST_NAME		VARCHAR2(150)
USR_LAST_NAME	NOT NULL	VARCHAR2(150)
...
OIM_BLKLD_USRSEQ	NOT NULL	NUMBER(19)

34.5.3 Determining Values for the Input Parameters of the Utility

The following are input parameters of the utility:

- Oracle Home
Value of the ORACLE_HOME environment variable on the host computer for the Oracle Identity Manager database
- Database Connection String
Connection string to connect to the database that must be entered in the following format:
`//HOST_IP_ADDRESS:PORT_NUMBER/SERVICE_NAME`
- OIM DB User
Database login ID of the Oracle Identity Manager database user
- OIM DB Pwd
Password of the Oracle Identity Manager database user
The database user password is to be entered twice when prompted.
- Master file name
Name of the file containing names of the CSV data files to be loaded
This parameter is used only if the input source is a single or multiple CSV files. You place the master file and CSV data files in the oimbulkload/csv_files directory. See ["Using CSV Files As the Input Source"](#) for more information.
- Tmp table name
Name of the temporary table to be used as the input source
This parameter is used only if the input source for the bulk load operation is a database table. See ["Creating Database Tables As the Input Source"](#) for more information.
- Control Line
Comma-separated list of names of columns to be loaded from the database table into Oracle Identity Manager

This parameter is used only if the input source for the bulk load operation is a database table.

- **Tablespace Name**

Name of the tablespace in which temporary tables are to be created during the bulk load operation. If the user does not provide the tablespace name, then it will pick the default tablespace.

See "[Preparing Your Database for a Bulk Load Operation](#)" on page 34-5 for more information.

- **Date format**

Date format used by date columns in the CSV files

This parameter is used only if the input source is a single or multiple CSV files.

The date format must match the following:

- Oracle supported date formats, such as dd-mm-yyyy or MM-DD-YYYY
- The date format specified in the CSV file

- **Batch Size**

Number of user records that must be processed by the utility as a single transaction

The batch size can influence the performance of the bulk load operation. The default value of this parameter is 10000.

- **Debug Flag**

You can specify Y or N as the value of this parameter. If this parameter is set to Y, then the utility records detailed information about events that occur during the bulk load operation. See "[Data Recorded During the Operation](#)" on page 34-28 for more information.

- **User ID for default password**

Login name of the OIM User that you create by performing the procedure described in "[Setting a Default Password for OIM Users Added by the Utility](#)" on page 34-7.

34.5.4 Monitoring the Progress of the Operation

During the bulk load operation, you can query the OIM_BLKLD_LOG table for information about the progress of the operation. For example, you can run the following query to see progress messages generated during the bulk load operation to load OIM User data:

```
SELECT MSG FROM OIM_BLKLD_LOG
WHERE MODULE = 'USER' AND LOG_LEVEL = 'PROGRESS_MSG'
ORDER BY MSG_SEQ_NO;
```

Errors encountered during the bulk load operation can be viewed by querying the OIM_BLKLD_LOG table. The following is an example of the query to retrieve error messages:

```
SELECT MSG FROM OIM_BLKLD_LOG
WHERE MODULE = 'USER' AND LOG_LEVEL = 'ERROR'
ORDER BY MSG_SEQ_NO;
```

34.5.5 Handling Exceptions Recorded During the Operation

At the end of a bulk load operation, the utility records statistics related to the operation in the following file:

```
oimbulkload/logs_YYYYMMDD_hhmm/oim_blkld_user_load_summary.log
```

To determine if there were exceptions during the operation, open this log file and look for the number against the Number of Records Rejected label. If the number of rejected records is greater than zero, then exceptions were thrown during the operation. User records that are rejected by the utility are recorded in the exception table (OIM_BLKLD_EX_SUFFIX). For each rejected record, the EXCEPTION_MSG column in the OIM_BLKLD_EX_SUFFIX table stores information about the reason the record could not be loaded.

[Example 34-1](#) shows sample statistics recorded in the log file at the end of a bulk load operation to store OIM User data.

Example 34-1 Sample Log File Generated After Loading OIM User Data

```
*****
Processing File: u10.csv
=====
U S E R   L O A D   S T A T I S T I C S   F O R   F I L E : u10.csv
=====
Start Time:   08-AUG-08 11.44.12.228000 AM
End Time:     08-AUG-08 11.44.13.368000 AM
Number of Records Processed:  10
Number of Records Loaded:     8
Number of Records Rejected:   2
=====
The name of the TMP table used during the load:
OIM_BLKLD_TMP_U101

The name of the Exception table used during the load:
OIM_BLKLD_EX_U101

*****
Processing File: u10b.csv
=====
U S E R   L O A D   S T A T I S T I C S   F O R   F I L E : u10b.csv
=====
Start Time:   08-AUG-08 11.44.15.368000 AM
End Time:     08-AUG-08 11.44.15.540000 AM
Number of Records Processed:  16
Number of Records Loaded:     15
Number of Records Rejected:   1
=====
The name of the TMP table used during the load:
OIM_BLKLD_TMP_U10B2

The name of the Exception table used during the load:
OIM_BLKLD_EX_U10B2
=====

=====
Time taken in re-building indexes and enabling FK constraints
=====
Start time:      08-AUG-08 11.44.15.556000 AM
```

End Time: 08-AUG-08 11.46.50.586000 AM
=====

In this sample, the number of rejected records is 2. If the log file shows that any records were rejected by the utility, then see ["Fixing Exceptions and Reloading Data Records"](#) for information about retrying the load operation for these records.

Note: At the end of each bulk load operation, it is recommended that you create a backup of the exception tables.

34.5.6 Fixing Exceptions and Reloading Data Records

As mentioned earlier, errors encountered during the bulk load operation can be viewed by querying the OIM_BLKLD_LOG table. The following is an example of the query to retrieve error messages:

```
SELECT MSG FROM OIM_BLKLD_LOG
WHERE MODULE = 'USER' AND LOG_LEVEL = 'ERROR'
ORDER BY MSG_SEQ_NO;
```

An exception table OIM_BLKLD_EX_SUFFIX is created for each data table used as the input source during the bulk load operation. Records that do not meet the criteria for the operation are copied into this exception table. The suffix appended to the name of each exception table is the same as suffix appended to the name of the corresponding data table.

To reload rejected records:

1. Create a backup of the exception table in which rejected records are stored.

Note: Although this is an optional step, it is recommended that you create a backup.

2. Review each record in the exception table, and fix errors in the data based on the message recorded in the EXCEPTION_MSG column.
3. After you fix errors in all the rejected records in an exception table, rename the table to OIM_BLKLD_TMP_SUFFIX and then use it as the input source.
4. Load records from the OIM_BLKLD_TMP_SUFFIX table by running the utility. See ["Running the Utility"](#) for more information.
5. Repeat Steps 1 through 4 until the Number of Records Rejected label in the oim_blkld_user_load_summary.log file shows the value 0.
6. Restart Oracle Identity Manager.

34.5.7 Verifying the Outcome of the Bulk Load Operation

To verify the outcome of the bulk load operation, check if you are able to perform the following steps for one of the OIM User added by the utility:

Note: These steps leave footprints in the system, and therefore, the bulk load verification must be performed by using a test user. If you do not want to leave the footprints in the system, then revert the changes. For example, if you have provisioned a resource to a OIM User, then deprovision the resource after testing the outcome of the bulk load operation.

- Log in as the OIM User. The system should prompt you to change the password.
- Provision a resource for the OIM User.
- Add the OIM User to a role.
- Modify the account profile of the OIM User.
- Revoke the resource provisioned to the OIM User.
- Unassign the OIM User from the role to which the user was added earlier.
- Modify the account profile again to restore the profile to its original state.
- Check if the User Resource Access report (an operational report) and the User Resource Access History report can be generated for the user.
- Create an Attestation and check its status using the Diagnostic Dashboard.

34.6 Loading Account Data

The following is a summary of the steps involved in loading account data:

1. Prepare your database for a bulk load operation, if not already done. See ["Preparing Your Database for a Bulk Load Operation"](#) on page 34-5 for details.
2. Create the input source for the bulk load operation.

If you want to use a database table as the input source, then create the table and copy account data into the table.

If you want to use CSV files as the input source, then create the CSV files and copy account data into the files.
3. Determine values for the input parameters of the utility.
4. Stop Oracle Identity Manager.
5. Run the oim_blkld.sh or oim_blkld.bat script.
6. Monitor the progress of the bulk load operation.
7. Determine the outcome of the bulk load operation.
8. If required, reload data that was not loaded during the first run.
9. Restart Oracle Identity Manager.
10. Verify the outcome of the bulk load operation.
11. Gather performance data from the operation.
12. Remove temporary tables and files created during the operation.
13. Generate an audit snapshot.

Requirements and Features of the Bulk Load Operation for Account Data

The following are requirements and features of the bulk load operation for account data:

- Reconciliation must be set up and you should be able to test reconciliation by importing a few accounts from the target system.
- Only accounts for which there are corresponding OIM Users can be loaded.
- A target system that requires multiple IT resources is not supported.
- Duplicate accounts cannot be detected during a bulk load operation. If there are multiple entries for the same account in the input source, then multiple accounts are created for the corresponding OIM User.
- For a particular target system, if there are multiple provisioning processes/process forms in Oracle Identity Manager, then the utility uses the default provisioning process for the resource object.
- Information about the stage up to which earlier bulk load operations progressed is not stored. In other words, the utility cannot resume a bulk load operation. You must backup the Oracle Identity Manager database before a bulk load operation. If you want to retry a bulk load operation, you must first restore the database and then rerun the procedure.

The following sections provide detailed information about the steps involved in loading account data:

- [Creating the Input Source for the Bulk Load Operation](#)
- [Determining Values for the Input Parameters of the Utility](#)
- [Monitoring the Progress of the Operation](#)
- [Handling Exceptions Recorded During the Operation](#)
- [Fixing Exceptions and Reloading Data Records](#)
- [Verifying the Outcome of the Bulk Load Operation](#)

34.6.1 Creating the Input Source for the Bulk Load Operation

Depending on the input source that you want to use, apply the guidelines given in one of the following sections:

- [Using CSV Files As the Input Source](#)
- [Creating Database Tables As the Input Source](#)

34.6.1.1 Using CSV Files As the Input Source

If you want to use CSV files as the input source for the bulk load operation, then apply the following guidelines while creating the CSV files:

- The CSV files must be placed in the `oimbulkload/csv_files` directory.
- The first line in the CSV file is called the control line. This line must contain a comma-separated list of column names in the account (UD_*) table into which you want to load the account data. To find out the UD_ table, go to the process form in the Design Console. See [Chapter 12, "Developing Provisioning Processes"](#) for information about process forms.

Note: Ensure that the Password column or any other encrypted column is not included in the list of columns.

- From the second line onward, the file must contain values for the columns in the control line. The order of columns in the first line and the values in the rest of the lines must be the same.
- If the value in any column contains a comma, then that value must be enclosed in double quotation marks ("").
- The CSV file must contain values for all columns that are designated as mandatory in the account table. The key mandatory columns in the account table must be ignored.
- If you want to load account data into parent and child tables, then you must create one parent CSV file and one child CSV file for each child table. For example if you are loading data into one parent table and three child tables, then you must create one parent CSV file and three child CSV files.
- If you want to load account data into parent and child tables, then at least one column must be the same in both tables. This column corresponds to the link attribute between the parent and child CSV files. The following example illustrates this:

The following are sample contents of a parent CSV file:

```
UD_ADUSER_UID,UD_ADUSER_ORGNAME,UD_ADUSER_FNAME,UD_ADUSER_LNAME,UD_ADUSER_MNAME
,UD_ADUSER_FULLNAME,UD_ADUSER_OBJECTGUID
ADTEST1,"7~CN=ForeignSecurityPrincipals,DC=vivek01,DC=com",adtest1,adtest1,,ad
test1,102
```

The following are sample contents of a child CSV file:

```
UD_ADUSER_UID,UD_ADUSER_ORGNAME,UD_ADUSRC_GROUPNAME
ADTEST1,"7~CN=ForeignSecurityPrincipals,DC=vivek01,DC=com",group2
```

The UD_ADUSER_UID column is common to both the parent file and the child file.

- If the CSV file is generated on Microsoft Windows and is to be loaded on Linux environment, then remove the special characters, such as '\n\r', to avoid run-time errors.

34.6.1.2 Creating Database Tables As the Input Source

If you want to use a database table as the input source for loading account data, then apply the following guidelines while creating the database table:

- Create the table in the Oracle Identity Manager database.
- The table must contain the following primary key column:

```
OIM_BLKLD_USRSEQ NUMBER(19)
```

The utility uses this column as the primary key. If required, you can use a database sequence to populate this column.

- The rest of the columns must be the same as the ones in the account (UD_) table that you want to use. In other words, ignore optional UD_ columns that you do not want to include in the table that you create.

Table 34–2 shows the structure of a sample parent table.

Table 34–2 Structure of a Sample Database Table

Name	Null?	Type
UD_ADUSER_UID		VARCHAR2(20)
UD_ADUSER_ORGNAME		VARCHAR2(256)
UD_ADUSER_FNAME		VARCHAR2(80)
UD_ADUSER_LNAME		VARCHAR2(80)
UD_ADUSER_MNAME		VARCHAR2(80)
UD_ADUSER_FULLNAME		VARCHAR2(240)
OIM_BLKLD_SEQ	NOT NULL	NUMBER(19)

Table 34–3 shows the structure of a sample child table.

Table 34–3 Structure of a Sample Child Database Table

Name	Null?	Type
UD_ADUSER_UID		VARCHAR2(20)
UD_ADUSER_ORGNAME		VARCHAR2(256)
UD_ADUSRC_GROUPNAME		VARCHAR2(32)
OIM_BLKLD_SEQ	NOT NULL	NUMBER(19)

34.6.2 Determining Values for the Input Parameters of the Utility

The following are input parameters of the utility:

- Oracle Home
Value of the ORACLE_HOME environment variable on the host computer for the Oracle Identity Manager database
- Database Connection String
Connection string to connect to the database that must be entered in the following format:
//HOST_IP_ADDRESS:PORT_NUMBER/SERVICE_NAME
- OIM DB User
Database login ID of the Oracle Identity Manager database user
- OIM DB Pwd
Password of the Oracle Identity Manager database user
- Object name (OBJ_NAME)
Name of the resource object corresponding to the account data to be loaded
- CSV file names
Names of the CSV files to be used as the input source

This parameter is used only if the input source is CSV files. See ["Using CSV Files As the Input Source"](#) on page 34-16 for more information. If you are loading data from parent and child CSV file, then use a comma-delimited list to enter the names of the files. The name of the parent CSV file must be provided first, and it must be followed by the names of the child CSV files.

- **Tmp table name**
 Name of the temporary table to be used as the input source
 This parameter is used only if the input source for the bulk load operation is a database table. See "[Creating Database Tables As the Input Source](#)" on page 34-17 for more information.
- **Control Line**
 Comma-separated list of names of columns to be loaded from the database table into Oracle Identity Manager
 This parameter is used only if the input source for the bulk load operation is a database table.
- **Tablespace Name**
 Name of the tablespace in which temporary tables are to be created during the bulk load operation (if end user won't provide the tablespace name then it will pick the default tablespace)
 See "[Preparing Your Database for a Bulk Load Operation](#)" on page 34-5 for more information.
- **Date format**
 Date format used by date columns in the CSV files
 This parameter is used only if the input source is a single or multiple CSV files.
 The date format must match the following:
 - Oracle supported date formats, such as dd-mm-yyyy or MM-DD-YYYY
 - The date format specified in the CSV file
- **Batch Size**
 Number of user records that must be processed by the utility as a single transaction
 The batch size can influence the performance of the bulk load operation. The default value of this parameter is 10000.
- **Debug Flag**
 You can specify Y or N as the value of this parameter. If this parameter is set to Y, then the utility records detailed information about events that occur during the bulk load operation. See "[Data Recorded During the Operation](#)" on page 34-28 for more information.
- **IT Resource Name**
 Name of the IT resource created for the target system

34.6.3 Monitoring the Progress of the Operation

During the bulk load operation, you can query the OIM_BLKLD_LOG table for information about the progress of the operation. For example, you can run the following query to see progress messages generated during the bulk load operation to load account data:

```
SELECT MSG FROM OIM_BLKLD_LOG
WHERE MODULE = 'ACCOUNT' AND LOG_LEVEL = 'PROGRESS_MSG'
ORDER BY MSG_SEQ_NO;
```

For example, you can run the following query to see progress messages generated during the bulk load operation to load account data:

```
SELECT MSG FROM OIM_BLKLD_LOG
WHERE MODULE = 'ACCOUNT' AND LOG_LEVEL = 'PROGRESS_MSG'
ORDER BY MSG_SEQ_NO;
```

Errors encountered during the bulk load operation can be viewed by querying the OIM_BLKLD_LOG table. The following is an example of the query to retrieve error messages:

```
SELECT MSG FROM OIM_BLKLD_LOG
WHERE MODULE = 'ACCOUNT' AND LOG_LEVEL = 'ERROR'
ORDER BY MSG_SEQ_NO;
```

34.6.4 Handling Exceptions Recorded During the Operation

At the end of a bulk load operation, the utility records statistics related to the operation in the following file:

oimbulkload/logs_YYYYMMDD_hhmm/oim_blkld_account_load_summary.log

To determine if there were exceptions during the operation, open this log file and look for the number against the Number of Records Rejected label. If the number of rejected records is greater than zero, then exceptions were thrown during the operation. User records that are rejected by the utility are recorded in the exception table (OIM_BLKLD_EX_SUFFIX). For each rejected record, the EXCEPTION_MSG column in the OIM_BLKLD_EX_SUFFIX table stores information about the reason the record could not be loaded.

[Example 34-2](#) shows sample statistics recorded in the log file at the end of a bulk load operation to store account data.

Example 34-2 Sample Log File Generated After Loading Account Data

```
=====
A C C O U N T   L O A D   S T A T I S T I C S
=====
Start Time:    22-JUL-08 03.59.30.206000 PM
End Time:     22-JUL-08 04.03.21.126000 PM
Number of Records Processed:  100026
Number of Records Loaded:    100000
Number of Records Rejected:   26
=====
```

The names of the TMP tables used during the load:

```
OIM_BLKLD_TMP_P100001
OIM_BLKLD_TMP_C100002
```

The names of the Exception tables used during the load:

```
OIM_BLKLD_EX_P100001
OIM_BLKLD_EX_C100002
```

In this sample, the number of rejected records is 26. If the log file shows that any records were rejected by the utility, then see ["Fixing Exceptions and Reloading Data Records"](#) for information about retrying the load operation for these records.

Note: At the end of each bulk load operation, it is recommended that you create a backup of the exception tables.

34.6.5 Fixing Exceptions and Reloading Data Records

Note: If you want to load data from CSV files for multiple target systems, then you can apply one of the following approaches:

- **Approach 1:** Run the utility for all the sets of CSV files, and then perform the procedure described in this section.
 - **Approach 2:** Run the utility for one set of CSV files, and perform the procedure described in this section. Then, repeat this procedure for the next set of CSV files.
-

As mentioned earlier, errors encountered during the bulk load operation can be viewed by querying the OIM_BLKLD_LOG table. The following is an example of the query to retrieve error messages:

```
SELECT MSG FROM OIM_BLKLD_LOG
WHERE MODULE = 'ACCOUNT' AND LOG_LEVEL = 'ERROR'
ORDER BY MSG_SEQ_NO;
```

An exception table OIM_BLKLD_EX_SUFFIX is created for each data table used as the input source during the bulk load operation. Records that do not meet the criteria for the operation are copied into this exception table. The suffix appended to the name of each exception table is the same as suffix appended to the name of the corresponding data table.

To reload rejected records:

1. Create a backup of the exception table in which rejected records are stored.

Note: Although this is an optional step, it is recommended that you create a backup.

2. Review each record in the exception table, and fix errors in the data based on the message recorded in the EXCEPTION_MSG column.
3. After you fix errors in all the rejected records in an exception table, rename the table to OIM_BLKLD_TMP_SUFFIX and then use it as the input source.
4. Load records from the OIM_BLKLD_TMP_SUFFIX table by running the utility. See ["Running the Utility"](#) for more information.
5. Repeat Steps 1 through 4 until the Number of Records Rejected label in the oim_blkld_account_load_summary.log file shows the value 0.
6. Restart Oracle Identity Manager.

34.6.6 Verifying the Outcome of the Bulk Load Operation

To verify the outcome of the bulk load operation, check if you are able to perform the following steps for one of the OIM Users for whom an account has been added by the utility:

- Log in as the OIM User, and check if the newly created account is displayed in the resource profile of the user.
- Log in to the target system by using the credentials of the newly created account.

34.7 Loading Role, Role Hierarchy, Role Membership, and Role Category Data

The following is a summary of the steps involved in loading role-related data:

1. Prepare your database for a bulk load operation, if not already done. See ["Preparing Your Database for a Bulk Load Operation"](#) on page 34-5 for details.

2. Create the input source for the bulk load operation.

If you want to use a database table as the input source, then create the table and copy role-related data into the table.

If you want to use CSV files as the input source, then create the CSV files and copy role-related data into the files. In addition, create a master.txt file containing the names of the files in the sequence in which you want to load data from them.

3. Determine values for the input parameters of the utility.
4. Stop Oracle Identity Manager.
5. Run the oim_blkld.sh or oim_blkld.bat script.
6. Monitor the progress of the bulk load operation.
7. Determine the outcome of the bulk load operation.
8. If required, reload data that is not loaded during the first run.
9. Restart Oracle Identity Manager.
10. Verify the outcome of the bulk load operation.
11. Gather performance data from the operation.
12. Remove temporary tables and files created during the operation.
13. Generate an audit snapshot.

The following sections provide detailed information about the steps involved in loading OIM User data:

- [Creating the Input Source for the Bulk Load Operation](#)
- [Determining Values for the Input Parameters of the Utility](#)
- [Monitoring the Progress of the Operation](#)
- [Handling Exceptions Recorded During the Operation](#)
- [Fixing Exceptions and Reloading Data Records](#)
- [Verifying the Outcome of the Bulk Load Operation](#)

34.7.1 Creating the Input Source for the Bulk Load Operation

Depending on the input source that you want to use, apply the guidelines given in one of the following sections:

- [Using CSV Files As the Input Source](#)
- [Creating Database Tables As the Input Source](#)

34.7.1.1 Using CSV Files As the Input Source

If you want to use CSV files as the input source for the bulk load operation, then apply the following guidelines while creating the CSV files:

- The CSV files must be placed in the oimbulkload/csv_files directory.
- The first line in the CSV file is called the control line.
- This line must contain a comma-separated list of column names based on the selected role upload (role, role hierarchy, role membership, and role category) in the Oracle Identity Manager database.
- From the second line onward, the file must contain values for the columns in the control line. The order of columns in the first line and the values in the rest of the lines must be the same. The following is a sample content of a role (UGP) CSV file:

```
USR_ROLE_NAME,UGP_NAMESPACE,USR_LOGIN Administrators, null,
XELSYSADM operators, null, XELSYSADM
```

- If the value in any column contains a comma, then that value must be enclosed in double quotation marks ("").
- The CSV file must contain values for all columns that are designated as mandatory in the respective role tables.
- The CSV file must contain values for all columns that are designated as mandatory depending on the upload role data, role hierarchy data, role membership data, and role category data.

- Role (UGP): UGP_ROLENAME, UGP_NAMESPACE, USR_LOGIN
(UGP_NAMESPACE can be left as null when not required)

- Role Hierarchy (GPG): UGP_NAME, GPG_UGP_NAME

- Role Membership (USG): UGP_NAME, USR_LOGIN

- Role Category (ROLE_CATEGORY): ROLE_CATEGORY_NAME

Each row in the CSV file must have a unique value for the combination of mandatory columns.

- The following default values are inserted into Oracle Identity Manager if the CSV file does not contain values for these columns:
 - For Role (UGP)
 - UGP_ROLE_CATEGORY_KEY: ROLE_CATEGORY_KEY from ROLE_CATEGORY table with ROLE_CATEGORY_NAME as 'Default'.
 - UGP_DISPLAY_NAME: Defaults to UGP_NAME.
 - For Role Hierarchy (GPG)
 - For Role Membership (USG)
 - RUL_KEY: RUL_KEY from RUL table with RUL_NAME as 'Default'
 - USG_PRIORITY: group and rank based on UGP_KEY based on the rows given for upload.
 - Role Category (ROLE CATEGORY)
 - None
- Create a master TXT file containing the names of the CSV files containing role data to be loaded. You can specify any name for the file, for example, master.txt. Save the master file in the oimbulkload/csv_files directory.

If you want to load multiple CSV files, then enter the name of each data CSV file on a separate line in the master file. Order the list of CSV file names in the

sequence in which you want the utility to load data from the files. For example, suppose you have created three data CSV files, Role1.csv, Role2.csv, and Role3.csv. In the master file, enter the names of the data CSV files in the following order:

Role1.csv

Role2.csv

Role3.csv

When you run the utility, data is loaded in this order.

- If the CSV file is generated on Microsoft Windows and is to be loaded on Linux environment, then remove the special characters, such as '\n\r', to avoid run-time errors.

34.7.1.2 Creating Database Tables As the Input Source

If you want to use a database table as the input source for loading OIM User data, then apply the following guidelines while creating the database table:

- Create the table in the Oracle Identity Manager database.
- The table must contain the following primary key column:
OIM_BLKLD_USRSEQ NUMBER(19)
The utility uses this column as the primary key. If required, you can use a database sequence to populate this column.
- The rest of the columns must be the same as the ones in the respective role tables that you want to use.

Table 34–4 shows the structure of a sample database role table.

Table 34–4 Structure of a Sample Database Table

Role	NULL	Type
UGP_ROLENAME	NOT NULL	VARCHAR2(2000)
UGP_NAMESPACE		VARCHAR2(512)
...
OIM_BLKLD_USRSEQ	NOT NULL	NUMBER(19)

34.7.2 Determining Values for the Input Parameters of the Utility

The following are input parameters of the utility:

- Oracle Home
Value of the ORACLE_HOME environment variable on the host computer for the Oracle Identity Manager database
- Database Connection String
Connection string to connect to the database that must be entered in the following format:
`//HOST_IP_ADDRESS:PORT_NUMBER/SERVICE_NAME`
- OIM DB User
Database login ID of the Oracle Identity Manager database user

- **OIM DB Pwd**
Password of the Oracle Identity Manager database user
- **Object name (OBJ_NAME)**
Name of the resource object corresponding to the account data to be loaded
- **CSV file names**
Names of the CSV files to be used as the input source

This parameter is used only if the input source is CSV files. See ["Using CSV Files As the Input Source"](#) on page 34-22 for more information. If you are loading data from parent and child CSV file, then use a comma-delimited list to enter the names of the files. The name of the parent CSV file must be provided first, and it must be followed by the names of the child CSV files.
- **Tmp table name**
Name of the temporary table to be used as the input source

This parameter is used only if the input source for the bulk load operation is a database table. See ["Creating Database Tables As the Input Source"](#) on page 34-24 for more information.
- **Control Line**
Comma-separated list of names of columns to be loaded from the database table into Oracle Identity Manager

This parameter is used only if the input source for the bulk load operation is a database table.
- **Tablespace Name**
Name of the tablespace in which temporary tables are to be created during the bulk load operation (if end user won't provide the tablespace name then it will pick the default tablespace)

See ["Preparing Your Database for a Bulk Load Operation"](#) on page 34-5 for more information.
- **Date format**
Date format used by date columns in the CSV files

This parameter is used only if the input source is a single or multiple CSV files.
The date format must match the following:

 - Oracle supported date formats, such as dd-mm-yyyy or MM-DD-YYYY
 - The date format specified in the CSV file
- **Batch Size**
Number of user records that must be processed by the utility as a single transaction

The batch size can influence the performance of the bulk load operation. The default value of this parameter is 10000.
- **Debug Flag**
You can specify Y or N as the value of this parameter. If this parameter is set to Y, then the utility records detailed information about events that occur during the

bulk load operation. See ["Data Recorded During the Operation"](#) on page 34-28 for more information.

- IT Resource Name
Name of the IT resource created for the target system

34.7.3 Monitoring the Progress of the Operation

During the bulk load operation, you can query the OIM_BLKLD_LOG table for information about the progress of the operation. For example, you can run the following query to see progress messages generated during the bulk load operation to load OIM Role data:

```
SELECT MSG FROM OIM_BLKLD_LOG
WHERE MODULE = 'ROLE' AND LOG_LEVEL = 'PROGRESS_MSG'
ORDER BY MSG_SEQ_NO;
```

Errors encountered during the bulk load operation can be viewed by querying the OIM_BLKLD_LOG table. The following is an example of the query to retrieve error messages:

```
SELECT MSG FROM OIM_BLKLD_LOG
WHERE MODULE = 'ROLE' AND LOG_LEVEL = 'ERROR'
ORDER BY MSG_SEQ_NO;
```

34.7.4 Handling Exceptions Recorded During the Operation

At the end of a bulk load operation, the utility records statistics related to the operation in the following file:

oimbulkload/logs_YYYYMMDD_HHMM/oim_blkld_ENTITY_NAME_load_summary.log

In the log file name, *ENTITY_NAME* stands for the entity being loaded. For example:

- For roles, the log file name is oim_blkld_role_load_summary.log.
- For role memberships, the log file name is oim_blkld_rolemem_load_summary.log.

To determine if there were exceptions during the operation, open this log file and look for the number against the Number of Records Rejected label. If the number of rejected records is greater than zero, then exceptions were thrown during the operation. User records that are rejected by the utility are recorded in the exception table (OIM_BLKLD_EX_SUFFIX). For each rejected record, the EXCEPTION_MSG column in the OIM_BLKLD_EX_SUFFIX table stores information about the reason the record could not be loaded.

[Example 34-3](#) shows sample statistics recorded in the log file at the end of a bulk load operation to store OIM Role data.

Example 34-3 Sample Log File Generated After Loading OIM Role Data

```
*****
*****
Processing File: Role.csv
=====
=====
R O L E   L O A D   S T A T I S T I C S   F O R   F I L E : Role.csv
=====
```

```

=====
Start Time: 17-NOV-09 02.48.18.447767 AM
End Time:   17-NOV-09 02.48.19.228710 AM
Number of Records Processed: 2
Number of Records Loaded:    2
Number of Records Rejected:  0
=====
=====

```

The name of the TMP table used during the load:
OIM_BLKLD_TMP_ROLE1

The name of the Exception table used during the load:
OIM_BLKLD_EX_ROLE1

```

=====
=====
Time taken in re-building indexes and enabling FK constraints
=====
=====

```

Start time: 17-NOV-09 02.48.19.243781 AM

In this sample, the number of rejected loaded is 2. If the log file shows that any records have been rejected by the utility, then see ["Fixing Exceptions and Reloading Data Records"](#) on page 34-27 for information about retrying the load operation for these records.

Note: You cannot use the utility to load data into a remote Oracle Identity Manager database.

34.7.5 Fixing Exceptions and Reloading Data Records

As mentioned earlier, errors encountered during the bulk load operation can be viewed by querying the OIM_BLKLD_LOG table. The following is an example of the query to retrieve error messages:

```

SELECT MSG FROM OIM_BLKLD_LOG
WHERE MODULE = 'ROLE' AND LOG_LEVEL = 'ERROR'
ORDER BY MSG_SEQ_NO;

```

An exception table OIM_BLKLD_EX_SUFFIX is created for each data table used as the input source during the bulk load operation. Records that do not meet the criteria for the operation are copied into this exception table. The suffix appended to the name of each exception table is the same as suffix appended to the name of the corresponding data table.

To reload rejected records:

1. Create a backup of the exception table in which rejected records are stored.

Note: Although this is an optional step, it is recommended that you create a backup.

2. Review each record in the exception table, and fix errors in the data based on the message recorded in the EXCEPTION_MSG column.

3. After you fix errors in all the rejected records in an exception table, rename the table to OIM_BLKLD_TMP_SUFFIX and then use it as the input source.
4. Load records from the OIM_BLKLD_TMP_SUFFIX table by running the utility. See "Running the Utility" on page 34-6 for more information.
5. Repeat Steps 1 through 4 until the Number of Records Rejected label shows the value 0 in the oim_blkld_role_load_summary.log file or the corresponding log file for role membership, role hierarchy, and role category.
6. Restart Oracle Identity Manager.

34.7.6 Verifying the Outcome of the Bulk Load Operation

To verify the outcome of the bulk load operation, check if you are able to perform the following steps for one of the OIM Role added by the utility:

1. Log in to Oracle Identity Manager Administrative and User Console and verify that the newly created role is displayed in the search result for roles.
2. For the newly created role hierarchy and role members, click the **Hierarchy** and **Members** tabs respectively on the role details page.
3. To verify the newly created role category, in the Welcome page of Oracle Identity Administration, click **Advanced Search - Role Categories**. Then, perform an advanced search to find the newly created role.

34.8 Data Recorded During the Operation

During the bulk load operation, the utility inserts progress and error messages in the OIM_BLKLD_LOG table. Data in this table is not deleted at the start of a new bulk load operation. One of the columns in this table holds the time stamp at which messages are recorded in the table.

Table 34-5 describes the structure of the OIM_BLKLD_LOG table.

Table 34-5 Structure of the OIM_BLKLD_LOG Table

Column	NULL	Type	Description
MSG_SEQ_NO	NULL	NUMBER(19)	This column stores the number that denotes the order in which messages are inserted in this table. The column is populated by using the OIM_BLKLD_LOG_SEQ sequence. You can use this column to query for messages in the order in which they are recorded in the table.

Table 34–5 (Cont.) Structure of the OIM_BLKLD_LOG Table

Column	NULL	Type	Description
MODULE	NOT NULL	VARCHAR2(20)	<p>This column stores one of the following values:</p> <p>ROLE: This value indicates that the message has been recorded while loading OIM Role data.</p> <p>ROLE HIERARCHY: This value indicates that the message has been recorded while loading role hierarchy data.</p> <p>ROLE MEMBERSHIP: This value indicates that the message has been recorded while loading OIM role membership data.</p> <p>ROLE CATEGORY: This value indicates that the message has been recorded while loading OIM role category data.</p>
LOG_LEVEL	NOT NULL	VARCHAR2(20)	<p>This column stores one of the following values:</p> <p>ERROR: Designates fine-grained informational events that are useful to debug.</p> <p>DEBUG: Designates error events that might allow the application to continue running. Error is used to log all unhandled exceptions.</p> <p>PROGRESS_MSG: Designates intermediate progress messages.</p>
LOAD_SOURCE	NOT NULL	VARCHAR2(40)	<p>This column indicates the source of data for the bulk load operation during which the row was inserted. The value can be one of the following:</p> <p>CSV File: <i>FILE_NAME</i></p> <p>DB Table</p>
MSG	NOT NULL	VARCHAR2(4000)	<p>This column stores a message corresponding to the value stored in the LOG_LEVEL column.</p>
CREATE_DATE		DATE	<p>This column holds the time stamp at which the record was created. The format for entries in this column is as follows:</p> <p>yyyy/mm/dd hh24:mi:ss</p> <p>For example:</p> <p>2008/06/23 21:49:16:32</p>

34.9 Gathering Performance Data from the Bulk Load Operation

As mentioned earlier in this document, the following log files are created during the bulk load operation:

- For OIM Users:
oimbulkload/logs_YYYYMMDD_HHMM/oim_blkld_user_load_summary.log
- For accounts:
oimbulkload/logs_YYYYMMDD_HHMM/oim_blkld_account_load_summary.log
- For roles, role hierarchies, memberships, and role categories:
oimbulkload/logs_YYYYMMDD_HHMM/oim_blkld_ENTITY_NAME_load_summary.log

In the log file name, *ENTITY_NAME* stands for the entity being loaded. For example:

- For roles, the log file name is oim_blkld_role_load_summary.log.
- For role memberships, the log file name is oim_blkld_rolemem_load_summary.log.

Data recorded in this file can be used to collate performance-related information about the bulk load operation. The following information can be collected after the bulk load operation:

- Start time
- Input source
- Number of records in the system before the load
- Number of records successfully loaded
- Number of records rejected
- Total time taken

You can use this information during future runs of the utility.

See Also: [Table 34–5, "Structure of the OIM_BLKLD_LOG Table"](#) for information about the log levels that stores error events

34.10 Cleaning Up After a Bulk Load Operation

If you do not want to save the results of a bulk load operation, then:

- Remove the OIM_BLKLD_TMP_SUFFIX, OIM_BLKLD_EX_SUFFIX, and OIM_BLKLD_LOG tables.
- Remove any files that you created or used during the operation.
- If you created a tablespace for the operation, then remove the tablespace.
- See "[Gathering Performance Data from the Bulk Load Operation](#)" before you remove log files created in the logs_timestamp directory.

Note: At this point, you can restart Oracle Identity Manager if you have not already done so.

34.11 Generating an Audit Snapshot

If required, you can generate an audit snapshot of Oracle Identity Manager data after a bulk load operation, or at any time during the bulk load operation. You can also generate audit snapshots by selecting option 7 in the Bulk Load utility. The utility uses the audit engine shipped with Oracle Identity Manager. Internally, the `GenerateSnapshot` script is called when you run the audit utility. Similarly, the `GenerateSnapshot` script is called when you select the option to generate an audit snapshot.

Note:

- Oracle Identity Manager must be up and running when you run the audit utility.
 - If you encounter a problem in running the Bulk Load utility by selecting option 7 to generate audit snapshots, then run the `GenerateSnapshot.sh` script from the `OIM_HOME/server/bin/` directory.
-
-

Before you generate an audit snapshot:

1. Open the `OIM_HOME/bin/GenerateSnapshot.sh` (or `GenerateSnapshot.bat`) file in a text editor.
2. In this file, search for the following line:
 - In the `GenerateSnapshot.bat` file:
`SET XEL_HOME=..`
 - In the `GenerateSnapshot.sh` file:
`XEL_HOME=..`
3. Replace this line with the following line:
 - In the `GenerateSnapshot.bat` file:
`SET XEL_HOME=OIM_HOME_DIRECTORY_PATH`
 - In the `GenerateSnapshot.sh` file:
`XEL_HOME=OIM_HOME_DIRECTORY_PATH`
4. Save and close the file.

See "Auditing" in the *Oracle Fusion Middleware User's Guide for Oracle Identity Manager* for information about the procedure to generate audit snapshots.

Upload JAR and Resource Bundle Utilities

When migrating from test to production environment, all the connector artifacts must be migrated to the respective database tables, which can be done using the following upload utilities:

- [Upload JAR Utility](#)
- [Download JAR Utility](#)
- [Delete JAR Utility](#)
- [Upload Resource Bundle Utility](#)
- [Download Resource Bundle Utility](#)
- [Delete Resource Bundle Utility](#)

Note:

- All the Upload JAR and Resource Bundle utilities must be run from the *OIM_HOME/bin/* directory.
- Make sure that *wfullclient.jar* is generated before running these utilities.
- Set *WL_HOME* before running the scripts.
- All the scripts for the JAR files and resource bundles support both interactive mode and command-line mode usage. But it is recommended to use interactive mode because this is secure and the passwords are not echoed on the console.
- For running the scripts in command-line mode, run it with the *-help* argument. For example:

```
sh UploadJars.sh -help
```

To upload a JAR file in the silent mode:

```
UploadJars.sh [-username USERNAME] [-password PASSWORD]
[-serverURL <t3://OIM_HOSTNAME:OIM_PORT>] [-ctxFactory
<weblogic.jndi.WLInitialContextFactory>] [-JavaTasks
LOCATION_OF_JAVA_TASK_JAR]
```

To upload multiple JAR files in the silent mode:

```
UploadJars.sh [-username USERNAME] [-password PASSWORD]
[-serverURL <t3://OIM_HOSTNAME:OIM_PORT>] [-ctxFactory
<weblogic.jndi.WLInitialContextFactory>] [-JavaTasks <Location
of the Java Task Jar>] [-ScheduleTask
LOCATION_OF_SCHEDULED_TASK_JAR] [-ThirdParty
LOCATION_OF_THIRD_PARTY_JAR]
```

- In this document, interactive mode usage of the JAR and Resource Bundle utilities are explained because it is a secure way of running the utilities and is recommended.

To run the JAR or Resource Bundle utilities in interactive mode, run the scripts without specifying any arguments. For example:

```
sh UploadJars.sh
```

35.1 Upload JAR Utility

The *UploadJars.sh* and *UploadJars.bat* scripts are available in the *OIM_HOME/bin/* directory. Running these scripts upload the JAR files in to the database.

A sample invocation of this utility is as shown:

```
[Enter OIM admin username :]ADMINISTRATOR_LOGIN
[Enter the admin password :]ADMINISTRATOR_PASSWORD
[Enter serverURL :[ t3://localhost:14000 ]]t3://xyz.com:14000
[Enter context Factory :[ weblogic.jndi.WLInitialContextFactory
]]weblogic.jndi.WLInitialContextFactory
Enter the jar type
 1.JavaTasks
 2.ScheduleTask
 3.ThirdParty
```

```

1
Enter the path/location of jar file :
/tmp/example.jar
Do u want to load more jars [y/n] :n

```

Note: 14000 is Oracle Identity Manager port.

35.2 Download JAR Utility

The DownloadJars.sh and DownloadJars.bat scripts are available in the *OIM_HOME/bin/* directory. Running these scripts download the JAR files from the database.

A sample invocation of this utility is as shown:

```

[Enter OIM admin username :]ADMINISTRATOR_LOGIN
[Enter the admin password :]ADMINISTRATOR_PASSWORD
[Enter serverURL :[ t3://localhost:14000 ]]t3://localhost:14000
[Enter context Factory :[ weblogic.jndi.WLInitialContextFactory
]]weblogic.jndi.WLInitialContextFactory
Enter the jar type
1.JavaTasks
2.ScheduleTask
3.ThirdParty
1
Enter the full path of the download directory :
/home/joe/tmp
Enter the name of jar file to be downloaded from DB :
example.jar
Do u want to download more jars [y/n] :n

```

Note: 14000 is Oracle Identity Manager port.

35.3 Delete JAR Utility

The DeleteJars.sh and DeleteJars.bat scripts are available at the *OIM_HOME/bin/* directory. Running these scripts delete the JAR files from the database.

A sample invocation of this utility is as shown:

```

[Enter OIM admin username :]ADMINISTRATOR_LOGIN
[Enter the admin password :]ADMINISTRATOR_PASSWORD
[Enter serverURL :[ t3://localhost:14000 ]]t3://localhost:14000
[Enter context Factory :[ weblogic.jndi.WLInitialContextFactory
]]weblogic.jndi.WLInitialContextFactory
Enter the jar type
1.JavaTasks
2.ScheduleTask
3.ThirdParty
1
Enter the name of jar to be deleted from DB :
example.jar
Do u want to delete more jars [y/n] :n

```

35.4 Upload Resource Bundle Utility

The UploadResourceBundles.sh and UploadResourceBundles.bat scripts are available in the `OIM_HOME/server/bin/` directory. Running these scripts upload the connector or custom resources to the database.

A sample invocation of this utility is as shown:

```
Enter OIM admin username :]ADMINISTRATOR_LOGIN
[Enter the admin password :]ADMINISTRATOR_PASSWORD
[Enter serverURL :[ t3://localhost:14000 ]]t3://localhost:14000
[Enter context Factory :[ weblogic.jndi.WLInitialContextFactory
]]weblogic.jndi.WLInitialContextFactory
Enter the resource bundle type
  1.Custom Resource
  2.Connector Resource
  2
Enter the path/location of resource bundle file :
/tmp/example.properties
Do u want to load more resource bundles [y/n] :n
```

35.5 Download Resource Bundle Utility

The DownloadResourceBundles.sh and DownloadResourceBundles.bat scripts are available in the `OIM_HOME/bin/` directory. Running these scripts download the resource bundles from the database.

A sample invocation of this utility is as shown:

```
[Enter OIM admin username :]ADMINISTRATOR_LOGIN
[Enter the admin password :]ADMINISTRATOR_PASSWORD
[Enter serverURL :[ t3://localhost:14000 ]]t3://localhost:14000
[Enter context Factory :[ weblogic.jndi.WLInitialContextFactory
]]weblogic.jndi.WLInitialContextFactory
Enter the resource bundle type
  1.Custom Resource
  2.Connector Resource
  2
Enter the full path of the download directory :
/home/joe/tmp
Enter the name of resource bundle file :
example.properties
Do u want to download more resource bundles [y/n] :n
```

35.6 Delete Resource Bundle Utility

The DeleteResourceBundles.sh and DeleteResourceBundles.bat are available in the `OIM_HOME/bin/` directory. Running these utilities delete the resource bundles from the database.

A sample invocation of this utility is as shown:

```
[Enter OIM admin username :]ADMINISTRATOR_LOGIN
[Enter the admin password :]ADMINISTRATOR_PASSWORD
[Enter serverURL :[ t3://localhost:14000 ]]t3://localhost:14000
[Enter context Factory :[ weblogic.jndi.WLInitialContextFactory
]]weblogic.jndi.WLInitialContextFactory
Enter the resource bundle type
  1.Custom Resource
```

2.Connector Resource

2

Enter the name of resource bundle file to be deleted from DB:

example.properties

Do u want to delete more resource bundles [y/n] :n

Part X

Reporting

This part describes how to configure reports in Oracle Identity Manager and develop entitlement-level reports.

It contains the following chapters:

- [Chapter 36, "Configuring Reports"](#)
- [Chapter 37, "Developing Entitlements"](#)

Configuring Reports

This chapter describes Oracle Identity Manager Reports and contains the following topics:

- [What is Oracle Identity Manager Reports?](#)
- [What is Oracle BI Publisher?](#)
- [Supported Products](#)
- [Licensing](#)
- [Prerequisites for Deploying Oracle Identity Manager Reports](#)
- [Configuring Oracle Identity Manager Reports](#)
- [Generating Oracle Identity Manager Reports](#)

36.1 What is Oracle Identity Manager Reports?

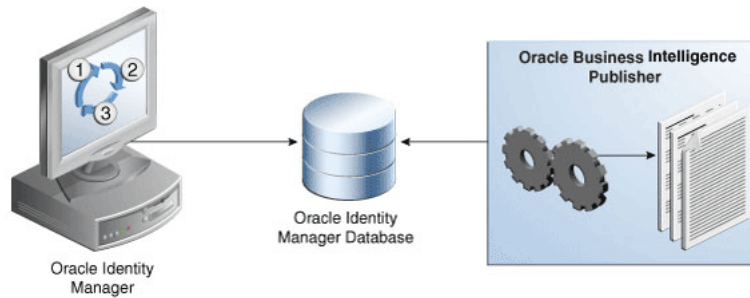
Oracle Identity Manager Reports enables you to use Oracle BI Publisher as the reporting solution for Oracle Identity Management products.

Note: Oracle Identity Manager Reports are classified based on the functional areas. For instance, Access Policy Reports, Attestation, Request and Approval Reports, Password Policy Reports and so on. It is no longer named Operational and Historical.

Oracle Identity Manager Reports provides a restricted-use license for Oracle BI Publisher and easy-to-use reporting packages for multiple Oracle Identity Management products.

As shown in [Figure 36–1](#), Oracle Identity Manager Reports uses Oracle BI Publisher to query and report on information in Oracle Identity Management product databases. With minimal setup, Oracle Identity Manager Reports provides a common method to create, manage, and deliver Oracle Identity Manager Reports.

Figure 36–1 Oracle Identity Manager Reports Architecture



The report templates included in Oracle Identity Manager Reports are standard Oracle BI Publisher templates. However, you can customize each template to change its look and feel. If schema definitions for an Oracle Identity Management product are available, you can use that information to create your own custom reports.

Note: Oracle strongly recommends creating back-up copies of the original default report templates before customizing them.

36.2 What is Oracle BI Publisher?

Oracle BI Publisher is an Oracle's enterprise reporting solution and provides a single reporting environment to author, manage, and deliver all of your reports and business documents. Utilizing a set of familiar desktop tools, such as Microsoft Word, Microsoft Excel, or Adobe Acrobat, you can create and maintain report layouts based on data from diverse sources, including Oracle Identity Management products.

See Also: *Oracle Business Intelligence Publisher Documentation* to learn more about Oracle BI Publisher functionality.

36.3 Supported Products

Oracle Identity Manager Reports 11g Release 1 (11.1.1) supports the products listed in [Table 36–1](#).

Table 36–1 Supported Products

Technology	Product and Version
Oracle Identity Management	<p>You can use one of the following releases of Oracle Identity Manager:</p> <ul style="list-style-type: none"> ■ Oracle Identity Manager 11g Release 1 (11.1.1.5.3) or later ■ Oracle Identity Manager 11g Release 1 (11.1.1.7)

Table 36–1 (Cont.) Supported Products

Technology	Product and Version
Oracle BI Publisher	<p>Depending on the Oracle Identity Manager release that is running in your environment, you can use one of the following releases of Oracle BI Publisher:</p> <ul style="list-style-type: none"> ■ For Oracle Identity Manager 11g Release 1 (11.1.1.5.3) or later: Oracle BI Publisher 11g (11.1.1.5.0) ■ For Oracle Identity Manager 11g Release 1 (11.1.1.7): Oracle BI Publisher 11g (11.1.1.6.4) with patch# 14088000 and 14630670

36.4 Licensing

Oracle Identity Manager can be separately licensed, independent of any Oracle Application Server or WebLogic edition. BI Publisher is included when you separately license Oracle Identity Manager:

- Shipped BI Publisher reports. Layout changes are allowed, AND
- Shipped or newly created BI Publisher reports that are modified to access data from the existing Identity Management schema that has not been customized.

36.5 Prerequisites for Deploying Oracle Identity Manager Reports

This section explains the prerequisites for deploying Oracle Identity Manager Reports 11g Release 1 (11.1.1) and contains the following topics:

- [Creating the Metadata Repository](#)
- [Installing BI Publisher 11g \(11.1.1.6\)](#)

36.5.1 Creating the Metadata Repository

Each Oracle Business Intelligence system (BI domain) requires its own set of database schemas. Two or more systems cannot share the same set of schemas or repositories.

You must create a repository in your database by using the Repository Configuration Utility (RCU) before installing BI Publisher 11g (11.1.1.6). For this, you need the RCU utility, which you can download from the Oracle Web site by using the following URL:

<http://www.oracle.com/technetwork/middleware/bi-enterprise-edition/downloads/bi-downloads-1525270.html>

For installing BI Publisher 11g (11.1.1.6), the following metadata repositories are required:

- Metadata Store (MDS)
- Business Intelligence Platform (BI Platform)

To create the repository in your database by using the RCU utility:

1. To run RCU, you must have the DBA privilege. Therefore, you must log in as SYSDBA, for example, as user SYS.
2. Navigate to the `RCU_HOME/bin/` directory.
3. To start RCU:

- For UNIX, run:
 - `./rcu`
 - For Microsoft Windows, run:
 - `rcu.bat`
4. In the Welcome screen that is displayed, click **Next**. The Repository Creation Utility wizard is displayed.
 5. In step 1 of the wizard, select **Create**, and then click **Next**. Step 2 of 7: Database Connection Details page is displayed.
 6. Specify the connection details, as listed in the following table:

Field	Data to Enter
Database Type	Oracle Database
Host Name	Name of the host on which the database is deployed.
Port	Port number to connect to the host identified in the Host Name field.
Service Name	A string that is the global database name, a name comprised of the database name and domain name, entered during installation or database creation. If you are not sure what the global database name is, then you can obtain it from the combined values of the SERVICE_NAMES parameter in the database initialization file, which is INITSID.ORA. For example, a service name can be SALES.COM, where SALES is the database name and COM is the domain.
Username	Username for a database schema user that has access to Oracle Identity Manager, such as SYS.
Password	Password for the user identified in the Username field.
Role	The role with DBA privilege, such as SYSDBA.

7. Click **Next**. Step 3 of 7: Component Detail page is displayed.
8. Select the Oracle Business Intelligence component. This action automatically selects the MDS schema under the AS Common Schemas group, which is also required by Oracle Business Intelligence.
9. Click **Next**. Step 4 of 7: Schema Passwords page is displayed.
10. Specify the same password for schemas.
11. Click **Next**. Step 5 of 7: Map TableSpaces page is displayed.
12. Click **Next**. A message is displayed after the validation is complete.
13. Click **OK**. Step 6 of 7: Summary page is displayed with the details about the component, schema owner, tablespace type, and tablespace name.
14. Click **Next**. Step 7 of 7: Completion Summary page is displayed.
15. Click **Close**. The metadata repository is created in your database.

Tip: The log files are saved in the `RCU_HOME\log\` directory.

36.5.2 Installing BI Publisher 11g (11.1.1.6)

Note: You can extend only empty Oracle WebLogic domains with the BI domain. Extending BI components by using Oracle Universal Installer in an existing domain is not supported.

All Oracle Business Intelligence products run on Oracle WebLogic Server domains. Therefore, Oracle WebLogic Server must be installed and configured before you install BI Publisher 11g (11.1.1.6).

If you do not have Oracle WebLogic Server installed, then OBIEE 11g installs the WebLogic Server by default, and creates the bi domain named bifoundation_domain under the user_projects/domains directory.

Installing BI Publisher 11g (11.1.1.6) is described in the following sections:

- [Starting the Oracle Business Intelligence Wizard](#)
- [Installing BI Publisher 11g \(11.1.1.6\) When Oracle WebLogic Server is Not Installed](#)
- [Installing BI Publisher 11g \(11.1.1.6\) When Oracle WebLogic Server is Installed](#)

Starting the Oracle Business Intelligence Wizard

To start the Oracle Business Intelligence wizard, go to the bishiphome/Disk1/ directory, and run the following:

For UNIX:

```
./runInstaller
```

For Microsoft Windows:

```
setup.exe
```

Installing BI Publisher 11g (11.1.1.6) When Oracle WebLogic Server is Not Installed

After starting the Oracle Business Intelligence wizard, perform the following steps:

1. In Step 1 of 15: Welcome page of the wizard that is displayed, click **Next**. Step 2 of 15: Type Install page is displayed.
2. Select the **Install Software Updates** option, and click **Next**. Step 3 of 15: Select Installation Type page is displayed.
3. Select the **Enterprise Install** option, and click **Next**. Step 4 of 15: Pre-requisite Check page is displayed.
4. Click **Next**. Step 5 of 15: Create or Scale BI System page is displayed.
5. Select the **Create New BI System** option. Then, enter values in the following fields:

User Name: Enter the WebLogic user name.

Password: Enter a password for the WebLogic user.

Confirm Password: Re-enter the password for the WebLogic user.

Domain Name: Name of the WebLogic domain, which is bifoundation_domain by default.

6. Click **Next**. Step 6 of 15: Specify Install Location page is displayed.

7. Enter the directory paths for the installation, and click **Next**.
8. Click **Next**. Step 7 of 12: Configure Components page is displayed.
9. Click **Next**. Step 8 of 12: Database Details page is displayed.
10. Enter the details of your database with the credentials that you have specified in the Step 5 of "[Creating the Metadata Repository](#)" on page 36-3.
11. Complete the remaining steps of the wizard by clicking **Next**.

Installing BI Publisher 11g (11.1.1.6) When Oracle WebLogic Server is Installed

When BI Publisher 11g (11.1.1.6) is already installed, perform the following steps in the Oracle Business Intelligence wizard:

1. In Step 2 of 15: Type Install page, select the **Simple Install** option, and click **Next**. Step 3 of 15: Prerequisite Check page is displayed.
2. Click **Next**. Step 4 of 12: Middleware Home page is displayed.
3. Enter the Middleware home directory path without trailing space, for example, /u01/app/ Oracle_IDM1/Middelware/.
4. Click **Next**. Step 5 of 12: Administrator Details page is displayed.
5. Enter the administrator user name and password. This account is used for the administration of the WebLogic Server and Enterprise Manager.
6. Click **Next**. Step 6 of 12: Configure Components page is displayed.
7. Click **Next**. Step 7 of 12: Database Details page is displayed.
8. Enter the details of your database with the credentials that you have specified in the Step 5 of "[Creating the Metadata Repository](#)" on page 36-3.
9. Complete the remaining steps of the wizard by clicking **Next**.

36.6 Configuring Oracle Identity Manager Reports

This section describes configuring BI Publisher 11g (11.1.1.6) in the following topics:

- [Deploying Oracle Identity Manager Reports on BI Publisher 11g \(11.1.1.6\)](#)
- [Configuring Data Sources for Running Oracle Identity Manager Reports](#)

36.6.1 Deploying Oracle Identity Manager Reports on BI Publisher 11g (11.1.1.6)

To deploy security in BI Publisher 11g (11.1.1.6):

1. In the Oracle_IDM1/Middleware/user_projects/domains/bifoundation_domain/config/bipublisher/repository/Reports/ directory, create a new directory and name it as Oracle Identity Manager.

Note: After installing BI Publisher 11g (11.1.1.6), the Oracle_IDM1/Middleware/user_projects/domains/bifoundation_domain/config/bipublisher/repository/Reports/ directory is created in the WebLogic domain.

2. Extract the contents of the OIM_11gR1_BIP11gReports.zip file in the newly created Oracle Identity Manager directory.

3. Login to BI Publisher. To do so:
 - a. In a Web browser, enter the URL in the following format:
`http://HOST_NAME:PORT_NUMBER/xmlpserver/`
For example, `http://localhost:7001/xmlpserver/`
 - b. In the Oracle BI Publisher login page, enter the username and password with WebLogic privileges.
4. Select **Administration, Security Centre**. Then, click **Security Configuration**.
5. Go to the Security Model page and select the security model according to the implementation. In OBIEE, the default security model is Oracle Fusion Middleware Security Model. For information about configuring Oracle Fusion Middleware security model, refer to the following URL:
http://docs.oracle.com/cd/E21764_01/bi.1111/e13880/T526682T559093.htm#ofm1
6. According to the selected security model, login to BI Publisher as the Administrator or OIM User. The Home page is displayed.
7. Upload Oracle Identity Manager reports to BI Publisher. To do so:
 - a. Select **Administration, System Maintenance**, and then click **Server Configuration**.
 - b. Scroll down to the Catalog section and verify that the path to the repository folder is correct in the BI Publisher repository field.
 - c. Click **Upload to BI Presentation Catalog** to upload the Oracle Identity Manager reports to BI Publisher.
 - d. If required, restart both the BI Publisher managed server and admin server for the changes to take affect.
 - e. Login to BI Publisher again and click **Catalog**. Expand the Shared Folders tree in the left pane to verify that the reports are present.

36.6.2 Configuring Data Sources for Running Oracle Identity Manager Reports

For Oracle Identity Manager reports, JDBC connections described in the following sections are required:

- [Configuring Oracle Identity Manager JDBC Connection](#)
- [Configuring BPEL-Based JDBC Connection](#)

36.6.2.1 Configuring Oracle Identity Manager JDBC Connection

To configure Oracle Identity Manager JDBC connection:

1. Click the **Administration** link on the top of the Home page. The BI Publisher Administration page is displayed.
2. Under Data Sources, click the **JDBC Connection** link. The Data Sources page is displayed.
3. In the JDBC tab, click **Add Data Source** to create a JDBC connection to your database. The Add Data Source page is displayed.
4. Enter values in the following fields:

- **Data Source Name:** Specify the Oracle Identity Manager JDBC connection name, for example, OIM JDBC.
 - **Driver Type:** Select a driver type to suit your database. For example, you can select Oracle 10g or Oracle 11g to suit your database.
 - **Database Driver Class:** Specify a driver class to suit your database, such as `oracle.jdbc.driver.OracleDriver`.
 - **Connection String:** Specify the database connection details in the format `jdbc:oracle:thin:@HOST_NAME:PORT_NUMBER:SID`. For example, `jdbc:oracle:thin:@localhost:7003:orcl`.
 - **User name:** Specify the Oracle Identity Manager database user name.
 - **Password:** Specify the Oracle Identity Manager database user password.
5. Click **Test** to verify the connection, and then click **Apply** to establish the connection.
 6. If the connection to the database is established, a confirmation message is displayed indicating the success. Click **Apply**.

In the JDBC page, you can see the newly defined Oracle Identity Manager JDBC connection in the list of JDBC data sources.

36.6.2.2 Configuring BPEL-Based JDBC Connection

In BI Publisher, only one data source can be assigned to a report. The first data source is the Oracle Identity Manager data source. The following reports have a secondary data source, which connects to the BPEL database to retrieve BPEL data:

- Task Assignment History
- Request Details
- Request Summary
- Approval Activity

To configure a secondary data source for BPEL-based reports:

1. In the BI Publisher Home page, click **Administration**. The BI Publisher Administration page is displayed.
2. Under Data Sources, click the **JDBC Connection** link. The Data Sources page is displayed.
3. In the JDBC tab, click **Add Data Source** to create a JDBC connection to your database. The Add Data Source page is displayed.
4. Enter values in the following fields:
 - **Data Source Name:** Specify the BPEL JDBC connection name, for example, BPEL JDBC.
 - **Driver Type:** Select a driver type to suit your database. For example, you can select Oracle 10g or Oracle 11g to suit your database.
 - **Database Driver Class:** Specify a driver class to suit your database, such as `oracle.jdbc.driver.OracleDriver`.
 - **Connection String:** Specify the database connection details in the format `jdbc:oracle:thin:@HOST_NAME:PORT_NUMBER:SID`. For example, `jdbc:oracle:thin:@localhost:7003:orcl`.
 - **User name:** Specify the SOA schema or user name.

- **Password:** Specify the SOA database user password.
- 5. Click **Test** to verify the connection, and then click **Apply** to establish the connection.
- 6. If the connection to the database is established, a confirmation message is displayed indicating the success. Click **Apply**.

In the JDBC page, you can see the newly defined BPEL JDBC connection in the list of JDBC data sources.

36.7 Generating Oracle Identity Manager Reports

This section explains how to generate Oracle Identity Manager Reports and contains the following topics:

- [Generating Sample Reports Against the Sample Data Source](#)
- [Generating Reports Against the Oracle Identity Manager JDBC Data Source](#)
- [Generating Reports Against the BPEL-Based JDBC Data Source](#)

Note: BI Publisher cannot be accessed through the Oracle Identity Self Service or Oracle Identity System Administration. You must open BI publisher explicitly to access the Oracle Identity Manager 11g reports.

36.7.1 Generating Sample Reports Against the Sample Data Source

If you want to see an example of what report data will look like without running a report against the production JDBC Data Source, you can generate a sample report against the Sample Data Source. You must create the Sample Data Source before you can generate sample reports. Refer to appropriate section for your Oracle Identity Management product in "[Configuring Oracle Identity Manager Reports](#)" on page 36-6 for information on creating the Sample Data Source.

After you create the Sample Data Source you can generate sample reports against it by performing the following steps:

1. Login to Oracle BI Publisher. See step 3 in [Deploying Oracle Identity Manager Reports on BI Publisher 11g \(11.1.1.6\)](#) for more information about logging in to Oracle BI Publisher.
2. Click **Shared Folders, Oracle Identity Manager Reports**, and then select **Sample Reports**.
3. Click **View** for the sample report you want to generate.
4. Select an output format for the sample report and click **View**.

The sample report is generated.

36.7.2 Generating Reports Against the Oracle Identity Manager JDBC Data Source

To generate reports against the Oracle Identity Manager JDBC data source:

1. Log in to Oracle BI Publisher. See step 3 in [Deploying Oracle Identity Manager Reports on BI Publisher 11g \(11.1.1.6\)](#) for more information about logging in to Oracle BI Publisher.
2. Navigate to Oracle Identity Manager reports. To do so:

- a. In the BI Publisher Home page, under Browse/Manage, click **Catalog Folders**. Alternatively, you can click **Catalog** at the top of the page.

The Catalog page is displayed with a tree structure on the left side of the page and the details on the right.

- b. On the left pane, expand **Shared Folders**, and navigate to Oracle Identity Manager. All the objects in the Oracle Identity Manager folder are displayed.

You are ready to navigate to BI Publisher 11g and use the Oracle Identity Manager BI Publisher reports.

3. Click **View** for the report you want to generate.
4. Select an output format for the report and click **View**.

The report is generated.

See Also: *Oracle Business Intelligence Publisher Documentation* to learn more about Oracle BI Publisher.

36.7.3 Generating Reports Against the BPEL-Based JDBC Data Source

The following four reports have a secondary data source, which connects to the BPEL database to retrieve BPEL data:

- Task Assignment History
- Request Details
- Request Summary
- Approval Activity

These reports have a secondary data source, which is the BPEL-based JDBC Data Source, and is called `BPEL JDBC`.

To generate reports against the BPEL-based JDBC data source:

1. Ensure that a BPEL data source exists in BI Publisher. This BPEL Data Source must point to the BPEL database. See "[Configuring BPEL-Based JDBC Connection](#)" on page 36-8 for more information about creating a BPEL data source.
2. Log in to Oracle BI Publisher. See step 3 in [Deploying Oracle Identity Manager Reports on BI Publisher 11g \(11.1.1.6\)](#) for more information about logging in to Oracle BI Publisher.
3. Navigate to Oracle Identity Manager reports. To do so:
 - a. In the BI Publisher Home page, under Browse/Manage, click **Catalog Folders**. Alternatively, you can click **Catalog** at the top of the page.
The Catalog page is displayed with a tree structure on the left side of the page and the details on the right.
 - b. On the left pane, expand **Shared Folders**, and navigate to Oracle Identity Manager. All the objects in the Oracle Identity Manager folder are displayed.
You are ready to navigate to BI Publisher 11g and use the Oracle Identity Manager BI Publisher reports.
4. Click **View** for the report you want to generate.
5. Select an output format for the report, and click **View**.

The report is generated based on the BPEL-based JDBC data source.

Developing Entitlements

An entitlement granted to an account on a target system enables the account owner (user) to perform a specific task or function. An entitlement can be a role, responsibility, or group membership. For example, if user Richard is granted the Inventory Analyst role on a target system, then Richard can use that entitlement to access and generate inventory-related reports from the target system.

In Oracle Identity Manager, there is one process form for each account (resource) provisioned to an OIM User. Entitlement data is stored in child process forms of the process form. In the example described earlier, the process form for Richard's account on the target system has a child process form that holds Inventory Manager role data.

Attributes that constitute entitlement data stored on a child process form may vary from one target system to another. In addition, different types of entitlements, such as roles and responsibilities, may have different attributes. For example, Target System A contains the following role data attributes:

- Role Name
- Role Description
- Start Date
- End Date

The same target system can have a different set of attributes for responsibility data:

- Responsibility ID
- Date Assigned
- Proxy User
- Escalation User

You can mark or highlight the attribute that uniquely identifies an entitlement on a target system. For the sample role and responsibility data attributes listed earlier, the Role Name and Responsibility ID attributes uniquely identify the role and responsibility entitlements on Target System A. By marking attributes that uniquely identify entitlements, you enable the capture of entitlement data that can be used by other identity management solutions and also displayed in reports.

This chapter discusses the following sections:

- [Available Entitlements and Assigned Entitlements](#)
- [Entitlement Data Capture Process](#)
- [Marking Entitlement Attributes on Child Process Forms](#)
- [Configuring Scheduled Tasks for Working with Entitlement Data](#)

- [Disabling the Capture of Modifications to Assigned Entitlements](#)
- [Entitlement-Related Reports](#)

37.1 Available Entitlements and Assigned Entitlements

A target system can have a set of entitlements defined and ready for assignment to accounts (users) on the target system. When you integrate this target system with Oracle Identity Manager, you can import (synchronize) entitlement data from the target system into the LKV table on Oracle Identity Manager.

Note: If you use a predefined connector to integrate the target system, then you can use scheduled tasks to fetch entitlement data into this table.

Entitlements in the LKV table are available for assignment to accounts. In this guide, these entitlements are called available entitlements.

During a provisioning operation, you select the entitlement that you want to assign from a lookup field on the child process form. In this guide, entitlements assigned to accounts are called assigned entitlements. Data about assigned entitlements is stored in child process form tables.

37.2 Entitlement Data Capture Process

After you mark the entitlement attribute in each child process form, the following processes take place:

- [Capture of Data About Available Entitlements](#)
- [Capture of Data About Assigned Entitlements](#)

37.2.1 Capture of Data About Available Entitlements

The following steps describe how data about available entitlements is captured:

Note: You must mark the entitlement attribute in each child process form to enable the process described in these steps. The procedure is described later in this chapter.

1. Data about available entitlements is stored in the LKV table through synchronization with the target system.
2. You schedule and run the Entitlement List scheduled task.
3. The scheduled task identifies the entitlement attribute from the UD_ tables.
4. The scheduled task copies data about available entitlements from the LKV table to the ENT_LIST table.

37.2.2 Capture of Data About Assigned Entitlements

This section describes how data about assigned entitlements is captured.

Note: You must mark the entitlement attribute in each child process form UD_ table to enable the process described in these steps. The procedure is described later in this chapter.

To perform first-time synchronization of assigned entitlements:

1. You schedule the Entitlement Assignments scheduled task to run once.
2. The scheduled task identifies the entitlement attribute from the child process form (UD_) tables.
3. The scheduled task creates INSERT, UPDATE, and DELETE triggers on each UD_ table. The scheduled task also creates triggers on the OIU table.
4. The scheduled task copies data about assigned entitlements from the UD_ tables to the ENT_ASSIGN table.

Note: The ENT_ASSIGN table holds data about entitlement currently assigned to resources (users). When an entitlement is revoked, the record for that entitlement is moved out of this table to history data. Details are given in the "[Entitlement Updates](#)" section.

To perform incremental synchronization of assigned entitlements:

1. When a change is made to assigned entitlements through provisioning operations or reconciliation, the INSERT, UPDATE, or DELETE trigger copies the added, modified, or deleted row from the UD_ table to a staging table.
2. You configure and run the Entitlement Updates scheduled task.
3. For each record in the staging (ENT_ASSIGN_DELTA) table, the action taken by the scheduled task depends on the type of operation that was performed to the assigned entitlement:

Note: The type of operation (INSERT, UPDATE, or DELETE) is one of the data items stored in the staging table.

- **Event:** The entitlement was newly assigned to the account.

Action: A new record is created (copied from the staging table) in the ENT_ASSIGN table.

- **Event:** An existing entitlement was modified.

Action: The existing record is copied from the ENT_ASSIGN table into the ENT_ASSIGN_HIST table. The existing record is deleted from the ENT_ASSIGN table. A record corresponding to the newly modified entitlement is created in the ENT_ASSIGN table.

- **Event:** An existing entitlement was revoked.

Action: The existing record is copied from the ENT_ASSIGN table into the ENT_ASSIGN_HIST table. The existing record is deleted from the ENT_ASSIGN table.

37.3 Marking Entitlement Attributes on Child Process Forms

You must mark the entitlement attribute in the child process form UD_ table for resources for which you want to capture entitlement data. Suppose there are 15 target systems in your operating environment. If you want to capture entitlement data from 12 of 15 resources, then you must mark the entitlement attribute in those 12 resources.

Apply the following guidelines while performing the procedure described in this section:

- On a child process form, only one attribute holding entitlement data can be marked.
- The attribute that you mark must be of the LookupField type and its property must be one of the following:
 - Lookup code
 - Lookup query

The Lookup query must satisfy the following conditions:

- * The query uses the LKU and LKV tables
- * The Lookup code in the query is from the LKU table
- * The LKV_ENCODED column value is used for saving
- * The LKV_DECODED column value is used for display purposes

To mark a field as an entitlement in a child process form:

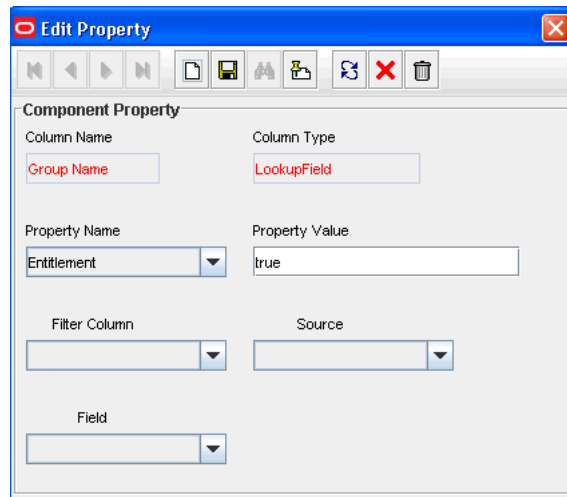
1. Log in to the Design Console.
2. Expand **Development Tools**, and then double-click **Form Designer**.
3. Search for and open the child form on which you want to mark an entitlement.
For example, you might want to mark an entitlement on the UD_ADUSRC child form.
4. Click **Create New Version**.
5. Enter a label for the new version, click the Save icon, and then close the dialog box.
6. From the **Current Version** list, select the version that you create.
7. On the Properties tab, select the field that you want to mark as an entitlement and then click **Add Property**.
8. From the Property Name list in the Add Property dialog box, select **Entitlement**.

Note: You can set Entitlement as the property of a field only if the column type is set to LookupField and the property name is set to Lookup Code.

9. In the **Property Value** field, enter `true`.

You need not specify values for any of the other fields in the dialog box.

The following screenshot shows the Edit Property dialog box for the lookup field:

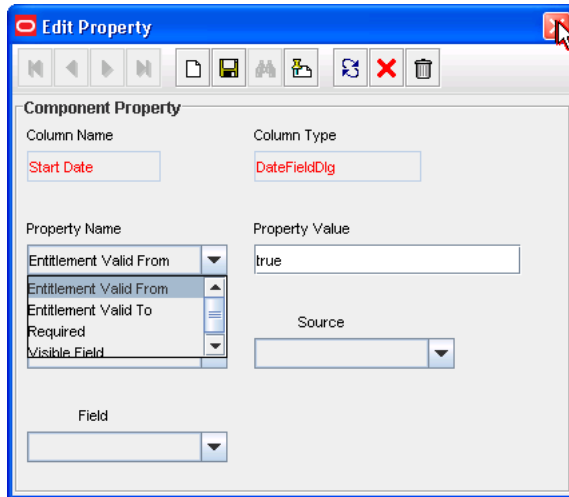


10. Click the Save icon and then close the dialog box.
11. If you want to enable the capture of Start Date and End Date values for the entitlement, then:

Note: You can enable the capture of the Start Date and End Date values only if the column type for both fields is DateFieldDlg.

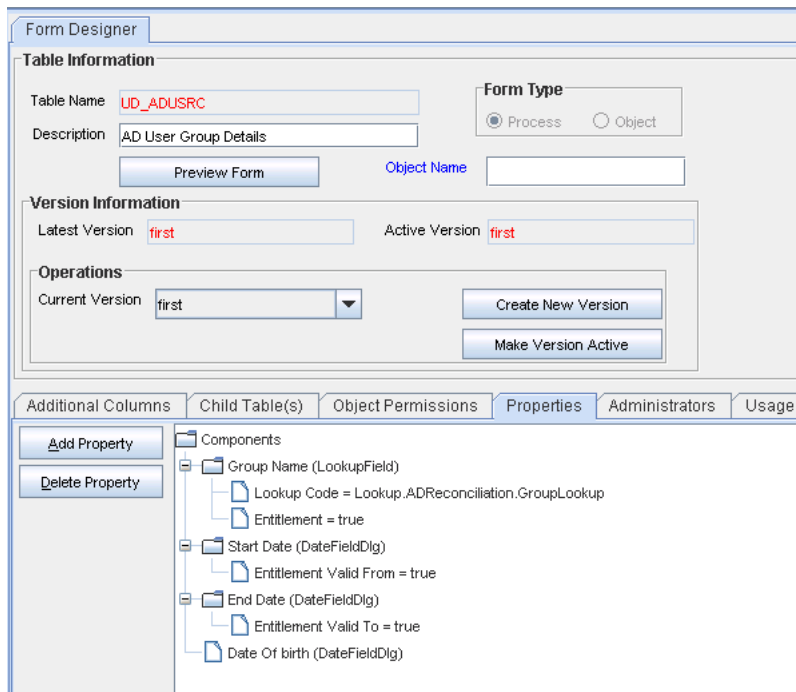
- a. On the Properties tab, select the **Start Date** field and then click **Add Property**.
- b. From the Property Name list in the Add Property dialog box, select **Entitlement Valid From**.
- c. In the Property Value field, enter `true`.
- d. Click the Save icon and then close the dialog box.
- e. On the Properties tab, select the **End Date** field and then click **Add Property**.
- f. From the Property Name list in the Add Property dialog box, select **Entitlement Valid To**.
- g. In the Property Value field, enter `true`.

The following screenshot shows the Edit Property dialog box for the Start Date field:



- h. Click the Save icon, and then close the dialog box.
- 12. Click the Save icon to save the changes made to the child process form.

The following screenshot shows the Properties tab of the child process form:



Note: Marking Start Date and End Date are optional.

- 13. Click Make Version Active.

37.4 Configuring Scheduled Tasks for Working with Entitlement Data

You configure the following scheduled tasks for working with entitlement data:

- [Entitlement List](#)

- [Entitlement Assignments](#)
- [Entitlement Updates](#)

37.4.1 Entitlement List

The Entitlement List scheduled task identifies the entitlement attribute from the child process form table and then copies entitlement data from the LKV table into the ENT_LIST table. A record created in the ENT_LIST table corresponds to an entitlement defined on a particular target system.

You must set a schedule for this task depending on how frequently new entitlements are defined on the target systems in your operating environment. In addition, you must run this scheduled task when new target systems are integrated with Oracle Identity Manager. In other words, you must run this task each time you mark a new entitlement. After the connector scheduled tasks fetch lookup field data from the target system into the LKV table, you can run the Entitlement List scheduled task to copy that entitlement data into the ENT_LIST table.

This scheduled task also handles updates to or deletion of entitlements from the target system. For example, if the Senior Accounts Analyst role is removed from the target system, then the connector scheduled task removes the entry for that role from the LKV table. When the Entitlement List scheduled task is run, it marks the row containing the role in the ENT_LIST table as a deleted row.

37.4.2 Entitlement Assignments

The Entitlement Assignments scheduled task is used for copying data about assigned entitlements into the ENT_ASSIGN table for the first time. This task identifies the entitlement attribute from the child process form table, and then copies data about assigned entitlements from the child process form table into the ENT_ASSIGN table. A record created in the ENT_ASSIGN table corresponds to an entitlement assigned to a particular user on a particular target system.

In addition, it creates INSERT, UPDATE, and DELETE triggers on the child process form tables from which it copies entitlement data. See "[Capture of Data About Assigned Entitlements](#)" for information about the function of these triggers.

You can use the RECORDS_TO_PROCESS_IN_BATCH attribute of this scheduled task to specify the number of records in each batch. The default batch size is 5000.

You must run this scheduled task the first time you start using entitlement data and whenever you mark a new child process form field as an entitlement.

37.4.3 Entitlement Updates

The Entitlement Updates scheduled task updates the ENT_ASSIGN table with changes to entitlement assignment data in the child process form tables. Triggers created by the Entitlement Assignments scheduled task copy changes made to entitlement assignment data into a staging table. The Entitlement Updates scheduled task processes data in the staging table and makes the required changes to data in the ENT_ASSIGN table. See "[Capture of Data About Assigned Entitlements](#)" on page 37-2 for information about the function of the INSERT, UPDATE, and DELETE triggers on the child process form tables.

37.5 Disabling the Capture of Modifications to Assigned Entitlements

You can manually disable incremental synchronization of assigned entitlement data in the ENT_ASSIGN table. In other words, you can disable the capture of modifications to assigned entitlements. To achieve this, you create and run an SQL script to drop the following triggers created on the child process form tables:

Note: These triggers are created by the Entitlement Assignments scheduled task.

- The OIU_U_TRG trigger created on the OIU table
- The following triggers created on the UD_ tables:
 - UD_TABLE_NAME_I_TRG
 - UD_TABLE_NAME_D_TRG
 - UD_TABLE_NAME_U_TRG

After you run the script, modifications to assigned entitlements are not copied into the staging table.

The following is a sample SQL script to drop the triggers on the child process form tables:

```
create or replace
TRIGGER UD_REQENTC_I_TRG
  AFTER INSERT ON UD_REQENTC
  FOR EACH ROW
  BEGIN
    INSERT INTO ENT_ASSIGN_DELTA(ENT_ASSIGN_DELTA_KEY,
orc_obi_key, ENT_OPERATION, DELTA_TABLE_OST_STATUS, DELTA_ENT_CODE, valid_from_date, OI
U_KEY, SDK_TYPE )
      VALUES(ENT_ASSIGN_DELTA_SEQ.nextval, :new.orc_key, 'grant', (select sdk_key
from sdk where sdk_name='UD_REQENTC'), :new.UD_REQENTC_LKUP, sysdate, (select
oiu.OIU_KEY from OIU oiu where oiu.ORG_KEY =:new.orc_key), 'P');
  END
```

UD_REQENTC is the child form name that contains the entitlement field. It is dynamically generated.

37.6 Entitlement-Related Reports

The following predefined reports provide data about assigned entitlements:

Note:

You must be a member of the ADMINISTRATORS group to be able to view these reports.

Duplicate assignments of the same entitlement to a particular user are suppressed in the reports because they are not copied to the ENT_ tables. For example, if user John Doe has been assigned the Sales Superintendent role twice on a target system, then the reports show only one instance of this entitlement.

- [Entitlement Access List](#)

- [Entitlement Access List History](#)
- [User Resource Entitlement](#)
- [User Resource Entitlement History](#)

37.6.1 Entitlement Access List

The Entitlement Access List report lists users who are currently assigned the entitlements that you specify while generating the report. The report provides basic information about the entitlements and the list of users to whom the entitlements are assigned.

37.6.2 Entitlement Access List History

The Entitlement Access List History report lists users who had been assigned the entitlements that you specify while generating the report. The report provides basic information about the entitlements and the list of users to whom the entitlements were assigned.

37.6.3 User Resource Entitlement

The User Resource Entitlement report lists the current entitlements of users whom you specify while generating the report. The report displays basic user information and entitlement details.

37.6.4 User Resource Entitlement History

The User Resource Entitlement History report lists details of past entitlements assigned to users whom you specify while generating the report. The report displays basic user information and entitlement details.

Part XI

Appendixes

This part contains the following appendixes:

- [Appendix A, "Scheduled Task Configuration File"](#)
- [Appendix B, "SPML Attributes and LDAP Mappings, and Oracle Identity Manager Attributes"](#)
- [Appendix C, "SPML Examples"](#)

Scheduled Task Configuration File

This appendix describes the structure and details of the XML file containing scheduler task definitions.

- [Structure of the Scheduler XML File](#)
- [The scheduledTasks Element](#)
- [The task Element](#)
- [The name Element](#)
- [The class Element](#)
- [The description Element](#)
- [The retry Element](#)
- [The parameters Element](#)
- [The string-param Element](#)
- [The number-param Element](#)
- [The boolean-param Element](#)

A.1 Structure of the Scheduler XML File

The following is a list of elements in the configuration XML file:

```
<scheduledTasks xmlns="http://xmlns.oracle.com/oim/scheduler">
  <task>
    <name>
    <class>
    <description>
    <retry>
    <parameters>
      <string-param>
        .....
      </string-param>

      <number-param>
        .....
      </number-param>

      <boolean-param>
        .....
      </boolean-param>
    </parameters>
  </task>
```

```
</scheduledTasks>
```

A.2 The scheduledTasks Element

The scheduledTasks element is the root element in XML used to define scheduled tasks.

[Table A-1](#) summarizes the properties of the scheduledTasks element.

Table A-1 *Properties of the scheduledTasks Element*

Property	Value
Parent Element	NA
Attributes	The XML namespace is specified as an attribute of the scheduledTasks element as follows: <pre><scheduledTasks xmlns="http://xmlns.oracle.com/oim/scheduler"></pre> <p>Note: The xmlns parameter is mandatory.</p>
Child Elements	task
Number of Occurrences	One for each scheduled task XML file to be created.
Element Value	NA
Mandatory or Optional?	Mandatory

A.3 The task Element

The task element is the child element of the scheduledTasks element.

You use the task element to define a scheduled task. The task element contains information about the scheduled task, for example, the name, class, description, and retry count of the scheduled task.

[Table A-2](#) summarizes the properties of the task element.

Table A-2 *Properties of the task Element*

Property	Value
Parent Element	scheduledTasks
Attributes	None
Child Elements	name, class, description, retry, and parameters
Number of Occurrences	One for each task to be created. <p>NOTE: If you want to define more than one task in a single scheduled task XML file, you must use one task element for every scheduled task being defined.</p>
Element Value	NA
Mandatory or Optional?	Mandatory

A.4 The name Element

The name element is the child element of the task element. The name element is used to specify the name of the scheduled task being created.

[Table A-3](#) summarizes the properties of the name element.

Table A-3 *Properties of the name Element*

Property	Value
Parent Element	task
Attributes	None
Child Elements	None
Number of Occurrences	One
Element Value	Name of the scheduled task being created. Note: The name of the scheduled task must be unique.
Mandatory or Optional?	Mandatory

A.5 The class Element

The class element is a mandatory element and is the child element of the task element. You use the class element to specify the name of the Java class that runs the scheduled task.

[Table A-4](#) summarizes the properties of the class element.

Table A-4 *Properties of the class Element*

Property	Value
Parent Element	task
Attributes	None
Child Elements	None
Number of Occurrences	One
Element Value	Name of the Java class that runs the scheduled task. See Section 6.4, "Develop the Scheduled Task Class" for information on developing a class for the scheduled task.
Mandatory or Optional?	Mandatory

A.6 The description Element

The description element is a mandatory element and is the child element of the task element. You can use the description element to provide a description of the task being created.

[Table A-5](#) summarizes the properties of the description element.

Table A-5 *Properties of the description Element*

Property	Value
Parent Element	task
Attributes	None
Child Elements	None
Number of Occurrences	One
Element Value	Description of the task being created
Mandatory or Optional?	Mandatory

A.7 The retry Element

Table A–6 summarizes the properties of the retry element.

Table A–6 Properties of the retry Element

Property	Value
Parent Element	task
Attributes	None
Child Elements	None
Number of Occurrences	One
Element Value	Number of seconds the scheduler must wait before it tries to schedule the task again
Mandatory or Optional?	Mandatory

A.8 The parameters Element

If you want to specify parameters at run time that the scheduled task requires for a successful job run, you must use the parameters element. For example, you might create a scheduled task that requires the user to specify the number of records to be retrieved at run time.

The parameters specified within this element are displayed under the Parameters section on the Create Job page.

Table A–7 summarizes the properties of the parameters element.

Table A–7 Properties of the parameters Element

Property	Value
Parent Element	task
Attributes	None
Child Elements	string-param, number-param, boolean-param
Number of Occurrences	One
Element Value	NA
Mandatory or Optional?	Optional

A.9 The string-param Element

You can use the string-param element to specify the name of the field that can take a value of the string data type. In other words, the string-param element specifies a label for the field that can hold a value of the string data type.

Table A–8 summarizes the properties of the string-param element.

Table A–8 Properties of the string-param Element

Property	Value
Parent Element	parameters
Attributes	required, helpText, encrypted
Child Elements	None
Number of Occurrences	One for every parameter of the string data type

Table A–8 (Cont.) Properties of the string-param Element

Property	Value
Element Value	Name of the string parameter
Mandatory or Optional?	Optional

As listed in [Table A–8](#), the string-param element contains the following attributes:

- **required**
This is a mandatory attribute and it can take a value of either `true` or `false`.
If the value of the required attribute is `true`, it is mandatory to enter a value for the parameter at run time.
If the value of the required attribute is `false`, it is not mandatory to enter a value for the parameter at run time.
- **helpText**
Use this attribute to specify the text that must appear at run time to help users know what to enter in the field. The text that is specified is usually the description of the field that is being created by the parameter.
- **encrypted**
By default, it has a value of `false` and this can take a value of either `true` or `false`.
If the value of the encrypted attribute is `true`, then the entered value for the parameter at run time is stored in encrypted form.
If the value of the required attribute is `false`, then the entered value for the parameter at run time is stored in plain text.

A.10 The number-param Element

You can use the number-param element to specify the name of the field that can take a value of the long data type.

[Table A–9](#) summarizes the properties of the number-param element.

Table A–9 Properties of the number-param Element

Property	Value
Parent Element	parameters
Attributes	required, helpText
Child Elements	None
Number of Occurrences	One for every parameter of the long data type
Element Value	Name of field that can hold a long data type
Mandatory or Optional?	Optional

The behavior and description of the `required` and `helpText` attributes for the number-param and string-param elements is the same. See "[The string-param Element](#)" on page A-4 for information about the `required` and `helpText` attributes.

A.11 The boolean-param Element

You can use the boolean-param element to specify the name of the field that can take a value of the boolean data type.

[Table A-10](#) summarizes the properties of the boolean-param element.

Table A-10 *Properties of the boolean-param Element*

Property	Value
Parent Element	parameters
Attributes	required, helpText
Child Elements	None
Number of Occurrences	One for every parameter of the boolean data type
Element Value	Name of field that can hold a boolean data type
Mandatory or Optional?	Optional

The behavior and description of the require and helpText attributes for the boolean-param element and the string-param element is the same. See [Section A.9, "The string-param Element"](#) for information about the require and helpText attributes.

B

SPML Attributes and LDAP Mappings, and Oracle Identity Manager Attributes

The SPML XSD Web Service uses Oracle Identity Manager as a back-end service to provide provisioning functionality to Fusion applications. A key building block of the SPML Web Service is the SPML Provisioning Service Object (PSO), which defines the object to be provisioned. Examples of PSO are identity and role.

This appendix shows the supported PSO attributes and their LDAP mappings, and explains the character restrictions on Oracle Identity Manager attributes. Finally, it describes additional operational data that the application can pass to the SPML Web Service. It contains the following sections:

- [Identity PSO Attributes](#)
- [Role PSO Attributes](#)
- [Preference Attributes](#)
- [Special Character Restrictions in Oracle Identity Manager Attributes](#)
- [Operation Data](#)

B.1 Identity PSO Attributes

Table B-1 shows identity attributes supported by the SPML implementation in Oracle Identity Manager and how these attributes map to LDAP objects/attributes.

Note: The syntax column lists relevant attribute properties such as the type, required, and so on.

Table B-1 Identity PSO Attributes

SPML Attribute Name	Syntax	Description	LDAP Mapping (Oracle Internet Directory)
ID	String, Read-Only, Required, Single	The identifier used to identify a user for modify request.	orclUserV2: orclguid
activeEndDate	Timestamp, Single	Termination time and date for the user	orclUserV2: orclActiveEndDate
activeStartDate	Timestamp, Single	Activation time and date for the user	orclUserV2: orclActiveStartDate

Table B-1 (Cont.) Identity PSO Attributes

SPML Attribute Name	Syntax	Description	LDAP Mapping (Oracle Internet Directory)
commonName	String, Required	The common names of the person, typically the person's full name and any variations of the same.	person: cn
countryName	String, Single	The business country of the person, expressed as a two-letter [ISO3166] country code.	orclUserV2: c
departmentNumber	String, Single	Codes for the departments within an organization to which this person belongs. This can be strictly numeric or alphanumeric.	inetOrgPerson: departmentNumber
description	String, Single	Human-readable descriptive phrases about the person.	person: description
displayName	String, Single, MLS	The preferred name to use when displaying an entry for the person. Provides MultiLingual Support (MLS) and also accepts language values for locale, for example "en" and "fr".	inetOrgPerson: displayName
employeeNumber	String, Single	Numeric or alphanumeric identifier assigned to a person, typically based on order of hire or association with an organization.	inetOrgPerson: employeeNumber
employeeType	String, Single	Identifies the type of employee. For the list of valid values see Table B-2 .	inetOrgPerson: employeeType
facsimileTelephoneNumber	String, Single	Telephone numbers for the person's business facsimile (FAX) terminals.	organizationalPerson: facsimileTelephoneNumber
generationQualifier	String, Single	Name strings that are typically the suffix part of the person's name (e.g. "III", "3rd", "Jr.").	N/A
givenName	String, Single	Name strings that are part of a person's name that is not their surname (for example, first name).	inetOrgPerson: givenName
hireDate	Timestamp, Single	Date of hire.	orclUserV2: orclHireDate
homePhone	Single, String	Home telephone numbers associated with the person.	inetOrgPerson:homePhone
homePostalAddress	Single, String	The home postal addresses of the person.	inetOrgPerson: homePostalAddress
initials	String, Single	Some or all of an individual's names, except the surname(s)	inetOrgPerson: initials
localityName	Single, String	Names of a business locality or place, such as a city, county, or other geographic region.	N/A
mail	Single, String	Business Internet mail addresses of the person in Mailbox [RFC2821] form.	inetOrgPerson: mail
manager	Single, String	The manager of the person.	N/A
middleName	String, Single	The middle names of the person.	orclUserV2: middleName

Table B-1 (Cont.) Identity PSO Attributes

SPML Attribute Name	Syntax	Description	LDAP Mapping (Oracle Internet Directory)
mobile	Single, String	Mobile telephone numbers associated with the person.	inetOrgPerson: mobile
organization	String, Single	Name of an organization—for example, my_company.	organization
organizationUnit	String, Single	Name of a unit within an organization, for example, IT Support.	organizationalUnitName
pager	Single, String	The business pager telephone numbers of the person.	inetOrgPerson: pager
password	String, Single	Password of the user.	person: userPassword
postalAddress	String, Single	Business addresses used by a Postal Service to perform services for the person.	organizationalPerson: postalAddress
postalCode	String, Single	Codes used by a Postal Service to identify postal service zones of the person's business.	organizationalPerson: postalCode
postOfficeBox	String, Single	Postal box identifiers that a Postal Service uses when a customer arranges to receive mail at a box on the premises of the Postal Service.	organizationalPerson: postOfficeBox
preferredLanguage	String, Single	The preferred written or spoken language for the person. This is useful for international correspondence or human-computer interaction. Values for this attribute type MUST conform to the definition of the Accept-Language header field defined in [RFC2068] with one exception: the sequence "Accept-Language" ":" should be omitted.	inetOrgPerson: preferredLanguage
state	String, Single	Full names of business states or provinces of the person.	organizationalPerson: st
street	String, Single	Site information from a business postal address (that is, the street name, place, avenue, and the house number) of the person.	organizationalPerson: street
surname	String, Single	Name strings for the family names (last name) of the person.	person: sn
telephoneNumber	String, Single	Business telephone number of the person	organizationalPerson: telephoneNumber
title	String, Single	Title of the person in their organizational context.	organizationalPerson: title
username	String, Single	Computer system login names associated with the person.	uid
userType	String, Single	The type of user. This attribute is used to provide Design Console access to the end-users. The allowed values are true and false.	

Table B-2 shows the valid values for the `employeeType` attribute:

Table B-2 Valid Values of `employeeType`

Value	Meaning
Full-Time	Full-Time Employee
Part-Time	Part-Time Employee
Temp	Temp
Intern	Intern
Consultant	Consultant
Contractor	Contractor
EMP	Employee
CWK	Contingent Worker
NONW	Non Worker
OTHER	Other Employee Type

Note: Oracle Identity Manager passes only the codes shown in the Value column; the meaning of each code is shown for reference.

B.1.1 Custom Identity Attributes

Custom attributes are provided to support Oracle Identity Manager functionality; these attributes are present in Oracle Identity Manager (such as when a user-defined field is added) but not in the PSO.

The custom attribute name must match the attribute name specified in the corresponding request dataset for the mapping to work end-to-end.

Here are some examples of custom attributes:

```
...
<data>
<pso:identity>
  <pso:attributes>
    <pso:attr name="Number Format">
      <pso:value>#,##0.##[.,]</pso:value>
    </pso:attr>
    <pso:attr name="Currency">
      <pso:value>USD</pso:value>
    </pso:attr>
  </attributes>
...

```

B.2 Role PSO Attributes

Table B-3 lists the role attributes supported by the SPML implementation in Oracle Identity Manager and how these attributes map to LDAP objects/attributes.

Table B-3 *PSO Role Attributes*

Attribute Name	Syntax	Description
ID	String, Read-Only, Required, Single	The PSO identifier that uniquely identifies a role. Usually directory GUID.
commonName	String, Required, MLS	The common name of the role.
description	Single	Human readable role description
displayName	String, Single, MLS	The preferred name to use when displaying an entry for the role.

B.2.1 Custom Role Attributes

Custom attributes are provided to support Oracle Identity Manager functionality; these attributes are present in Oracle Identity Manager but not in the PSO.

The custom attribute name must match the attribute name specified in the corresponding request dataset for the mapping to work end-to-end.

Here is an example of a custom role attribute:

```
...
<pso:attributes>
<pso:attr name="Role Category Name">
<pso:value>Cat1</pso:value>
</pso:attr>
...
```

Role Category Name is a special custom role attribute. It is the namespace for the roles. Each role belongs to a role category. This can be specified while creating a new role. If not specified, then the Default role category is selected. Each role category and role name uniquely identifies a role.

B.3 Preference Attributes

[Table B-4](#) lists the preference attributes supported by the SPML implementation in Oracle Identity Manager:

Table B-4 Preference Attributes

Attribute Name	Syntax	Description	LDAP Mapping
Number Format	String	The format to display numbers	orclNumberFormat Values are: ###0.##[,] ###0.###[\u00A0,] ###0.### ###0.###;###0.###- ###0.###[,] ###0.###;(###0.###)[,] ###0.##[\u00A0,] ###0.###['] ###0.###[']
Currency	String	The symbol that must be used for currency	orclCurrency Sample values are: USD YUN NZD INR

Table B-4 (Cont.) Preference Attributes

Attribute Name	Syntax	Description	LDAP Mapping
Date Format	String	The format to display the date	orclDateFormat Values are: MM-dd-yyyy MM-dd-yy MM.dd.yyyy MM.dd.yy MM/dd/yyyy MM/dd/yy M-d-yyyy M-d-yy M.d.yyyy M.d.yy M/d/yyyy M/d/yy dd-MM-yyyydd-MM-yy d-M-yyyy d-M-yy dd.MM.yyyy dd.MM.yy d.M.yyyy d.M.yy dd/MM/yyyy dd/MM/yy d/M/yyyy d/M/yy yyyy-MM-dd yy-MM-dd yyyy-M-d yy-M-d yyyy.MM.dd yy.MM.dd yyyy.M.d yy.M.d yy. M. d yyyy/MM/dd yy/MM/dd yyyy/M/d yy/M/d

Table B-4 (Cont.) Preference Attributes

Attribute Name	Syntax	Description	LDAP Mapping
Time Format	String	The format to display the time	orclTimeFormat Values are: HH.mm HH.mm.ss HH:mm HH:mm:ss H:mm H:mm:ss H.mm H.mm.ss a hh.mm a hh.mm.ss a hh:mm a hh:mm:ss ah:mm ah:mm:ss hh.mm a hh.mm.ss a hh:mm a hh:mm:ss a
Embedded Help	String	Whether or not to show embedded help	orclEmbeddedHelp Values are: true false
Font Size	String	The size of the font	orclFontSize Values are: LARGE MEDIUM
Color Constrast	String	Constrast of the color	orclColorContrast Values are: STANDARD HIGH

Table B–4 (Cont.) Preference Attributes

Attribute Name	Syntax	Description	LDAP Mapping
Accessibility Mode	String	Accessibility mode for the user	orclAccessibilityMode Values are: screenReader inaccessible default
FA Language	String	The default preference language	orclFALanguage
User Name Preferred Language	String	The preference language of the user used to only show the display name of the user in that language Note: The value set for this attribute is not used in Oracle Identity Manager.	orclDisplayNameLanguagePreference

B.4 Special Character Restrictions in Oracle Identity Manager Attributes

This section lists character restrictions applicable to Oracle Identity Manager attributes. Failure to observe these restrictions will cause errors when performing operations with attributes.

- [Characters Available in All Attributes](#)
- [Special Characters in the Password Field](#)
- [Usage of Single Quotation Mark](#)
- [Usage of Semicolon](#)
- [Unsupported Special Characters](#)

B.4.1 Characters Available in All Attributes

Alphanumeric characters (a through z, A through Z, and 0 through 9) and the underscore character (_) can be used in all Oracle Identity Manager attributes.

B.4.2 Special Characters in the Password Field

The following special characters can be used in the Password field:

- Percent sign (%)
- Plus sign (+)
- Equal sign (=)
- Comma (,)
- Backslash (\)
- Single quotation mark (')
- Slash (/)
- Vertical bar (|)

B.4.3 Usage of Single Quotation Mark

The single quotation mark (') can be used *only* in the following attributes:

- Login
- Manager ID
- First Name
- Last Name
- Middle Name
- Group Name
- Organization Name
- Resource Name

B.4.4 Usage of Semicolon

The semicolon (;) can be used only in access policy names.

B.4.5 Unsupported Special Characters

The following special characters are not supported in *any* Oracle Identity Manager attribute:

- Period (.)
- Number sign (#)
- Slash (/)
- Percent sign (%)
- Equal sign (=)
- Vertical bar (|)
- Plus sign (+)
- Comma (,)
- Backslash (\)
- Double quotation mark (")
- Less than symbol (<)
- Greater than symbol (>)

B.5 Operation Data

Requesting application such as HCM Fusion Application will act as a SPML requestor. In addition to PSO data, the application can also pass some operational data to the SPML Web Service. This section describes how applications can pass the operation data.

- [Passing Operation Data](#)
- [Passing Reference Data](#)

B.5.1 Passing Operation Data

It is possible to pass a requestor ID for each operation. When the Fusion application supplies credentials in a request, that is an application ID. For auditing purposes, it is also possible to pass a requestor ID. Oracle Identity Manager audits this ID, instead of the application ID, as the actual requestor of the operation.

Along with the requestorID, a justification for the request can also be specified.

The following is an example of the operation data:

```
...
</pso:identity>
</data>
<capabilityData
capabilityURI="http://xmlns.oracle.com/idm/identity/OperationData"
mustUnderstand="true">
<operationData
xmlns="http://xmlns.oracle.com/idm/identity/OperationData" requestorGUID="1"
justification="i need this account">
</capabilityData>
</addRequest>
```

B.5.2 Passing Reference Data

The application is also required to pass some reference data to SPML so that when a callback is received, it can be identified with the reference data for the callback in context. This is pass-through data, which is ignored by Oracle Identity Manager, but will be returned in the callback.

The following is an example that contains the <LdapRequestId>:

```
...
...
</pso:identity>
</data>
<capabilityData
capabilityURI="http://xmlns.oracle.com/idm/identity/OperationData"
mustUnderstand="true">
<operationData
xmlns="http://xmlns.oracle.com/idm/identity/OperationData" requestorGUID="1"
justification="i need this account">
<LdapRequestId
xmlns="http://xmlns.oracle.com/apps/hcm/users/ldapRequestService/">102329090340
</operationData>
</capabilityData>
</addRequest>
```

SPML Examples

This appendix provides the following SPML XSD examples:

- [SPML Example - Add User](#)
- [SPML Example - Delete User](#)
- [SPML Example - Modify User](#)
- [SPML Example - Resume User](#)
- [SPML Example - Suggest User Name](#)
- [SPML Example - Suspend User](#)
- [SPML Example - Validate User Name](#)
- [SPML Example - Check If User is Active](#)
- [SPML Example - Lookup Username Policy](#)
- [SPML Example - Assign Role Membership](#)
- [SPML Example – Add User with Role Assignment](#)
- [SPML Example - Add User Request with Notification](#)
- [SPML Example – Revoke Role Membership](#)
- [SPML Example - Add Role](#)
- [SPML Example - Add Role with Parent](#)
- [SPML Example - Modify Role](#)
- [SPML Example - Add Parent to a Role](#)
- [SPML Example - Role Grant](#)
- [SPML Example - Delete Role](#)
- [SPML Example - Status Request](#)
- [SPML Example - Reset Password](#)
- [SPML Example - Reset Password with Notification](#)
- [SPML Example - Lookup User Name Policy](#)
- [SPML Example - Cancel Request](#)
- [SPML Example - Batch Request](#)

C.1 SPML Example - Add User

The Request is as follows:

```
<addRequest xmlns="urn:oasis:names:tc:SPML:2:0"
xmlns:psd="http://xmlns.oracle.com/idm/identity/PSO" executionMode="asynchronous"
locale="en" policyURI="http://www.sample.com/string/string" requestID="string"
returnData="identifier" targetID="string">
<!--Zero or more repetitions:-->
<data>
<!--You have a CHOICE of the next 3 items at this level-->
<psd:identity>
<!--Optional:-->
<psd:attributes>
<!--Here, My Attribute is a UDF, with 'My Attribute' also added in
CreateUserDataset.xml -->
<psd:attr name="My Attribute">
<psd:value>New Value</psd:value>
</psd:attr>
</psd:attributes>
<!--Optional:-->
<psd:activeEndDate>2009-06-12T16:00:00</psd:activeEndDate>
<!--Optional:-->
<psd:activeStartDate>2009-06-11T18:00:00</psd:activeStartDate>
<psd:commonName>
<psd:value>All Optional Values</psd:value>
</psd:commonName>
<!--Optional:-->
<psd:countryName>India</psd:countryName>
<!--Optional:-->
<psd:departmentNumber>
<!--1 or more repetitions:-->
<psd:value>123456</psd:value>
</psd:departmentNumber>
<!--Optional:-->
<psd:description>
<!--1 or more repetitions:-->
<psd:values>
<!--1 or more repetitions:-->
<psd:value>All Optional Fields Profile</psd:value>
</psd:values>
</psd:description>
<!--Optional:-->
<psd:displayName>
<!--1 or more repetitions:-->
<psd:value locale="en">All Optional Values</psd:value>
</psd:displayName>
<!--Optional:-->
<psd:employeeNumber>24073</psd:employeeNumber>
<!--Optional:-->
<psd:employeeType>
<!--1 or more repetitions:-->
<psd:values>
<!--1 or more repetitions:-->
<psd:value>Part-Time</psd:value>
</psd:values>
</psd:employeeType>
<!--Optional:-->
<psd:facsimileTelephoneNumber>
<!--1 or more repetitions:-->
<psd:number>08041085304</psd:number>
```

```

</pso:facsimileTelephoneNumber>
<!--Optional:-->
<pso:generationQualifier>
<!--1 or more repetitions:-->
<pso:value>II</pso:value>
</pso:generationQualifier>
<!--Optional:-->
<pso:givenName>
<!--1 or more repetitions:-->
<pso:value>OptionalGivenName</pso:value>
</pso:givenName>
<!--Optional:-->
<pso:hireDate>2009-06-11T16:00:00</pso:hireDate>
<!--Optional:-->
<pso:homePhone>
<!--1 or more repetitions:-->
<pso:number>999999999</pso:number>
</pso:homePhone>
<!--Optional:-->
<pso:homePostalAddress>
<!--1 or more repetitions:-->
<pso:value>marathahalli</pso:value>
</pso:homePostalAddress>
<!--Optional:-->
<pso:initials>
<!--1 or more repetitions:-->
<pso:value>SJ</pso:value>
</pso:initials>
<!--Optional:-->
<pso:localityName>
<!--1 or more repetitions:-->
<pso:value>Munekolala</pso:value>
</pso:localityName>
<!--Optional:-->
<!--pso:mail>
<pso:value>jdong12@oracle.com</pso:value>
</pso:mail-->
<!--Optional:-->
<pso:middleName>MiddleName</pso:middleName>
<!--Optional:-->
<pso:mobile>
<!--1 or more repetitions:-->
<pso:number>9886078373</pso:number>
</pso:mobile>
<!--Optional:-->
<pso:organization>
<pso:value>2</pso:value>
</pso:organization>
<!--Optional:-->
<pso:organizationUnit>
<pso:value>Marketing</pso:value>
</pso:organizationUnit>
<!--Optional:-->
<pso:pager>
<!--1 or more repetitions:-->
<pso:number>7777</pso:number>
</pso:pager>
<!--Optional:-->
<pso:password>
<!--1 or more repetitions:-->

```

```

<pso:value>saijha</pso:value>
</pso:password>
<!--Optional:-->
<pso:postalAddress>
<!--1 or more repetitions:-->
<pso:value>Marathahalli</pso:value>
</pso:postalAddress>
<!--Optional:-->
<pso:postalCode>
<!--1 or more repetitions:-->
<pso:value>560037</pso:value>
</pso:postalCode>
<!--Optional:-->
<pso:postOfficeBox>
<!--1 or more repetitions:-->
<pso:value>999</pso:value>
</pso:postOfficeBox>
<!--Optional:-->
<pso:preferredLanguage>en</pso:preferredLanguage>
<!--Optional:-->
<pso:state>
<!--1 or more repetitions:-->
<pso:value>Karnataka</pso:value>
</pso:state>
<!--Optional:-->
<pso:street>
<!--1 or more repetitions:-->
<pso:value>Satyam Street</pso:value>
</pso:street>
<!--Optional:-->
<pso:surname>
<pso:values>
  <!--1 or more repetitions:-->
<pso:value>Jha</pso:value>
</pso:values>
</pso:surname>
<!--Optional:-->
<pso:telephoneNumber>
<!--1 or more repetitions:-->
<pso:number>08041085304</pso:number>
</pso:telephoneNumber>
<!--Optional:-->
<pso:title>
<pso:value>Mr</pso:value>
</pso:title>
<!--Optional:-->
<pso:username>
<!--1 or more repetitions:-->
<pso:value>jsmith</pso:value>
</pso:username>
<pso:manager>5</pso:manager>
</pso:identity>
</data>
</addRequest>

```

The Add User Response sample if user login already exists is as follows:

```

<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"><env:Header/>
<env:Body>
<ns3:addResponse xmlns:ns2="http://xmlns.oracle.com/idm/identity/PSO"
xmlns:ns3="urn:oasis:names:tc:SPML:2:0"

```

```

xmlns:ns4="urn:oasis:names:tc:SPML:2:0:suspend"
xmlns:ns5="http://xmlns.oracle.com/idm/identity/spmlv2custom/Username"
xmlns:ns6="urn:oasis:names:tc:SPML:2:0:password"
xmlns:ns7="urn:oasis:names:tc:SPML:2:0:reference"
xmlns:ns8="urn:oasis:names:tc:SPML:2:0:async" xmlns:ns9="urn:names:spml:ws:header"
status="failure" error="malformedRequest" extendedError="IAM-3076048">
<ns3:errorMessage>username jsmith already exists.</ns3:errorMessage>
</ns3:addResponse>
</env:Body>
</env:Envelope>

```

The Add User Response sample if multiple values are passed for attributes that accept only single value:

```

<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
<env:Header/>
<env:Body>
<ns3:addResponse xmlns:ns2="http://xmlns.oracle.com/idm/identity/PSO"
xmlns:ns3="urn:oasis:names:tc:SPML:2:0"
xmlns:ns4="urn:oasis:names:tc:SPML:2:0:suspend"
xmlns:ns5="http://xmlns.oracle.com/idm/identity/spmlv2custom/Username"
xmlns:ns6="urn:oasis:names:tc:SPML:2:0:password"
xmlns:ns7="urn:oasis:names:tc:SPML:2:0:reference"
xmlns:ns8="urn:oasis:names:tc:SPML:2:0:async" xmlns:ns9="urn:names:spml:ws:header"
status="pending" requestID="5" error="malformedRequest"
extendedError="IAM-3071022"><ns3:errorMessage>The attribute commonName is not
multi-language enabled in OIM. Only the value John Smith will be
saved.</ns3:errorMessage>
<ns3:errorMessage>The attribute organization is not multi-language enabled in OIM.
Only the value 1 will be saved.
</ns3:errorMessage>
</ns3:addResponse>
</env:Body>
</env:Envelope>

```

Note:

- To find the status of the add user request, see ["SPML Example - Status Request"](#) on page C-18.
 - The displayName attribute has Multiple Language Support (MLS), and language values can be specified as "en", "fr", and so on.
-
-

C.2 SPML Example - Delete User

The Request is as follows:

```

<deleteRequest xmlns="urn:oasis:names:tc:SPML:2:0"
xmlns:ps0="http://xmlns.oracle.com/idm/identity/PSO" executionMode="asynchronous"
locale="en" policyURI="http://www.sample.com/string/string"
requestID="string" returnData="identifier" targetID="string">
<ps0ID ID="identity:6C9B96E99FC8DC32E040E50A3D5252F5" />
</deleteRequest>

```

The Response is as follows:

```

<ns9:ResponseType xmlns="http://xmlns.oracle.com/idm/identity/PSO"
xmlns:ns2="urn:oasis:names:tc:SPML:2:0"

```

```

xmlns:ns3="urn:oasis:names:tc:SPML:2:0:reference"
xmlns:ns4="urn:oasis:names:tc:SPML:2:0:password"
xmlns:ns5="urn:oasis:names:tc:SPML:2:0:suspend"
xmlns:ns6="http://xmlns.oracle.com/idm/identity/spmlv2custom/Username"
xmlns:ns7="urn:names:spml:ws:header" xmlns:ns8="urn:oasis:names:tc:SPML:2:0:async"
xmlns:ns9="oasis:names:tc:SPML:2:0" requestID="19" status="pending"/>

```

C.3 SPML Example - Modify User

The Request is as follows:

```

<modifyRequest xmlns="urn:oasis:names:tc:SPML:2:0"
xmlns:ps0="http://xmlns.oracle.com/idm/identity/PSO" executionMode="asynchronous"
locale="string" policyURI="http://www.sample.com/string/string"
requestID="string" returnData="identifier">
<capabilityData capabilityURI="urn:oasis:names:tc:SPML:2:0:reference"
mustUnderstand="true" />
<ps0ID ID="identity:6C9B96E99FC8DC32E040E50A3D5252F5" />
<modification modificationMode="add">
<component path="/identity" namespaceURI="http://www.w3.org/TR/xpath20" />
<data>
<ps0:identity>
<ps0:initials>
<!--1 or more repetitions:-->
<ps0:value>J S</ps0:value>
</ps0:initials>
</ps0:identity>
</data>
</modification>
<modification modificationMode="replace">
<component path="/identity" namespaceURI="http://www.w3.org/TR/xpath20" />
<data>
<ps0:identity>
<ps0:localityName>
<!--1 or more repetitions:-->
<ps0:value>new_locality</ps0:value>
</ps0:localityName>
<ps0:homePhone>
<!--1 or more repetitions:-->
<ps0:number>0123456789</ps0:number>
</ps0:homePhone>
<ps0:commonName>
<!--1 or more repetitions:-->
<ps0:values>
<!--1 or more repetitions:-->
<ps0:value>FR Alice Krug1</ps0:value>
</ps0:values>
</ps0:commonName>
</ps0:identity>
</data>
</modification>
<modification modificationMode="delete">
<component path="/identity" namespaceURI="http://www.w3.org/TR/xpath20" />
<data>
<ps0:identity>
<ps0:pager>
<!--1 or more repetitions:-->
<ps0:number>333</ps0:number>
</ps0:pager>

```



```

</pso:identity>
</data>
</modification>
</modifyRequest>

```

The Response is as follows:

```

<ns9:ModifyResponseType xmlns="http://xmlns.oracle.com/idm/identity/PSO"
xmlns:ns2="urn:oasis:names:tc:SPML:2:0"
xmlns:ns3="urn:oasis:names:tc:SPML:2:0:reference"
xmlns:ns4="urn:oasis:names:tc:SPML:2:0:password"
xmlns:ns5="urn:oasis:names:tc:SPML:2:0:suspend"
xmlns:ns6="http://xmlns.oracle.com/idm/identity/spmlv2custom/Username"
xmlns:ns7="urn:names:spml:ws:header" xmlns:ns8="urn:oasis:names:tc:SPML:2:0:async"
xmlns:ns9="oasis:names:tc:SPML:2:0" requestID="15" status="pending"/>

```

C.4 SPML Example - Resume User

The Request is as follows:

```

<resumeRequest xmlns="urn:oasis:names:tc:SPML:2:0:suspend"
requestID="120">
<psoID ID="6C9B96E99FC8DC32E040E50A3D5252F5" />
</resumeRequest>

```

The Response is as follows:

```

<ns9:ResponseType xmlns="urn:oasis:names:tc:SPML:2:0"
xmlns:ns2="http://xmlns.oracle.com/idm/identity/PSO"
xmlns:ns3="urn:oasis:names:tc:SPML:2:0:password"
xmlns:ns4="urn:oasis:names:tc:SPML:2:0:suspend"
xmlns:ns5="http://xmlns.oracle.com/idm/identity/spmlv2custom/Username"
xmlns:ns6="urn:names:spml:ws:header" xmlns:ns7="urn:oasis:names:tc:SPML:2:0:async"
xmlns:ns8="urn:oasis:names:tc:SPML:2:0:reference"
xmlns:ns9="oasis:names:tc:SPML:2:0" requestID="120" status="pending"/>

```

C.5 SPML Example - Suggest User Name

The Request is as follows:

```

<ns4:suggestUsernameRequest
xmlns:ns4="http://xmlns.oracle.com/idm/identity/spmlv2custom/Username"
xmlns="http://xmlns.oracle.com/idm/identity/spmlv2custom/Username"
xmlns:ns2="http://xmlns.oracle.com/idm/identity/spmlv2custom/Username"
xmlns:ns3="http://xmlns.oracle.com/idm/identity/PSO">
<ns2:identity>
<ns3:givenName>
<ns3:value>testfn</ns3:value>
</ns3:givenName>
<ns3:surname>
<ns3:values>
<ns3:value>testln</ns3:value>
</ns3:values>
</ns3:surname>
</ns2:identity>
</ns4:suggestUsernameRequest>

```

The Response is as follows:

```
<ns9:SuggestUsernameResponseType xmlns="urn:oasis:names:tc:SPML:2:0"
xmlns:ns2="http://xmlns.oracle.com/idm/identity/PSO"
xmlns:ns3="http://xmlns.oracle.com/idm/identity/spmlv2custom/Username"
xmlns:ns4="urn:oasis:names:tc:SPML:2:0:reference"
xmlns:ns5="urn:oasis:names:tc:SPML:2:0:async"
xmlns:ns6="urn:oasis:names:tc:SPML:2:0:password"
xmlns:ns7="urn:names:spml:ws:header"
xmlns:ns8="urn:oasis:names:tc:SPML:2:0:suspend"
xmlns:ns9="oasis:names:tc:SPML:2:0" status="success">
<ns3:username>testfn.testln@mycompany.com</ns3:username>
</ns9:SuggestUsernameResponseType>
```

C.6 SPML Example - Suspend User

The Request is as follows:

```
<suspendRequest xmlns="urn:oasis:names:tc:SPML:2:0:suspend"
requestID="139">
<psoid ID="6C9B96E99FC8DC32E040E50A3D5252F5" />
</suspendRequest>
```

The Response is as follows:

```
<ns9:ResponseType xmlns="urn:oasis:names:tc:SPML:2:0"
xmlns:ns2="http://xmlns.oracle.com/idm/identity/PSO"
xmlns:ns3="urn:oasis:names:tc:SPML:2:0:password"
xmlns:ns4="urn:oasis:names:tc:SPML:2:0:suspend"
xmlns:ns5="http://xmlns.oracle.com/idm/identity/spmlv2custom/Username"
xmlns:ns6="urn:names:spml:ws:header" xmlns:ns7="urn:oasis:names:tc:SPML:2:0:async"
xmlns:ns8="urn:oasis:names:tc:SPML:2:0:reference"
xmlns:ns9="oasis:names:tc:SPML:2:0" requestID="28"
status="pending" /><ns9:ResponseType xmlns="urn:oasis:names:tc:SPML:2:0"
xmlns:ns2="http://xmlns.oracle.com/idm/identity/PSO"
xmlns:ns3="urn:oasis:names:tc:SPML:2:0:password"
xmlns:ns4="urn:oasis:names:tc:SPML:2:0:suspend"
xmlns:ns5="http://xmlns.oracle.com/idm/identity/spmlv2custom/Username"
xmlns:ns6="urn:names:spml:ws:header" xmlns:ns7="urn:oasis:names:tc:SPML:2:0:async"
xmlns:ns8="urn:oasis:names:tc:SPML:2:0:reference"
xmlns:ns9="oasis:names:tc:SPML:2:0" requestID="139" status="pending" />
```

C.7 SPML Example - Validate User Name

The Request is as follows:

```
<validateUsernameRequest
xmlns="http://xmlns.oracle.com/idm/identity/spmlv2custom/Username">
<username>testfn.testln</username>
</validateUsernameRequest>
```

The Response is as follows:

```
<ns9:ValidateUsernameResponseType xmlns="urn:oasis:names:tc:SPML:2:0"
xmlns:ns2="http://xmlns.oracle.com/idm/identity/PSO"
xmlns:ns3="http://xmlns.oracle.com/idm/identity/spmlv2custom/Username"
xmlns:ns4="urn:oasis:names:tc:SPML:2:0:reference"
xmlns:ns5="urn:oasis:names:tc:SPML:2:0:async"
xmlns:ns6="urn:oasis:names:tc:SPML:2:0:password"
xmlns:ns7="urn:names:spml:ws:header"
```

```
xmlns:ns8="urn:oasis:names:tc:SPML:2:0:suspend"
xmlns:ns9="oasis:names:tc:SPML:2:0" valid="true" status="success"/>
```

C.8 SPML Example - Check If User is Active

The request is as follows:

```
<activeRequest xmlns="urn:oasis:names:tc:SPML:2:0:suspend" requestID="143">
<psoID ID="5" targetID="string"/>
</activeRequest>
```

The Response is as follows:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns4:ResponseType xmlns="urn:oasis:names:tc:SPML:2:0"
xmlns:ns2="urn:oasis:names:tc:SPML:2:0:suspend"
xmlns:ns3="http://xmlns.oracle.com/idm/identity/PSO"
xmlns:ns4="oasis:names:tc:SPML:2:0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="ns2:ActiveResponseType" active="true" requestID="143"
status="success" />
```

C.9 SPML Example - Lookup Username Policy

The Request is as follows:

```
<lookupUsernamePolicyRequest
xmlns="http://xmlns.oracle.com/idm/identity/spmlv2custom/Username">
</lookupUsernamePolicyRequest>
```

The Response is as follows:

```
<ns9:LookupUsernamePolicyResponseType
xmlns="urn:oasis:names:tc:SPML:2:0"
xmlns:ns2="http://xmlns.oracle.com/idm/identity/PSO"
xmlns:ns3="http://xmlns.oracle.com/idm/identity/spmlv2custom/Username"
xmlns:ns4="urn:oasis:names:tc:SPML:2:0:reference"
xmlns:ns5="urn:oasis:names:tc:SPML:2:0:async"
xmlns:ns6="urn:oasis:names:tc:SPML:2:0:password"
xmlns:ns7="urn:names:spml:ws:header"
xmlns:ns8="urn:oasis:names:tc:SPML:2:0:suspend"
xmlns:ns9="oasis:names:tc:SPML:2:0" status="success"
<ns3:description>Generates user name based on email id if it is available, else
generate based on first name and last name appended with domain
name.</ns3:description>
>
```

Note: To view policy description in a specific locale, you can set locale attribute in the payload. If this locale is not supported, then by is displayed in the server locale by default, as shown:

```
<lookupUsernamePolicyRequest locale="th"
xmlns="http://xmlns.oracle.com/idm/identity/spmlv2custom/Username">
</lookupUsernamePolicyRequest>
```

C.10 SPML Example – Add User with Role Assignment

The Request to create user (identity) is as follows:

Note:

- There can only be one `topsoID` element under a reference element. For multiple roles, individual reference element must be used.
 - The GUID must be of 32 characters for all requests.
-
-

```
<addRequest
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="urn:oasis:names:tc:SPML:2:0"
  xmlns:spml="urn:oasis:names:tc:SPML:2:0"
  executionMode="asynchronous"
  policyURI="create_identity_policy_pr02.xml">
  <spml:data xsi:type="spml:PSOType">
    <identity
      xmlns="http://xmlns.oracle.com/idm/identity/PSO"
      xmlns:ps0="http://xmlns.oracle.com/idm/identity/PSO">
      <ps0:commonName>
        <ps0:values>
          <ps0:value>John Doe</ps0:value>
        </ps0:values>
      </ps0:commonName>
      <ps0:displayName>
        <ps0:value>John Doe</ps0:value>
      </ps0:displayName>
      <ps0:givenName>
        <ps0:value>John</ps0:value>
      </ps0:givenName>
      <ps0:mail>
        <ps0:value>john.doe@acme.com</ps0:value>
      </ps0:mail>
      <ps0:middleName/>
      <ps0:organization>
        <ps0:values>
          <ps0:value>ACME, Inc.</ps0:value>
        </ps0:values>
      </ps0:organization>
      <ps0:password>
        <ps0:value>qwert</ps0:value>
      </ps0:password>
      <ps0:surname>
        <ps0:values>
          <ps0:value>Doe</ps0:value>
        </ps0:values>
      </ps0:surname>
      <ps0:username>
        <ps0:value>jdoe</ps0:value>
      </ps0:username>
    </identity>
  </spml:data>
  <spml:capabilityData
    capabilityURI="urn:oasis:names:tc:SPML:2:0:reference"
    mustUnderstand="true" >
    <reference xmlns="urn:oasis:names:tc:SPML:2:0:reference"
```

```

        typeOfReference="memberOf">
        <toPsoID ID="15"/>
<!--To make the user a member of a default role-->
        </reference>

        <reference xmlns="urn:oasis:names:tc:SPML:2:0:reference"
        typeOfReference="memberOf">
        <toPsoID ID="6C9B96E99FC8DC32E040E50A3D5252F5" />
        </reference>
    </spml:capabilityData>
</addRequest>

```

The Response is as follows:

```

<spml:addResponse
    xmlns:spml="urn:oasis:names:tc:SPML:2:0"
    status="pending"
    requestID="10821" />

```

The Add User with Role Assignment response sample containing partial invalid roles is as follows:

```

<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
<env:Header/>
<env:Body>
<ns3:addResponse xmlns:ns2="http://xmlns.oracle.com/idm/identity/PSO"
xmlns:ns3="urn:oasis:names:tc:SPML:2:0"
xmlns:ns4="urn:oasis:names:tc:SPML:2:0:suspend"
xmlns:ns5="http://xmlns.oracle.com/idm/identity/spmlv2custom/Username"
xmlns:ns6="urn:oasis:names:tc:SPML:2:0:password"
xmlns:ns7="urn:oasis:names:tc:SPML:2:0:reference"
xmlns:ns8="urn:oasis:names:tc:SPML:2:0:async" xmlns:ns9="urn:names:spml:ws:header"
status="pending" requestID="5" error="malformedRequest"
extendedError="IAM-3071022">
<ns3:errorMessage>Request contains an invalid Id/Guid identifier -
xyzxyzxyz.</ns3:errorMessage>
</ns3:addResponse>
</env:Body>
</env:Envelope>

```

C.11 SPML Example - Assign Role Membership

The Request example is as follows:

```

<modifyRequest
xmlns="urn:oasis:names:tc:SPML:2:0"
xmlns:ps0="http://xmlns.oracle.com/idm/identity/PSO"
executionMode="asynchronous"
locale="en"
policyURI="gant_role_01">
<ps0ID ID="identity:6C9B96E99FC8DC32E040E50A3D5252F5" />
<modification modificationMode="add">
<capabilityData
        capabilityURI="urn:oasis:names:tc:SPML:2:0:reference"
    mustUnderstand="true">
<reference
        xmlns="urn:oasis:names:tc:SPML:2:0:reference"
    typeOfReference="memberOf">
<toPsoID ID="6C9B96E99FC8DC32E040E50A3D5252F5" />

```

```
</reference>
</capabilityData>
</modification>
</modifyRequest>
```

The Response example is as follows:

```
<spml:modifyResponse
  xmlns:spml="urn:oasis:names:tc:SPML:2:0"
  status="pending"
  requestID="10822"/>
```

C.12 SPML Example - Add User Request with Notification

The request is as follows:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:ns1="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
  <soap:Header>
    <ns1:Security>
      <ns1:UsernameToken>
        <ns1:Username>SYSTEM_ADMINISTRATOR_LOGIN</ns1:Username>
        <ns1:Password>SYSTEM_ADMINISTRATOR_PASSWORD</ns1:Password>
      </ns1:UsernameToken>
    </ns1:Security>
  </soap:Header>
  <soap:Body xmlns:ns2="urn:oasis:names:tc:SPML:2:0">
    <addRequest xmlns="urn:oasis:names:tc:SPML:2:0"
      xmlns:ps0="http://xmlns.oracle.com/idm/identity/PSO"
      executionMode="asynchronous"
      locale="en_US">
      <data>
        <ps0:identity>
          <ps0:commonName>
            <ps0:value>sample_person1</ps0:value>
          </ps0:commonName>
          <ps0:countryName>Country1</ps0:countryName>
          <ps0:displayName>
            <ps0:value locale="fr">sample_user1_2fr </ps0:value>
            <ps0:value locale="base">sample_user12_tenant1</ps0:value>
          </ps0:displayName>
          <ps0:employeeType>
            <ps0:values>
              <ps0:value>Full-Time</ps0:value>
            </ps0:values>
          </ps0:employeeType>
          <ps0:givenName>
            <ps0:value>sample_person1_fn</ps0:value>
          </ps0:givenName>
          <ps0:homePhone>
            <ps0:number>8888888888</ps0:number>
          </ps0:homePhone>
```

```

        <ps0:localityName>
            <ps0:value>Redwood Shores</ps0:value>
        </ps0:localityName>

        <ps0:mail>
            <ps0:value>sample_person1@mycompany.com</ps0:value>
        </ps0:mail>
    <ps0:password>
    <ps0:value>V2VsY29tZTc=</ps0:value>
    </ps0:password>
        <ps0:organization>
            <ps0:value>1</ps0:value>
        </ps0:organization>

        <ps0:pager>
            <ps0:number>666-666-6666</ps0:number>
        </ps0:pager>

        <ps0:surname>
            <ps0:values>
                <ps0:value>sample_person1_ln</ps0:value>
            </ps0:values>
        </ps0:surname>

        <ps0:username>
            <ps0:value>sample_person1</ps0:value>
        </ps0:username>

    <ps0:activeEndDate>2020-12-31T16:00:00</ps0:activeEndDate>
    <ps0:activeStartDate>2000-01-10T18:00:00</ps0:activeStartDate>
    <ps0:hireDate>2000-01-01T18:00:00</ps0:hireDate>

        </ps0:identity>
    </data>
    <notificationData>
        <sendNotification>true</sendNotification>

    <sendNotificationTo><emailAddress>john.doe@mycompany.com, jane.doe@mycompany.com</e
    mailAddress></sendNotificationTo>
    </notificationData>
    </addRequest>
    </soap:Body>
    </soap:Envelope>

```

The response is as follows:

```

<env:Envelope
xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"><env:Header/><env:Body><ns3:
addResponse xmlns:ns2="http://xmlns.oracle.com/idm/identity/PSO"
xmlns:ns3="urn:oasis:names:tc:SPML:2:0"
xmlns:ns4="urn:oasis:names:tc:SPML:2:0:suspend"
xmlns:ns5="http://xmlns.oracle.com/idm/identity/spmlv2custom/Username"
xmlns:ns6="urn:oasis:names:tc:SPML:2:0:password"
xmlns:ns7="urn:oasis:names:tc:SPML:2:0:reference"
xmlns:ns8="urn:oasis:names:tc:SPML:2:0:async"
xmlns:ns9="urn:oasis:names:tc:SPML:2:0:batch"
xmlns:ns10="urn:names:spml:ws:header" requestID="1"
status="pending" /></env:Body></env:Envelope>

```

C.13 SPML Example – Revoke Role Membership

The Request is as follows:

```
<modifyRequest
xmlns="urn:oasis:names:tc:SPML:2:0"
xmlns:ps0="http://xmlns.oracle.com/idm/identity/PSO"
executionMode="asynchronous"
locale="en"
policyURI="revoke_role_01">
<ps0ID ID="identity:6C9B96E99FC8DC32E040E50A3D5252F5" />
<modification modificationMode="delete">
<capabilityData
      capabilityURI="urn:oasis:names:tc:SPML:2:0:reference"
mustUnderstand="true">
<reference
      xmlns="urn:oasis:names:tc:SPML:2:0:reference"
typeOfReference="memberOf">
<toPsoID ID="6C9B96E99FC8DC32E040E50A3D5252F5" />
</reference>
</capabilityData>
</modification>
</modifyRequest>
```

The Response is as follows:

```
<spml:modifyResponse
xmlns:spml="urn:oasis:names:tc:SPML:2:0"
status="pending"
requestID="10826"/>
```

C.14 SPML Example - Add Role

The Request is as follows:

```
<addRequest xmlns="urn:oasis:names:tc:SPML:2:0"
xmlns:ps0="http://xmlns.oracle.com/idm/identity/PSO" executionMode="asynchronous"
locale="en_us" policyURI="Role Creation" requestID="string"
returnData="identifier" targetID="string">
  <!--Zero or more repetitions:-->
  <capabilityData capabilityURI="urn:oasis:names:tc:SPML:2:0:reference"
mustUnderstand="true" />
  <data>
    <!--You have a CHOICE of the next 3 items at this level-->
    <ps0:role>
      <ps0:attributes>
        <ps0:attr name="Role Category Name">
          <!-- ps0:value>OIM Roles</ps0:value-->
          <ps0:value>Default</ps0:value>
        </ps0:attr>
      </ps0:attributes>
      <ps0:commonName>
        <!--1 or more repetitions:-->
        <ps0:values>
          <!--1 or more repetitions:-->
          <ps0:value>TempAdmin</ps0:value>
        </ps0:values>
      </ps0:commonName>
    </ps0:role>
  </data>
</addRequest>
```



```

    </pso:commonName>
    <pso:description>
      <!--1 or more repetitions:-->
      <pso:values>
        <!--1 or more repetitions:-->
        <pso:value>Temporary Administrator</pso:value>
      </pso:values>
    </pso:description>
    <pso:displayName>
      <!--pso:value locale="en">Alice Krug_en_US</pso:value-->
      <!--pso:value locale="fr">Alice Kru_fr</pso:value-->
      <pso:value locale="base">Alice Kru_base</pso:value>
    </pso:displayName>
  </pso:role>
</data>
</addRequest>

```

The Response is as follows:

```

<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
  <env:Header/>
  <env:Body>
    <ns3:addResponse xmlns:ns2="http://xmlns.oracle.com/idm/identity/PSO"
      xmlns:ns3="urn:oasis:names:tc:SPML:2:0"
      xmlns:ns4="urn:oasis:names:tc:SPML:2:0:suspend"
      xmlns:ns5="http://xmlns.oracle.com/idm/identity/spmlv2custom/Username"
      xmlns:ns6="urn:oasis:names:tc:SPML:2:0:password"
      xmlns:ns7="urn:oasis:names:tc:SPML:2:0:reference"
      xmlns:ns8="urn:oasis:names:tc:SPML:2:0:async" xmlns:ns9="urn:names:spml:ws:header"
      status="pending" requestID="21792"/>
  </env:Body>
</env:Envelope>

```

C.15 SPML Example - Add Role with Parent

The Request is as follows:

```

<addRequest xmlns="urn:oasis:names:tc:SPML:2:0"
  xmlns:pso="http://xmlns.oracle.com/idm/identity/PSO" executionMode="asynchronous"
  locale="en" policyURI="http://www.sample.com/string/string"
  requestID="string" returnData="identifier" targetID="string">
  <data>
    <!--You have a CHOICE of the next 3 items at this level-->
    <pso:role>
      <pso:commonName>
        <!--1 or more repetitions:-->
        <pso:values>
          <!--1 or more repetitions:-->
          <pso:value>TempAdmin</pso:value>
        </pso:values>
      </pso:commonName>
      <pso:description>
        <!--1 or more repetitions:-->
        <pso:values>
          <!--1 or more repetitions:-->
          <pso:value>Temporary Administrator</pso:value>
        </pso:values>
      </pso:description>
    </pso:role>
  </data>
</addRequest>

```

```

    </data>
    <capabilityData capabilityURI="urn:oasis:names:tc:SPML:2:0:reference"
mustUnderstand="true">
      <reference typeOfReference="inheritsFrom"
xmlns="urn:oasis:names:tc:SPML:2:0:reference">
        <toPsoID ID="6C9B96E99F77DC32E040E50A3D5252F5" />
      </reference>
    </capabilityData>
  </addRequest>

```

The Response is as follows:

```

<ns9:AddResponseType xmlns="http://xmlns.oracle.com/idm/identity/PSO"
xmlns:ns2="urn:oasis:names:tc:SPML:2:0"
xmlns:ns3="urn:oasis:names:tc:SPML:2:0:reference"
xmlns:ns4="urn:oasis:names:tc:SPML:2:0:password"
xmlns:ns5="urn:oasis:names:tc:SPML:2:0:suspend"
xmlns:ns6="http://xmlns.oracle.com/idm/identity/spmlv2custom/Username"
xmlns:ns7="urn:names:spml:ws:header" xmlns:ns8="urn:oasis:names:tc:SPML:2:0:async"
xmlns:ns9="oasis:names:tc:SPML:2:0" requestID="22" status="pending"/>

```

C.16 SPML Example - Modify Role

The Request is as follows:

```

<modifyRequest xmlns="urn:oasis:names:tc:SPML:2:0"
xmlns:ps0="http://xmlns.oracle.com/idm/identity/PSO" executionMode="asynchronous"
locale="string" policyURI="http://www.sample.com/string/string"
requestID="string" returnData="identifier">
  <capabilityData capabilityURI="urn:oasis:names:tc:SPML:2:0:reference"
mustUnderstand="true" />
  <ps0ID ID="role:6C9B96E99FC8DC32E040E50A3D5252F5" />
  <modification modificationMode="replace">
    <component path="/role" namespaceURI="http://www.w3.org/TR/xpath20" />
    <data>
      <ps0:role>
        <ps0:description>
          <!--1 or more repetitions:-->
          <ps0:values>
            <ps0:value>UK Updated Administrator</ps0:value>
          </ps0:values>
        </ps0:description>
      </ps0:role>
    </data>
  </modification>
</modifyRequest>

```

The Response is as follows:

```

<ns9:ModifyResponseType xmlns="http://xmlns.oracle.com/idm/identity/PSO"
xmlns:ns2="urn:oasis:names:tc:SPML:2:0"
xmlns:ns3="urn:oasis:names:tc:SPML:2:0:reference"
xmlns:ns4="urn:oasis:names:tc:SPML:2:0:password"
xmlns:ns5="urn:oasis:names:tc:SPML:2:0:suspend"
xmlns:ns6="http://xmlns.oracle.com/idm/identity/spmlv2custom/Username"
xmlns:ns7="urn:names:spml:ws:header" xmlns:ns8="urn:oasis:names:tc:SPML:2:0:async"
xmlns:ns9="oasis:names:tc:SPML:2:0" requestID="24" status="pending"/>

```

C.17 SPML Example - Add Parent to a Role

The Request is as follows:

```
<modifyRequest xmlns="urn:oasis:names:tc:SPML:2:0"
xmlns:ps0="http://xmlns.oracle.com/idm/identity/PSO" executionMode="asynchronous"
locale="string" policyURI="http://www.sample.com/string/string"
requestID="string" returnData="identifier">

  <ps0ID ID="role:26" targetID="target" />
  <modification modificationMode="modify">
    <component path="/role" namespaceURI="http://www.w3.org/TR/xpath20" />
    <data>
      <ps0:role>
        <ps0:description>
          <!--1 or more repetitions:-->
          <ps0:values>
            <!--1 or more repetitions:-->
            <ps0:value>UK Updated Administrator</ps0:value>
          </ps0:values>
        </ps0:description>
      </ps0:role>
    </data>

    <capabilityData capabilityURI="urn:oasis:names:tc:SPML:2:0:reference"
mustUnderstand="true">
      <reference typeOfReference="inheritsFrom"
xmlns="urn:oasis:names:tc:SPML:2:0:reference">
        <toPsoID ID="25" />
      </reference>
    </capabilityData>
  </modification>
</modifyRequest>
```

The Response is as follows:

```
<ns9:ModifyResponseType xmlns="http://xmlns.oracle.com/idm/identity/PSO"
xmlns:ns2="urn:oasis:names:tc:SPML:2:0"
xmlns:ns3="urn:oasis:names:tc:SPML:2:0:reference"
xmlns:ns4="urn:oasis:names:tc:SPML:2:0:password"
xmlns:ns5="urn:oasis:names:tc:SPML:2:0:suspend"
xmlns:ns6="http://xmlns.oracle.com/idm/identity/spmlv2custom/Username"
xmlns:ns7="urn:names:spml:ws:header" xmlns:ns8="urn:oasis:names:tc:SPML:2:0:async"
xmlns:ns9="oasis:names:tc:SPML:2:0" requestID="25" status="pending"/>
```

C.18 SPML Example - Role Grant

You cannot assign a role to multiple identities by using a SPML payload. If multiple identities are given, then the latest identity only is assigned with the role. You remove either of the identity from the payload.

The Request is as follows:

```
<modifyRequest xmlns="urn:oasis:names:tc:SPML:2:0"
xmlns:ps0="http://xmlns.oracle.com/idm/identity/PSO" executionMode="asynchronous"
locale="string" policyURI="http://www.sample.com/string/string"
requestID="string" returnData="identifier">
  <!--Zero or more repetitions:-->
  <capabilityData capabilityURI="urn:oasis:names:tc:SPML:2:0:reference"
mustUnderstand="true" />
```

```

<psoID ID="identity:6C9B96E99FC8DC32E040E50A3D5252F5" />
<psoID ID="identity:6C9B96E99FC8DC32E040E50A3D5252F5" />
<!--1 or more repetitions:-->
<modification modificationMode="add">
<capabilityData capabilityURI="urn:oasis:names:tc:SPML:2:0:reference"
mustUnderstand="true">
<reference xmlns="urn:oasis:names:tc:SPML:2:0:reference"
typeOfReference="memberOf">
<toPsoID ID="6C9B96E99FC8DC32E040E50A3D5252F5" />
</reference>
</capabilityData>
</modification>
</modifyRequest>

```

The Response is as follows:

```

<ns9:ResponseType xmlns="urn:oasis:names:tc:SPML:2:0"
xmlns:ns2="http://xmlns.oracle.com/idm/identity/PSO"
xmlns:ns3="urn:oasis:names:tc:SPML:2:0:password"
xmlns:ns4="urn:oasis:names:tc:SPML:2:0:suspend"
xmlns:ns5="http://xmlns.oracle.com/idm/identity/spmlv2custom/Username"
xmlns:ns6="urn:names:spml:ws:header"
xmlns:ns7="urn:oasis:names:tc:SPML:2:0:async"
xmlns:ns8="urn:oasis:names:tc:SPML:2:0:reference"
xmlns:ns9="oasis:names:tc:SPML:2:0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="ns4:ActiveResponseType" requestID="143" status="failure"/>

```

C.19 SPML Example - Delete Role

The Request is as follows:

```

<deleteRequest xmlns="urn:oasis:names:tc:SPML:2:0"
xmlns:pso="http://xmlns.oracle.com/idm/identity/PSO" executionMode="asynchronous"
locale="en" policyURI="http://www.sample.com/string/string"
requestID="string" returnData="identifier" targetID="string">
<psoID ID="role:6C9B96E99FC8DC32E040E50A3D5252F5" />
</deleteRequest>

```

The Response is as follows:

```

<ns9:ResponseType xmlns="http://xmlns.oracle.com/idm/identity/PSO"
xmlns:ns2="urn:oasis:names:tc:SPML:2:0"
xmlns:ns3="urn:oasis:names:tc:SPML:2:0:reference"
xmlns:ns4="urn:oasis:names:tc:SPML:2:0:password"
xmlns:ns5="urn:oasis:names:tc:SPML:2:0:suspend"
xmlns:ns6="http://xmlns.oracle.com/idm/identity/spmlv2custom/Username"
xmlns:ns7="urn:names:spml:ws:header" xmlns:ns8="urn:oasis:names:tc:SPML:2:0:async"
xmlns:ns9="oasis:names:tc:SPML:2:0" requestID="18" status="pending"/>

```

C.20 SPML Example - Status Request

The Request is as follows:

```

<statusRequest xmlns="urn:oasis:names:tc:SPML:2:0:async"
requestID="3456563"
asyncRequestID="75779"/>

```

The Response is as follows:

```
<statusResponse xmlns="urn:oasis:names:tc:SPML:2:0:async"
  requestID="3456563" status="success">
  <addResponse requestID="75779" status="pending" />
</statusResponse>
```

Another Request is as follows:

```
<statusRequest xmlns="urn:oasis:names:tc:SPML:2:0:async"
  requestID="12" asyncRequestID="1" returnResults="true" />
```

Here, returnResults=true. Therefore, the response will have all the attributes of the request.

The Response is as follows:

```
<ns9:StatusResponseType xmlns="urn:oasis:names:tc:SPML:2:0"
  xmlns:ns2="urn:oasis:names:tc:SPML:2:0:async"
  xmlns:ns3="http://xmlns.oracle.com/idm/identity/PSO"
  xmlns:ns4="urn:oasis:names:tc:SPML:2:0:reference"
  xmlns:ns5="urn:oasis:names:tc:SPML:2:0:password"
  xmlns:ns6="urn:oasis:names:tc:SPML:2:0:suspend"
  xmlns:ns7="http://xmlns.oracle.com/idm/identity/spmlv2custom/Username"
  xmlns:ns8="urn:names:spml:ws:header" xmlns:ns9="oasis:names:tc:SPML:2:0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:type="ns2:StatusResponseType" requestID="12" status="success">
  <ns2:addResponse requestID="14" status="success">
    <pso>
      <psoID targetID="Identity" />
      <data>
        <ns4:Identity xmlns:ns4="oasis:names:tc:SPML:2:0"
          xmlns:ns2="http://xmlns.oracle.com/idm/identity/PSO"
          xmlns:ns3="urn:oasis:names:tc:SPML:2:0:async"
          xmlns:ns5="urn:oasis:names:tc:SPML:2:0:async"
          xmlns:ns6="http://xmlns.oracle.com/idm/identity/spmlv2custom/Username"
          xmlns:ns7="urn:oasis:names:tc:SPML:2:0:suspend">
          <ns2:attributes>
            <ns2:attr xmlns=""
              xmlns:ns11="urn:oasis:names:tc:SPML:2:0" name="Organization">
              <ns2:value>1</ns2:value>
            </ns2:attr>
          </ns2:attributes>
          <ns2:activeEndDate>2009-12-25T16:00:00.000Z</ns2:activeEndDate>
          <ns2:activeStartDate>2009-12-24T18:00:00.000Z</ns2:activeStartDate>
          <ns2:commonName>
            <ns2:values>
              <ns2:value>Alice Krug</ns2:value>
            </ns2:values>
          </ns2:commonName>
          <ns2:countryName>Canada</ns2:countryName>
          <ns2:departmentNumber>
            <ns2:value>123</ns2:value>
          </ns2:departmentNumber>
          <ns2:description>
            <ns2:values>
              <ns2:value>Alice Krugs profile</ns2:value>
            </ns2:values>
          </ns2:description>
          <ns2:displayName>
```

```
<ns2:value>Alice Krug</ns2:value>
</ns2:displayName>
<ns2:employeeNumber>333</ns2:employeeNumber>
<ns2:employeeType>
  <ns2:values>
    <ns2:value>Full-Time</ns2:value>
  </ns2:values>
</ns2:employeeType>
<ns2:facsimileTelephoneNumber>
  <ns2:number>6506072253</ns2:number>
</ns2:facsimileTelephoneNumber>
<ns2:generationQualifier>
  <ns2:value>II</ns2:value>
</ns2:generationQualifier>
<ns2:givenName>
  <ns2:value>Alice</ns2:value>
</ns2:givenName>
<ns2:hireDate>1999-12-24T16:00:00.000Z</ns2:hireDate>
<ns2:homePhone>
  <ns2:number>8888888888</ns2:number>
</ns2:homePhone>
<ns2:homePostalAddress>
  <ns2:value>Baker street</ns2:value>
</ns2:homePostalAddress>
<ns2:initials>
  <ns2:value>J S</ns2:value>
</ns2:initials>
<ns2:localityName>
  <ns2:value>SFO</ns2:value>
</ns2:localityName>
<ns2:middleName>A</ns2:middleName>
<ns2:mobile>
  <ns2:number>4083485309</ns2:number>
</ns2:mobile>
<ns2:organization>
  <ns2:values>
    <ns2:value>1</ns2:value>
  </ns2:values>
</ns2:organization>
<ns2:organizationUnit>
  <ns2:values>
    <ns2:value>Sales</ns2:value>
  </ns2:values>
</ns2:organizationUnit>
<ns2:pager>
  <ns2:number>333</ns2:number>
</ns2:pager>
<ns2:postalAddress>
  <ns2:value>Baker street 222</ns2:value>
</ns2:postalAddress>
<ns2:postalCode>
  <ns2:value>4081</ns2:value>
</ns2:postalCode>
<ns2:postOfficeBox>
  <ns2:value>333n</ns2:value>
</ns2:postOfficeBox>
<ns2:preferredLanguage>en</ns2:preferredLanguage>
<ns2:state>
  <ns2:value>CA</ns2:value>
</ns2:state>
```

```

        <ns2:street>
            <ns2:value>Baker</ns2:value>
        </ns2:street>
        <ns2:surname>
            <ns2:values>
                <ns2:value>Krug</ns2:value>
            </ns2:values>
        </ns2:surname>
        <ns2:telephoneNumber>
            <ns2:number>6506072253</ns2:number>
        </ns2:telephoneNumber>
        <ns2:title>
            <ns2:values>
                <ns2:value>Mr</ns2:value>
            </ns2:values>
        </ns2:title>
        <ns2:username>
            <ns2:value>akrug3478</ns2:value>
        </ns2:username>
        <ns2:userType>End-User</ns2:userType>
    </ns4:Identity>
</data>
</ps>
</ns2:addResponse>
</ns9:StatusResponseType>

```

C.21 SPML Example - Reset Password

The request is:

```

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns1="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
<soap:Header>
    <ns1:Security>
        <ns1:UsernameToken>
            <ns1:Username>SYSTEM_ADMINISTRATOR_LOGIN</ns1:Username>
            <ns1:Password>SYSTEM_ADMINISTRATOR_PASSWORD</ns1:Password>
        </ns1:UsernameToken>
    </ns1:Security>
</soap:Header>
<soap:Body xmlns="urn:oasis:names:tc:SPML:2:0">
<resetPasswordRequest xmlns="urn:oasis:names:tc:SPML:2:0:password">

    executionMode="asynchronous"
    locale="en_US">
<psoID ID="BD7A621E8C7147D2E040E50AFC801934"></psoID>
</resetPasswordRequest>
    </soap:Body>
</soap:Envelope>

```

The response is as follows:

```

<env:Envelope
xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"><env:Header/><env:Body><ns6:
resetPasswordResponse xmlns:ns2="http://xmlns.oracle.com/idm/identity/PSO"
xmlns:ns3="urn:oasis:names:tc:SPML:2:0"
xmlns:ns4="urn:oasis:names:tc:SPML:2:0:suspend"
xmlns:ns5="http://xmlns.oracle.com/idm/identity/spmlv2custom/Username"

```

```

xmlns:ns6="urn:oasis:names:tc:SPML:2:0:password"
xmlns:ns7="urn:oasis:names:tc:SPML:2:0:reference"
xmlns:ns8="urn:oasis:names:tc:SPML:2:0:async"
xmlns:ns9="urn:oasis:names:tc:SPML:2:0:batch"
xmlns:ns10="urn:names:spml:ws:header"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="ns6:ResetPasswordResponseType"
status="success" /></env:Body></env:Envelope>

```

C.22 SPML Example - Reset Password with Notification

The request is:

```

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns1="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
<soap:Header>
  <ns1:Security>
    <ns1:UsernameToken>
      <ns1:Username>SYSTEM_ADMINISTRATOR_LOGIN</ns1:Username>
      <ns1:Password>SYSTEM_ADMINISTRATOR_PASSWORD</ns1:Password>
    </ns1:UsernameToken>
  </ns1:Security>
</soap:Header>
<soap:Body xmlns="urn:oasis:names:tc:SPML:2:0">
<resetPasswordRequest xmlns="urn:oasis:names:tc:SPML:2:0:password">
  executionMode="asynchronous"
  locale="en_US">
<psoID ID="BD7A621E8C7147D2E040E50AFC801934"></psoID>
<notificationData>
  <sendNotification>true</sendNotification>
<sendNotificationTo><emailAddress>john.doe@mycompany.com,jane.doe@mycompany.com,terrence.hill@mycompany.com</emailAddress></sendNotificationTo>
</notificationData>
</resetPasswordRequest>
</soap:Body>
</soap:Envelope>

```

The response is as follows:

```

<env:Envelope
xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"><env:Header /><env:Body><ns6:
resetPasswordResponse xmlns:ns2="http://xmlns.oracle.com/idm/identity/PSO"
xmlns:ns3="urn:oasis:names:tc:SPML:2:0"
xmlns:ns4="urn:oasis:names:tc:SPML:2:0:suspend"
xmlns:ns5="http://xmlns.oracle.com/idm/identity/spmlv2custom/Username"
xmlns:ns6="urn:oasis:names:tc:SPML:2:0:password"
xmlns:ns7="urn:oasis:names:tc:SPML:2:0:reference"
xmlns:ns8="urn:oasis:names:tc:SPML:2:0:async"
xmlns:ns9="urn:oasis:names:tc:SPML:2:0:batch"
xmlns:ns10="urn:names:spml:ws:header"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="ns6:ResetPasswordResponseType"
status="success" /></env:Body></env:Envelope>

```


C.23 SPML Example - Lookup User Name Policy

The request is:

```
<ns2:lookupUsernamePolicyRequest
xmlns:ns2="http://xmlns.oracle.com/idm/identity/spmlv2custom/Username"
requestID=" "
      executionMode="synchronous" locale="en" policyURI=" "
xmlns:ns3="urn:oasis:names:tc:SPML:2:0">
</ns2:lookupUsernamePolicyRequest>
```

The response is as follows:

```
<ns5:lookupUsernamePolicyResponse
xmlns:ns2="http://xmlns.oracle.com/idm/identity/PSO"
xmlns:ns3="urn:oasis:names:tc:SPML:2:0"
xmlns:ns4="urn:oasis:names:tc:SPML:2:0:suspend"
xmlns:ns5="http://xmlns.oracle.com/idm/identity/spmlv2custom/Username"
xmlns:ns6="urn:oasis:names:tc:SPML:2:0:password"
xmlns:ns7="urn:oasis:names:tc:SPML:2:0:reference"
xmlns:ns8="urn:oasis:names:tc:SPML:2:0:async"
xmlns:ns9="urn:oasis:names:tc:SPML:2:0:batch"
xmlns:ns10="urn:names:spml:ws:header" status="success"><ns5:description>Generates
user name based on email id if it is available, else if first name is present then
<&lt;first name&gt;& &lt;last name&gt;& @&lt;domain&gt;&, else &lt;last
name&gt;& @&lt;domain&gt;& </ns5:description></ns5:lookupUsernamePolicyResponse>
```

C.24 SPML Example - Cancel Request

The request is:

```
<ns1:cancelRequest xmlns:ns1="urn:oasis:names:tc:SPML:2:0:async"
asyncRequestID="162" />
```

The response is as follows:

A request that could be successfully withdrawn:

```
<ns8:cancelResponse xmlns:ns2="http://xmlns.oracle.com/idm/identity/PSO"
xmlns:ns3="urn:oasis:names:tc:SPML:2:0"
xmlns:ns4="urn:oasis:names:tc:SPML:2:0:suspend"
xmlns:ns5="http://xmlns.oracle.com/idm/identity/spmlv2custom/Username"
xmlns:ns6="urn:oasis:names:tc:SPML:2:0:password"
xmlns:ns7="urn:oasis:names:tc:SPML:2:0:reference"
xmlns:ns8="urn:oasis:names:tc:SPML:2:0:async"
xmlns:ns9="urn:oasis:names:tc:SPML:2:0:batch"
xmlns:ns10="urn:names:spml:ws:header" asyncRequestID="162" status="success" />
```

A request that could not successfully withdrawn:

```
<ns8:cancelResponse xmlns:ns2="http://xmlns.oracle.com/idm/identity/PSO"
xmlns:ns3="urn:oasis:names:tc:SPML:2:0"
xmlns:ns4="urn:oasis:names:tc:SPML:2:0:suspend"
xmlns:ns5="http://xmlns.oracle.com/idm/identity/spmlv2custom/Username"
xmlns:ns6="urn:oasis:names:tc:SPML:2:0:password"
xmlns:ns7="urn:oasis:names:tc:SPML:2:0:reference"
xmlns:ns8="urn:oasis:names:tc:SPML:2:0:async"
xmlns:ns9="urn:oasis:names:tc:SPML:2:0:batch"
xmlns:ns10="urn:names:spml:ws:header" asyncRequestID="161" status="failure"
error="malformedRequest" extendedError="IAM-3076087"><ns3:errorMessage>User cannot
withdraw specified request.</ns3:errorMessage></ns8:cancelResponse>
```

C.25 SPML Example - Batch Request

The request is as follows:

```
<urn1:batchRequest processing="parallel" onError="resume"
xmlns:urn1="urn:oasis:names:tc:SPML:2:0:batch"
xmlns:urn2="urn:oasis:names:tc:SPML:2:0"
xmlns:ps0="http://xmlns.oracle.com/idm/identity/PSO"
xmlns:urn3="urn:oasis:names:tc:SPML:2:0:password" >
  <!--Zero or more repetitions:-->
  <urn1:addRequest requestID="?" executionMode="asynchronous"
locale="en" policyURI="User Creation" targetID="?" returnData="identifier">
    <urn2:data>
      <!--You may enter ANY elements at this point-->
      <!--You have a CHOICE of the next 3 items at this level-->
      <ps0:identity>
        <ps0:commonName>
          <ps0:values locale="en">
            <ps0:value>John Smith</ps0:value>
          </ps0:values>
        </ps0:commonName>
        <ps0:countryName>Canada</ps0:countryName>
        <ps0:departmentNumber>
          <ps0:value>123</ps0:value>
        </ps0:departmentNumber>
        <ps0:description>
          <ps0:values>
            <ps0:value>John Smiths profile</ps0:value>
          </ps0:values>
        </ps0:description>
        <ps0:displayName>
          <ps0:value>John Smith</ps0:value>
        </ps0:displayName>
        <ps0:employeeNumber>333</ps0:employeeNumber>
        <ps0:employeeType>
          <ps0:values>
            <ps0:value>Full-Time</ps0:value>
          </ps0:values>
        </ps0:employeeType>
        <ps0:facsimileTelephoneNumber>
          <ps0:number>6506072253</ps0:number>
        </ps0:facsimileTelephoneNumber>
        <ps0:generationQualifier>
          <ps0:value>II</ps0:value>
        </ps0:generationQualifier>
        <ps0:givenName>
          <ps0:value>John</ps0:value>
        </ps0:givenName>
        <ps0:hireDate>1999-12-24T16:00:00</ps0:hireDate>
        <ps0:homePhone>
          <ps0:number>8888888888</ps0:number>
        </ps0:homePhone>
        <ps0:homePostalAddress>
          <ps0:value>Baker street</ps0:value>
        </ps0:homePostalAddress>
        <ps0:initials>
          <ps0:value>J S</ps0:value>
        </ps0:initials>
      </ps0:identity>
    </urn2:data>
  </urn1:addRequest>
</urn1:batchRequest>
```

```

<pso:jpegPhoto>
  <pso:value>c3RyaW5n</pso:value>
</pso:jpegPhoto>
<pso:localityName>
  <pso:value>SFO</pso:value>
</pso:localityName>
<pso:mail>
  <pso:value>jsmith@oracle.com</pso:value>
</pso:mail>
<pso:middleName>Park</pso:middleName>
<pso:mobile>
  <pso:number>4083485309</pso:number>
</pso:mobile>
<pso:organization>
  <pso:values locale="en">
    <pso:value>1</pso:value>
  </pso:values>
</pso:organization>
<pso:organizationUnit>
  <pso:values locale="en">
    <pso:value>Sales</pso:value>
  </pso:values>
</pso:organizationUnit>
<pso:pager>
  <pso:number>333</pso:number>
</pso:pager>
<pso:password>
  <pso:value>V2VsY29tZTE=</pso:value>
</pso:password>
<pso:postalAddress>
  <pso:value>Baker street 222</pso:value>
</pso:postalAddress>
<pso:postalCode>
  <pso:value>4081</pso:value>
</pso:postalCode>
<pso:postOfficeBox>
  <pso:value>333n</pso:value>
</pso:postOfficeBox>
<pso:preferredLanguage>en-US</pso:preferredLanguage>
<pso:state>
  <pso:value>CA</pso:value>
</pso:state>
<pso:street>
  <pso:value>Baker</pso:value>
</pso:street>
<pso:surname>
  <pso:values locale="en">
    <pso:value>Smith</pso:value>
  </pso:values>
</pso:surname>
<pso:telephoneNumber>
  <pso:number>6506072253</pso:number>
</pso:telephoneNumber>
<pso:title>
  <pso:values locale="en">
    <pso:value>Mr</pso:value>
  </pso:values>
</pso:title>
<pso:username>
  <pso:value>jsmith</pso:value>

```

```

        </pso:username>
    </pso:identity>
</urn2:data>
</urn1:addRequest>
<urn1:modifyRequest executionMode="asynchronous">
    <urn2:psoID ID="9924000" />
    <urn2:modification modificationMode="add">
        <urn2:component path="/identity"
namespaceURI="http://www.w3.org/TR/xpath20" />
        <urn2:data>
            <pso:identity>
                <pso:initials>
                    <pso:value>X Y</pso:value>
                </pso:initials>
            </pso:identity>
        </urn2:data>
    </urn2:modification>
    <urn2:modification modificationMode="replace">
        <urn2:component path="/identity"
namespaceURI="http://www.w3.org/TR/xpath20" />
        <urn2:data>
            <pso:identity>
                <pso:localityName>
                    <!--1 or more repetitions-->
                    <pso:value>new_locality</pso:value>
                </pso:localityName>
                <pso:homePhone>
                    <!--1 or more repetitions-->
                    <pso:number>0123456789</pso:number>
                </pso:homePhone>
            </pso:identity>
        </urn2:data>
    </urn2:modification>
    <urn2:modification modificationMode="delete">
        <urn2:component path="/identity"
namespaceURI="http://www.w3.org/TR/xpath20" />
        <urn2:data>
            <pso:identity>
                <pso:pager>
                    <pso:number>333</pso:number>
                </pso:pager>
            </pso:identity>
        </urn2:data>
    </urn2:modification>
</urn1:modifyRequest>
<urn1:deleteRequest executionMode="asynchronous" locale="en"
policyURI="http://www.sample.com/string/string" requestID="string"
returnData="identifier" targetID="string">
    <urn2:psoID ID="9924000" />
</urn1:deleteRequest>
<urn1:resetPasswordRequest executionMode="asynchronous">
<urn3:psoID ID="924000" />
<urn3:notificationData>
    <urn2:sendNotification>true</urn2:sendNotification>
    <urn2:sendNotificationTo>
        <urn2:emailAddress>john@oracle.com</urn2:emailAddress>
    </urn2:sendNotificationTo>
</urn3:notificationData>
</urn1:resetPasswordRequest>
</urn1:batchRequest>

```

The response is as follows:

```
<urn:batchResponse xmlns:urn="urn:oasis:names:tc:SPML:2:0:batch"
xmlns:urn1="urn:oasis:names:tc:SPML:2:0">
  <!--Zero or more repetitions:-->
  <urn:addResponse status="pending" requestID="1234"/>
  <urn:modifyResponse status="pending" requested="2345"/>
  <urn:deleteResponse status="pending" requestID="3456"/>
  <urn:resetPasswordResponse status="success" />
</urn:batchResponse>
```


A

access policies

Resource Administrator option, 11-3

account status reconciliation, 18-6

action field, 13-19

activeRequest, 32-7

adapter, 3-18, 3-19

mapping, 3-18

adapter mapping information, 3-18

Adapter Variables, 2-13

Adapters, 2-10

adapters

entity adapter variable mappings, 3-27

literals, 3-20

organization definition, 3-20

prepopulate adapter variable mappings, 3-28

process definition, 3-21

process task adapter variable mappings, 3-23

references, 3-20

rule generator adapter variable mappings, 3-27

task, 3-19

task assignment adapter variable mappings, 3-25

task mapping, 3-19

tasks, 3-19

user definition, 3-21

variable, 3-22

variable mapping, 3-22

variables, 3-19

add

custom ADF tags to Self Service, 29-1

custom SoD engine, 27-42

addRequest, 32-2

administrator groups

assigning, 11-3

updating permissions, 11-4

APIs, 31-1

commonly used services, 31-3

developing clients for Oracle Identity

Manager, 31-4

legacy APIs, 31-5

OIMclient, 31-1

Oracle Identity Manager services, 31-1, 31-2

reconciliation, 14-2

tcUtilityFactory, 31-2

approval levels, 23-26

operation-level approvals, 23-28

request-level approvals, 23-27

template-level approvals, 23-27

approval policies

creating, 23-29

approval process

developing, 24-4

approval workflows, 23-25

Archiving Directory parameter, 19-4

archiving directory, permissions on, 19-6

assigning and event handler or adapter, 11-64

Assignment windows, 1-13

asynchronous SoD validation process, 27-33

Attaching Pre-Populate Adapters, 3-9

Attaching Process Task Adapters to Process

Tasks, 3-14

Attaching Task Assignment Adapters to Process

Tasks, 3-5

attribute element, 23-12

AttributeReference element, 23-5

mandatory properties, 23-5

optional properties, 23-7

B

Batch Size parameter, 21-7

batched reconciliation, 18-7, 21-7

Bulk Load utility, 34-1

cleaning up, 34-30

creating a datafile, 34-5

creating a tablespace, 34-5

creating input source, 34-8

features, 34-1

fixing exceptions, 34-14

gathering performance data, 34-30

generating audit snapshot, 34-31

handling exceptions, 34-13

input parameters, 34-11

installing, 34-2

loading account data, 34-15

loading OIM User data, 34-6

loading role, role hierarchy, role membership, and
role category data, 34-22

monitoring progress, 34-12

options, 34-4

preparing your database, 34-5

- running, 34-6
- scripts, 34-3
- setting default password, 34-7
- temporary tables, 34-3
- verifying the outcome, 34-14

Business Rule Definition folder, 1-20

C

- callback
 - definition, 4-1
- callback service, 4-1
 - configuring, 4-8
 - event processing, 4-3
 - mapping attributes, 4-4
 - overview, 4-1
 - retries, 4-4
 - sending callbacks, 4-6
 - troubleshooting, 4-13
- CallbackConfiguration.xml, 4-8
- callbacks, 4-1
 - callback service, 4-1
 - event processing, 4-3
- child data, 23-15
- child data sets, 18-4, 18-5, 21-24, 21-28, 21-47, 22-10
- Close, 1-3
- Code, 13-19
- column header, 1-8
- column name, 15-9
- combo box, 1-11
- common request dataset, 23-20
- Concatenation Transformation Provider, 19-15
- configuration
 - callbacks, 4-8
- configure
 - custom request workflow, 23-1
 - plug-in, 7-3
 - reports, 36-6
 - scheduled task plugin.xml, 6-4
 - scheduled task XML file, 6-2
 - SOA composites
 - loading Oracle Identity Manager APIs, 26-1
 - SoD engine, 27-4
- connector objects, 18-9, 21-31, 21-38, 21-50
- containerID field, 21-19
- create, 12-7
 - approval policies, 23-29
 - process definition, 12-7
 - request dataset, 23-1
 - request templates, 23-30
 - scheduled task, 6-1
 - SOA composites, 25-2
 - transformation layer, 27-36
- Create a Remote Task, 2-21
- Create a Response, 2-36
- Create a Stored Procedure Task, 2-23
- Create a Utility Task, 2-25
- Create an Oracle Identity Manager API Task, 2-27
- create directory structure
 - scheduled task, 6-5

- create SOA composites, 23-24
- Create User dataset, 23-12
- Creating Adapter Tasks, 2-16
- creating IT resources, 11-45
- CSV files, 22-3
- CSV Reconciliation Format Provider, 19-7, 20-3, 20-16
- custom ADF tags, 29-1
- Custom Authentication Credentials Namespace
 - parameter, 19-9
- Custom Authentication Header Element
 - parameter, 19-10
- Custom Element to Store Password parameter, 19-10
- Custom Element to Store User Name
 - parameter, 19-10
- custom event handler
 - example
 - custom validation, 8-5
- custom event handlers, 8-4
 - creating plug-ins, 8-8
- custom events
 - defining, 8-8
- custom post-process event handler, 8-6
- custom pre-process event handler, 8-6
- custom providers, 18-8, 20-1, 21-50
- custom request workflow
 - configuring, 23-1
- custom SoD engine, 27-4
- customize
 - interface, 28-1
 - reconciliation operations, 14-1
- customizing
 - interface, 28-1
- customizing interface
 - Advanced Administration, 28-9
 - authenticated Self Service, 28-7
 - branding, 28-1
 - columns in tables, 28-19
 - custom URLs, 28-32
 - data customization, 28-32
 - disabling features, 28-18
 - Identity Administration, 28-4
 - menus and tabs, 28-15
 - popup properties, 28-33
 - renaming button labels, 28-12
 - style sheet modifications, 28-10
 - unauthenticated Self Service, 28-6
 - Workflow Designer, 28-34

D

- data field, 1-9
- Data Object Manager, 5-1
- data sets, 21-17
 - child, 18-4, 18-5, 21-24, 21-28, 21-47, 22-10
 - fields, adding or editing, 21-21
 - OIM, 21-17, 21-29, 21-49
 - OIM - Account, 22-8, 22-10
 - OIM - Account data set, 18-5
 - OIM - User, 22-9

- OIM - User data set, 18-5
- OIM Data Sets, 18-5
- Provisioning Staging, 21-19
- Provisioning Staging data sets, 18-5
- Reconciliation Staging, 18-4, 21-17, 22-9
- Source, 21-17
- Source data set, 18-4
- data type, 15-8
- data types, 30-1, 30-9
- DataSetValidator element, 23-4
- date, 1-10
- date formats, 18-8
- declare
 - plug-ins, 7-4
 - implicit declaration, 7-4
- Default Value, 15-9
- define
 - custom events, 8-8
 - plug-ins, 7-4
 - request approvals, 23-25
- define metadata
 - scheduled task, 6-2
- defining IT resources, 11-50
- Delete a Response, 2-37
- deleteRequest, 32-4
- Deleting Adapter Tasks, 2-35
- deleting IT resources, 11-48
- deploy
 - reports, 36-6
 - service components, 27-44
 - SOA composites, 25-3, 25-7, 26-6
 - testing the setup, 26-7
 - transformation layer, 27-36
- Description field, 13-19
- Design Console, 30-1
- Design Console Administration folder, 1-20
- design parameters, 21-6
- develop
 - approval process, 24-1, 24-4
 - clients for Oracle Identity Manager, 31-4
 - entitlements, 37-1
 - event handler, 8-1
 - LDAP synchronization operations, 9-1
 - plug-ins, 7-1, 7-3
 - scheduled task, 6-1
 - scheduled task class, 6-3
 - SOA composites, 25-1
- Development Tools folder, 1-20
- disable
 - SOA composites, 25-6
 - SoD, 27-9
- Disabling Adapters, 2-13

E

- Email Definition form, 12-1
- E-Mail Notification, 12-27
 - assign, 12-27
- enable
 - SOA composites, 25-7

- SoD, 27-8
- enable logging
 - SoD-related events, 27-57
- Encrypted field, 15-9
- entitlements, 37-1
 - available and assigned, 37-2
 - child process forms, 37-4
 - configuring scheduled tasks, 37-6
 - data capture process, 37-2
 - marking fields, 27-34
 - reports, 37-8
- Entity Adapters, 2-37
- event callbacks
 - sending, 4-6
- event handler
 - custom event handler
 - creating plug-ins, 8-8
 - custom validation, 8-5
 - developing, 8-1
 - developing LDAP synchronization
 - operations, 9-1
 - post-process, 8-6
 - pre-process, 8-6
- Event Handler Manager Form, 5-1
- event handlers, 11-64
 - custom, 8-4
 - extending user management operations, 8-2
- exception handling, 20-10
- Execution Schedule, 2-11
- extending
 - request management operations
 - prepopulating an attribute, 23-31
 - running custom code, 23-30
 - validating request data, 23-31

F

- failure threshold for stopping reconciliation, 18-8
- Field Size field, 15-8
- Field Type field, 15-8
- File Encoding parameter, 19-6
- File Prefix parameter, 19-4
- First, 1-3
- Fixed Column Width parameter, 19-5
- folders
 - Business Rule Definition, 1-20
 - Design Console Administration, 1-20
 - Development Tools, 1-20
 - Process Management, 1-19
 - Resource Management, 1-19
 - User Management, 1-18
- form names, 19-4, 21-29, 22-7
- full reconciliation, 18-7

G

- General tab, 12-15
- generate
 - Oracle Identity Management reports, 36-9
- generating reports

- against sample data source, 36-9
- against the BPEL-based JDBC data source, 36-10
- against the production JDBC data source, 36-9
- generic technology connector
 - connector objects, 18-9, 21-31, 21-38
- generic technology connector framework
 - features, 18-5
- generic technology connectors
 - account status reconciliation, 18-6
 - architecture, 18-3
 - batched reconciliation, 18-7
 - creating, 21-1
 - data sets
 - See* data sets
 - date formats, 18-8
 - exporting, 21-37
 - features, 18-5
 - full reconciliation, 18-7
 - functional architecture, 18-2
 - importing, 21-37, 22-7
 - incremental reconciliation, 18-7
 - managing, 21-35
 - mappings, purpose, 18-2
 - modifying, 21-36, 21-51
 - need for, 18-1
 - providers
 - See* providers
 - provisioning module, 18-4
 - reconciliation of multivalued attribute data
 - deletion, 18-7
 - troubleshooting, 22-1
 - configuration issues, 22-7
 - general issues, 22-1
 - trusted source reconciliation, 18-6

H

- Help URL, 13-19
- How a Process Task Adapter Works, 3-13

I

- ID Attribute for Child Dataset Holding Group
 - Membership Information parameter, 19-11
- ID field, 21-18, 21-21, 21-34
- identity connector framework
 - bundle JAR, 16-19
 - Java Connector Server, 16-22
 - .NET Connector Server, 16-25
 - server, 16-20
 - SSL, 16-25
- incremental reconciliation, 18-7
- Integration, 12-20
- integration
 - Oracle Identity Manager and Oracle SOA, 23-25, 24-1
- IT Resources Type Definition Form, 11-49
- IT resources, creating, 11-45
- IT resources, deleting, 11-48
- IT resources, managing, 11-46

- IT resources, modifying, 11-47
- IT resources, viewing, 11-47

J

- Java Connector Server, 16-22
- JDeveloper
 - modifying SOA composites, 25-6
 - setting application server connection, 26-2
 - setting up SOA composites, 26-2

K

- Key field, 13-19

L

- Label field, 15-8
- Last, 1-3
- LDAP synchronization operations
 - developing event handlers, 9-1
- listTargets, 32-5
- logging, 20-10
- Lookup Definition form, 15-1
- lookup fields, 1-10, 21-25
- Lookup Query, 30-11
- Lookup shortcut, 1-4
- lookupQuery element, 23-11
- lookupValues element, 23-10

M

- managing IT resources, 11-46
- mapped values
 - accessing, 7-9
- mapping
 - Identity Manager and SPML attributes, 4-4
- Mapping Rule Generator Adapter Variables, 3-1
- mappings, 18-8, 21-20, 21-22, 21-48
 - examples of, 21-21
 - limitations, 22-10
 - transformation mappings, 22-10
- mark fields as entitlements, 27-34
- MDS utilities, 33-1
 - examples, 33-5
 - properties file, 33-3
 - setting up environment, 33-1
 - weblogicDeleteMetadata, 33-1, 33-3
 - weblogicExportMetadata, 33-1, 33-2
 - weblogicImportMetadata, 33-1, 33-2
- menu items, for creating generic technology connectors, 21-3
- metadata, 21-15
- metadata definition
 - See* metadata detection
- metadata detection, 20-2, 21-11, 21-15, 21-37, 21-46, 22-12
- modify
 - approval workflow for SoD, 27-13
 - provisioning workflow for SoD, 27-30
 - registration XML file, 27-36

- registration XML file for SoD engine, 27-44
- SOA composites, 25-5
- SOA composites in JDeveloper, 25-6
- style sheet, 28-10
- Modify a Response, 2-36
- Modify an Adapter Variable, 2-15
- Modifying Adapter Tasks, 2-33
- modifying IT resources, 11-47
- modifying process tasks, 12-15
- modifyRequest, 32-3
- multilanguage support, 18-8, 20-13, 22-3
- multiple trusted source reconciliation, 11-73

N

- .NET Connector Server, 16-25
- Note field, 13-19
- Notes window, 1-11
- notification
 - definition, 4-1
- Notification tab, 12-26

O

- OAACG SIL Provider, 27-3
- objectClass field, 21-19
- OIA SIL Provider, 27-3
- OIM - Account data set, 18-5, 22-8, 22-10
- OIM - User data set, 18-5, 22-9
- OIM Data Sets, 18-5
- OIM data sets, 21-17, 21-29, 21-49
- OIM User, 18-9
- Oracle Application Access Controls Governor, 27-3, 27-4, 27-10, 27-35, 27-37, 27-39, 27-42, 27-43, 27-46
- Oracle e-Business Suite, 27-5, 27-35, 27-39, 27-42
- Oracle e-Business User Management, 27-2, 27-13, 27-31
- Oracle Identity Analytics, 27-7
- Oracle Identity Management reports, 36-1
- Oracle Identity Manager APIs, 31-1
- Oracle SOA
 - connecting, 24-5
 - developing approval process, 24-1
 - integration with Oracle Identity Manager, 23-25, 24-1
- Organization Provisioning, 30-2

P

- password fields, 21-47, 22-6
- Password Policies form, 15-1
- password-like fields, 21-47, 22-6
- permissions, for creating generic technology connectors, 21-3
- plug-in
 - configuring, 7-3
 - example plug-in.xml, 7-8
 - mapped values
 - simple mapped values, 7-8
 - Plug-in Framework, 7-1
 - Plug-in store

- database store, 7-2
- file store, 7-2
- Plug-in stores, 7-2
- plug-in metadata
 - specifying, 7-5
- plug-ins
 - access
 - mapped values, 7-9
 - custom event handlers, 8-8
 - declaring, 7-4
 - implicit declaration, 7-4
 - defining, 7-4
 - developing, 7-1, 7-3
 - directory structure, 7-5
 - registering, 7-6
 - using APIs, 7-6
 - using registration utility, 7-7
- preconfigured SoD connectors
 - Oracle e-Business User Management, 27-2, 27-13, 27-31
 - SAP User Management connector, 27-2, 27-13
- predefined
 - SOA composites, 24-3
- predefined providers
 - CSV Reconciliation Format Provider, 19-7, 20-3, 20-16
 - Shared Drive Reconciliation Transport Provider, 19-1, 20-16, 21-46, 21-49
 - SPML Provisioning Format Provider, 19-7, 20-3, 20-9, 20-17
 - Transformation Provider, 19-15
 - Validation Provider, 19-21
 - Web Services Provisioning Transport Provider, 19-12, 20-9, 20-17
- predefined request datasets, 23-2
- PrePopulationAdapter element, 23-9
- Process Definition form, 12-5
- process forms, 19-4, 21-29, 22-3, 22-7
- Process Management folder, 1-19
- providers
 - definition, 18-2
 - parameters, design, 21-6
 - parameters, run-time, 21-6
 - Provisioning Format Providers, 18-5, 20-3, 21-5
 - Provisioning Transport Providers, 18-5, 20-3, 21-5
 - Reconciliation Format Providers, 18-4, 20-2, 20-8, 21-4
 - Reconciliation Transport Providers, 18-3, 20-2, 20-8, 21-4
 - resource bundles, 20-13
 - reusing, 20-15
 - role, 20-1
 - selecting, 21-3
 - Transformation Provider, 18-4
 - Transformation Providers, 18-4, 21-22
 - Validation Providers, 18-4, 21-28
 - XML files, 20-10
 - See also* predefined providers
- Provision Resource dataset, 23-16
- Provisioning Format Providers, 18-5, 20-3, 21-5

- Provisioning Staging, 18-5
- Provisioning Staging data sets, 21-19
- Provisioning Transport Providers, 18-5, 20-3, 21-5
- Provisioning Workflow Definition, 11-9
 - event tabs, 11-9
 - tabs, 11-9
- PSO
 - regarding mapping, 4-4

R

- Reconcile Deletion of Multivalued Attribute Data
 - parameter, 21-9
- Reconciliation Format Providers, 18-4, 20-2, 20-8, 21-4
- reconciliation of multivalued attribute data
 - deletion, 18-7
- reconciliation operations
 - customizing, 14-1
- Reconciliation Staging data sets, 18-4, 21-17, 22-9
- Reconciliation Transport Providers, 18-3, 20-2, 20-8, 21-4
- Reconciliation Type parameter, 21-8
- register
 - new target system, 27-38
 - plug-ins, 7-6
 - using APIs, 7-6
 - using registration utility, 7-7
 - SIL provider, 27-46
 - SOA composites, 23-24, 25-4
- registration XML file
 - modifying, 27-36
- Remedy field, 13-19
- Remote Manager form, 15-1, 15-11
- removing an e-mail notification, 12-27
- Removing Process Task Adapters from Process Tasks, 3-17
- Removing Rule Generators from Form Fields, 3-4
- Removing Task Assignment Adapters from Process Tasks, 3-9
- renaming the JDBC-based JDBC data source, 36-10
- reports, 36-1
 - configuring, 36-6
 - deploying, 36-6
 - generating, 36-9
 - Oracle BI Publisher, 36-2
- request approval
 - approval levels, 23-26
 - approval workflows, 23-25
 - defining, 23-25
- request approvals
 - approval levels
 - operation-level approvals, 23-28
 - request-level approvals, 23-27
 - template-level approvals, 23-27
- request dataset, 23-1
 - child data, 23-15
 - common, 23-20
 - Create User, 23-12
 - creating, 23-1

- elements and properties, 23-3
 - attribute, 23-12
 - AttributeReference, 23-5
 - DataSetValidator, 23-4
 - lookupQuery, 23-11
 - lookupValues, 23-10
 - PrePopulationAdapter, 23-9
 - request-data-set, 23-3
 - mandatory properties
 - widget, 23-6
 - predefined, 23-2
 - Provision Resource, 23-16
 - sample, 23-12, 23-16
 - uploading to MDS, 23-24
- request management operations, 23-30
 - extending, 23-30
 - prepopulating an attribute, 23-31
 - running custom code, 23-30
 - validating request data, 23-31
- request templates
 - creating, 23-30
- request types and associated keys, 23-28
- request-data-set element, 23-3
- Reset Count field, 13-19
- Resource Administrator, 11-3
- resource bundles, 20-13
- Resource Management folder, 1-19
- Resource Objects form, 11-48
- Resources, 2-12
- resources
 - managing, 11-1
 - Organization Associated For a Resource option, 11-2
 - Resource Authorizers option, 11-4
 - Resource Workflows option, 11-4
 - Workflow Visualizer, 11-5
 - workflows, 11-4
- Responses, 2-13
- resumeRequest, 32-6
- reusing providers, 20-15
- row heading, 1-8
- Rule Designer form, 11-48
- rule elements, 30-1
- run-time parameters, 21-6

S

- sample XML
 - scheduled task, 6-3
- SAP CUA, 27-13, 27-35, 27-39, 27-42
- SAP GRC, 27-3, 27-6, 27-11, 27-35, 27-42
- SAP GRC SIL Provider, 27-3
- SAP R/3, 27-6, 27-39
- SAP User Management connector, 27-2, 27-13
- schdled task
 - developing schdled task class, 6-3
- scheduled task
 - configuring XML file, 6-2
 - creating, 6-1
 - creating directory structure, 6-5

- defining metadata, 6-2
- developing, 6-1
- sample XML file, 6-3
- scheduled task plugin.xml
 - configuring, 6-4
- scheduler
 - configuration file, A-1
- Scheduling Rule Generators, 2-37
- Sequence field, 15-9
- service account
 - management tasks, 30-14
- service accounts, 30-14
- service components
 - deploying, 27-44
- Severity field, 13-19
- Shared Drive Reconciliation Transport
 - Provider, 19-1, 20-16, 21-46, 21-49
- SIL, 27-2
- SIL provider, 27-4
 - registering, 27-46
- SIL providers, 27-3, 27-4, 27-37, 27-46
 - OAACG, 27-3
 - OIA, 27-3
 - SAP, 27-3
- SILConfig.xml, 27-38, 27-40, 27-41, 27-46
- SOA composites
 - application server connection, 26-2
 - creating, 23-24, 25-2
 - deploying, 25-3, 25-7, 26-6
 - testing the setup, 26-7
 - developing, 25-1
 - disabling, 25-6
 - enabling, 25-7
 - loading Oracle Identity Manager APIs, 26-1
 - configuring, 26-1
 - prerequisites, 26-1
 - modifying, 25-5
 - modifying in JDeveloper, 25-6
 - monitoring, 24-5
 - predefined, 24-3
 - registering, 23-24, 25-4
 - setting up in JDeveloper, 26-2
 - updating, 26-3
- SoD, 27-1
 - custom target system, 27-35
 - disabling, 27-9
 - enabling, 27-8
 - enabling logging, 27-57
 - modifying approval workflow, 27-13
 - modifying provisioning workflow, 27-30
 - provisioning workflow, 27-52
 - access policy-based provisioning, 27-56
 - direct provisioning, 27-52
 - request provisioning, 27-54
 - request provisioning with approver-only field, 27-56
 - request provisioning with DefaultSoDApproval, 27-55
 - request to modify provisioned workflow, 27-54
 - requesting for self, 27-56
 - updating entitlements, 27-53
 - updating entitlements by access policy-based provisioning, 27-57
 - SoD Invocation Library, 27-2
 - troubleshooting, 27-57
 - validation process, 27-1
- SoD check Web service, 27-12
- SoD engine
 - configuring, 27-4
 - custom, 27-42
 - modifying registration XML file, 27-44
- SoD engines
 - Oracle Application Access Controls
 - Governor, 27-3, 27-4, 27-10, 27-35, 27-37, 27-39, 27-42, 27-43, 27-46
 - Oracle Identity Analytics, 27-7
 - SAP GRC, 27-3, 27-6, 27-11, 27-35, 27-42
- SoD target systems
 - Oracle e-Business Suite, 27-5, 27-35, 27-39, 27-42
 - SAP CUA, 27-13, 27-35, 27-39, 27-42
 - SAP R/3, 27-6, 27-39
- Source, 18-4
- Source data sets, 21-17
- Source Date Format parameter, 21-9
- Specified Delimiter parameter, 19-5
- specify
 - plug-in metadata, 7-5
- SPML, 32-1
 - activeRequest, 32-7
 - addRequest, 32-2
 - attributes and LDAP mappings, B-1
 - deleteRequest, 32-4
 - examples, C-1
 - listTargets, 32-5
 - modifyRequest, 32-3
 - resumeRequest, 32-6
 - securing Web services, 32-10
 - statusRequest, 32-5
 - suggestUsername, 32-8
 - suspendRequest, 32-6
 - validateUsername, 32-7
 - XSD, 32-1
- SPML attributes, 4-4
- SPML operations, supported, 19-8
- SPML Provisioning Format Provider, 19-7, 20-3, 20-9, 20-17
- SPML Web Service Binding Style (DOCUMENT or RPC) parameter, 19-10
- SPML Web Service Complex Data Type
 - parameter, 19-10
- SPML Web Service Operation Name
 - parameter, 19-10
- SPML Web Service Soap Message Body Prefix
 - parameter, 19-11
- SPML Web Service Target Namespace
 - parameter, 19-11
- SSL
 - and identity connector framework, 16-25
 - SSL, configuring for SoD validation, 27-43

- SSL, configuring for Web services, 19-12
- Staging Directory (Multivalued identity data) parameter, 19-3
- Staging Directory (Parent identity data) parameter, 19-1
- staging directory, permissions on, 19-6
- statusRequest, 32-5
- Step 1 Provide Basic Information page, 18-9, 19-7, 21-3, 21-30, 21-45
- Step 2 Specify Parameter Values page, 21-6, 21-31, 21-45, 21-46
- Step 3 Modify Connector Configuration page, 19-4, 21-15, 21-31, 21-47, 22-3, 22-4
- Step 4 Verify Connector Form Names page, 19-4, 21-29, 21-31, 22-1
- Step 5 Verify Connector Information page, 21-30, 22-1
- Stop Reconciliation Threshold parameter, 21-7
- Stop Threshold Minimum Records parameter, 21-8
- suggestUsername, 32-8
- suspendRequest, 32-6

T

- Tab Delimiter parameter, 19-5
- Tabs of the Adapter Factory Form, 2-11
- Target Date Format parameter, 21-11
- Target ID parameter, 19-9
- target system
 - registering, 27-38
- Task Dependency tab, 12-22
- The Form Designer Form, 13-1
- Time, 1-10
- transformation layer
 - creating, 27-36
 - deploying, 27-36
- Transformation Providers, 18-4, 19-15
 - Concatenation Transformation Provider, 19-15
 - Translation Transformation Provider, 19-16
- Transformation providers, 21-22
- Translation Transformation Provider, 19-16
- troubleshooting
 - callbacks, 4-13
- trusted source reconciliation, 18-6, 21-4

U

- UDF, 22-6
- Unique Attribute (Parent Data) parameter, 19-5
- update
 - SOA composites, 26-3
- Usage Lookup, 2-12
- user account menu items, 21-3
- user account permissions, 21-3
- user account status reconciliation, 19-18, 21-22, 21-28
- User Defined Columns tab, 15-6
- User Defined Field Definition, 15-1
- User Defined Field Definition form, 15-5
- User Management folder, 1-18
- user management operations, 8-1

- extending with event handlers, 8-2
- stages, 8-1
- user modifiable metadata, 33-1
- user modifiable metadata files, 33-4
- User Name (authentication) parameter, 19-9
- User Password (authentication) parameter, 19-9
- utilities
 - Bulk Load, 34-1
 - Delete JAR, 35-3
 - Delete Resource Bundle, 35-4
 - Download JAR, 35-3
 - Download Resource Bundle, 35-4
 - MDS, 33-1
 - Upload JAR, 35-2
 - Upload Resource Bundle, 35-4

V

- validateUsername, 32-7
- Validation Providers, 18-4, 21-28
- Validation Providers, predefined, 19-21
- value objects, 20-9
- Variable List, 2-12
- variables, 30-1, 30-8
- viewing IT resources, 11-47

W

- Web Service SOAP Action parameter, 19-9
- Web Service URL parameter, 19-12
- Web Services Provisioning Transport Provider, 19-12, 20-9, 20-17
- Web services, configuring SSL for, 19-12
- widget, 23-6
- wildcard, 1-15
- Workflow Definition Renderer, 12-9
- Workflow Designer
 - customizing, 28-34
- Workflow Visualizer, 11-5
 - accessing task details, 11-14
 - elements, 11-5
 - expansion nodes, 11-13
 - opening, 11-5
 - Provisioning Workflow Definition, 11-9
- Working with Responses, 2-35
- WSSE Configured for SPML Web Service? parameter, 19-9

X

- XML files
 - generic technology connectors, 21-31
 - providers, 20-10