Oracle® Fusion Middleware

Developer's Guide for Oracle Identity Manager

11*g* Release 2 (11.1.2.2.0)

E27150-25

June 2015

ORACLE

Oracle Fusion Middleware Developer's Guide for Oracle Identity Manager, 11*g* Release 2 (11.1.2.2.0)

E27150-25

Primary Author: Debapriya Datta

Contributing Author: Anju Poovaiah

Contributor: Atul Goyal, Javed Beg, Rajesh Pakkath, Sanjay Rallapalli

Contents

3 Security Architecture

5 Developing Provisioning Processes

6 Developing Process Forms

7 Managing Lookup Definitions

Part III Connectors

8 Using the Adapter Factory

9 Understanding the Identity Connector Framework

10 Developing Identity Connectors Using Java

11 Developing Identity Connectors Using .NET

12 Integrating ICF with Oracle Identity Manager

13 Using Java APIs for ICF Integration

14 Configuring ICF Connectors

15 Understanding ICF Best Practices and FAQs

16 Understanding Generic Technology Connectors

17 Predefined Providers for Generic Technology Connectors

18 Creating Custom Providers for Generic Technology Connectors

19 Creating and Managing Generic Technology Connectors

20 Troubleshooting Generic Technology Connectors

Part IV Requests and Approval Processes

21 Developing Workflows for Approval and Manual Provisioning

22 Using Segregation of Duties (SoD)

Part V Data Synchronization

23 Customizing Reconciliation

24 Using the Bulk Load Utility

Part VIII Interfaces to Integrate With Other Applications

31 Using APIs

32 Using SPML Services

34 Developing Notification Events

35 Using the Callback Service

Part X Customization Lifecycle

36 Understanding Customization Types

37 Deploying and Undeploying Customizations

38 Migrating Configurations and Customizations

Part XI Reports and Audit

39 Configuring Reports

List of Examples

List of Figures

List of Tables

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 requests and approval workflows, developing connectors by using Identity Connector Framework, 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 that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Related Documents

For 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 and Access 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 Management*

- *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 Authorization Policy Manager Administrator's Guide*

Conventions

The following text conventions are used in this document:

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

Part I

Concepts

This part provides concepts related to Oracle Identity Manager architecture and security.

It contains the following chapters:

1

Product Overview

Oracle Identity Manager is an enterprise identity management system that manages user's access privileges in enterprise IT resources by controlling users, roles, accounts, and entitlements. It provides the functionalities for provisioning, identity and role administration, approval and request management, policy-based entitlement management, technology integration, and audit and compliance automation. Oracle Identity Manager is designed to administer intranet as well as extranet users, roles, and organizational access privileges across a company's resources throughout the entire identity management life cycle.

Oracle Identity Manager platform automates access rights management, security, and provisioning of IT resources. It connects users to resources, and revokes and restricts unauthorized access to protect sensitive corporate information.

This chapter contains the following sections:

1.1 Key Features and Benefits

Oracle Identity Manager architecture is flexible and scalable, and provides the following features:

- Audit and Compliance Management
- Integration Solutions
- User Provisioning

1.1.1 Ease of Deployment

Oracle Identity Manager provides a flexible Deployment Manager utility to assist in the migration of integration and configuration information between environments. The utility exports integration and configuration information as XML files. These files are then imported into the destination environment, which can be staging or production. You can use the XML files to archive configurations and maintain versions, as well as replicate integrations.

The Deployment Manager provides you with the flexibility to select what to import and export. It also helps you to identify data object dependencies during both import and export steps. This flexibility enables you to merge integration work done by multiple people and to ensure the integrity of any migration.

1.1.2 Simplified UI Customization

Oracle Identity Manager provides a browser-based customization framework that does not require writing codes or relying on developers. The interface is based on Application Development Framework (ADF), which ensures that all customizations are consistent and safe from upgrades and patches. The browser-based customization can be performed by using the Oracle Web Composer.

Oracle Identity Manager also provides business-friendly user personalization by enabling users to save and reuse frequently-searched items, configure columns in the search results table, and sort and filter data.

1.1.3 Simplified Configuration

Oracle Identity Manager abstracts and simplifies the configuration complexities through a business user-friendly extension framework for all entities. This allows users to extend the user, role, organization, catalog, and resource schemas by using the Form Designer. In addition, Web browser-based management of disconnected applications is allowed instead of cumbersome configuration in the Design Console. The SOA Tasklist embedded in Oracle Identity Manager simplifies disconnected application fulfillment, in which the provisioning of a disconnected resource is performed manually. To do so, SOA Tasklist leverages additional features of the SOA Tasklist, such as reassign or suspend manual application fulfillment.

1.1.4 Flexibility and Resilience

You can deploy Oracle Identity Manager in single or multiple server instances. Multiple server instances provide optimal configuration options, supporting geographically dispersed users and resources for increased flexibility, performance, and control. The Java 2 Enterprise Edition (J2EE) application server model of Oracle Identity Manager also provides scalability, fault tolerance, redundancy, failover, and system load balancing. As deployments grow, moving from a single server to a multiserver implementation is a seamless operation.

1.1.5 Maximum Reuse of Existing Infrastructure

To lower cost, minimize complexity, and leverage existing investments, Oracle Identity Manager is built on an open architecture. This allows Oracle Identity Manager to integrate with and leverage existing software and middleware already implemented within the IT infrastructure of an organization. For example, if an implementation requires integrating with an existing customer portal, then the advanced APIs of Oracle Identity Manager offer programmatic access to a comprehensive set of system functions. This allows IT staff to customize any part of its Oracle Identity Manager provisioning implementation to meet the specific needs of the organization.

1.1.6 Extensive User Management

Oracle Identity Manager enables you to define unlimited user organizational hierarchies and roles. It supports inheritance, customizable user ID policy management, password policy management, and user access policies that reflect customers' changing business needs. It also helps you to manage application parameters and entitlements, and to view a history of resource allocations. In addition, it provides delegated administration with comprehensive permission settings for user management.

Oracle Identity Manager contains a Web-based customizable Oracle Identity Manager Self Service that helps you extensively in user management.

1.1.7 Web-Based User Self-Service

Oracle Identity Manager contains a customizable Web-based, user self-service portal. This portal enables management of user information, self registration, changing passwords, resetting forgotten passwords, retrieving forgotten user login, requesting available applications, reviewing and editing available entitlements, and initiating or reacting to workflow tasks.

1.1.8 Modular and Scalable Architecture

Oracle Identity Manager is built on Java EE architecture. The J2EE application server model of Oracle Identity Manager provides scalability, fail over, load-balancing, and Web deployment. It is based on an open, standards-based technology and has a three-tier architecture (the client application, an Oracle Identity Manager supported J2EE-compliant Application Server, and an ANSI SQL-compliant database). Oracle Identity Manager can provision LDAP-enabled and non-LDAP-enabled applications.

Java EE is a standard, robust, scalable, and secure platform that forms the basis for many enterprise applications. Oracle Identity Manager runs on leading Java EE compliant application server platforms, including Oracle WebLogic, to take advantage of the performance and scalability features inherent in these servers. Java EE defines a set of standardized, modular components, provides a complete set of services to those components, and handles many details of the application behavior.

The application server, on which Oracle Identity Manager runs, provides the life-cycle management, security, deployment, and run-time services to the logical components that constitute the Oracle Identity Manager application. These services include:

- **Scalable management of resources through clustering and failover:** A cluster in Java EE architecture is defined as a group of two or more Java EE compliant Web or application servers that cooperate with each other through transparent object replication mechanisms to ensure that each server in the group presents the same content. Each server or node in the cluster is identical in configuration and acts as a single virtual server. Any Java EE server in the cluster can handle client requests

directed to this virtual server independently, which gives the impression of a single entity hosting the Java EE application in the cluster.

High availability refers to the capability to ensure that applications hosted in the middle tier remain consistently accessible and operational to the clients. This is achieved through the redundancy of multiple Web and application servers within the cluster, and is implemented by the failover mechanisms of the cluster. If an application component fails to process its task, then the cluster's failover mechanism reroutes the task and any supporting information to a copy of the object on another server to continue the task. Oracle Identity Manager supports a clustered environment. This includes ensuring that the EJBs and the Value Objects used to store data support serialization for the object replication to work.

- **Transaction management through load balancing:** Load balancing refers to the capability to optimally partition inbound client processing requests across all the Java EE servers that constitute a cluster based on certain factors, such as capacity, availability, response time, current load, historical performance, and administrative priorities placed on the clustered servers. A load balancer, which can be based on software or hardware, sits between the Internet and the physical server cluster, acting as a virtual server. When each client request arrives, the load balancer decides how the Java EE server satisfies that request.
- **Security management:** Oracle Identity Manager architecture relies on the application server for certain security services as part of its overall security infrastructure. In addition, Oracle Identity Manager leverages the Java EE security framework to provide a secure application environment. It also has a flexible permission model to provide control over the various functions within the application
- **Messaging:** The basic concept behind messaging is that distributed applications can communicate by using a self-contained package of business data and routing headers. These packages are called messages. While RMI and HTTP rely on a two-way active communication between a client and a server, messaging relies on two or more interested parties communicating asynchronously through a messaging server without waiting for a response. Java Messaging Service (JMS) is a wrapper API incorporated in the J2EE standard as a way to standardize messaging functionality. All standard application servers provide their own JMS server implementations as a part of their service offerings.

1.1.9 Based on Leading Software Development Standards

Oracle Identity Manager incorporates leading industry standards. For example, Oracle Identity Manager components are fully based on a J2EE architecture, so customers can run them from within their standard application server environments. Complete J2EE support results in performance and scalability benefits while aligning with existing customer environments to leverage in-house expertise.

Oracle develops its identity management products on a foundation of current and emerging standards. For example, Oracle is a Management Board member of Liberty Alliance, and incorporates Liberty Alliance developments in its solutions. Oracle participates in the Provisioning Services Technical Committee (PSTC), which operates under the auspices of the Organization for the Advancement of Structured Information Standards (OASIS).

1.1.10 Powerful and Flexible Process Engine

With Oracle Identity Manager, you can create business and provisioning process models in easy-to-use applications. Process models include support for approval

workflows and escalations. You can track the progress of each provisioning event, including the current status of the event and error code support. Oracle Identity Manager supports complex, branching, and nested processes with data interchange and dependencies. The process flow is fully customizable and does not require programming.

1.1.11 Built-In Change Management

Oracle Identity Manager enables you to package new processes, import and export existing ones, and move packages from one system to another.

1.1.12 Workflow and Policy

The use of workflow and policy to automate business and IT processes can lead to improved operational efficiency, enhanced security, and more cost-effective compliance tracking. Oracle Identity Manager provides the following features in this category.

Policy Management

Oracle Identity Manager enables policy-based automated provisioning of resources with fine-grained entitlements. For any set of users, administrators can specify access levels for each resource to be provisioned, granting each user only the exact level of access required to complete the job. These policies can be driven by user roles or attributes, enabling implementation of role-based access control as well as attribute-based access control. Effective blending of role-based and attribute-based policies is key to a scalable and manageable organization provisioning solution.

A request goes through multiple approvals before it is executed. When the request is submitted, it must acquire approvals at different levels. An approval in the system can be configured by using an approval policy. An approval policy defines the approval process to be invoked and the approval rules associated with the policy. These approval rules help the request engine to select the approval process. Business analysts can define approval policies and approval rules.

Workflow Management

Oracle Identity Manager supports the separation of approval and provisioning workflows. An *approval workflow* enables an organization to model its preferred approval processes for managing resource access requests. A *provisioning workflow* enables an organization to automate IT tasks for provisioning resources with the most complex of provisioning procedures.

The separation of these two workflows empowers business and IT process owners to manage work efficiently with minimum cross-process interferences. It also enables an organization to leverage existing workflows already deployed in systems such as a help desk and HRMS. Oracle Identity Manager provides the Workflow Visualizer that allows business users, administrators, and auditors to visualize task sequences and dependencies to understand process flow and the Workflow Designer to edit and manage the process flow.

Dynamic Error Handling

The error-handling capability of Oracle Identity Manager enables you to handle exceptions that occur during provisioning. Frequent problems, for example, absence of resources, do not stop the entire provisioning transaction or cause it to fail. Business logic defined within the provisioning workflow offers customized fail-safe capabilities within an Oracle Identity Manager implementation.

Transaction Integrity

Based on embedded state management capabilities, Oracle Identity Manager provides the high level of transaction integrity required by other mission-critical organization systems. Oracle Identity Manager features a state engine with rollback and recovery capabilities. When a provisioning transaction fails or is stopped, the system is able to recover and roll back to the last successful state or reroute to a different path, in accordance with predefined rules.

Real-Time Request Tracking

To maintain better control and provide improved visibility into all provisioning processes, Oracle Identity Manager enables users and administrators to track request status in real time, at any point during a provisioning transaction.

1.1.13 Audit and Compliance Management

Identity management forms a key component in any audit compliance solution of an organization. Oracle Identity Manager helps an organization to minimize risk and reduces the cost of meeting internal and external governance and security audits. This section discusses the features of Oracle Identity Manager that are listed in the audit and compliance management category.

Identity Reconciliation

Reconciliation is one of the significant capabilities of Oracle Identity Manager that enables it to monitor and track the creation, updation, and deletion of account across all managed resources. The process of reconciliation is performed by the reconciliation engine. If Oracle Identity Manager detects any accounts or changes to user access privileges are affected beyond its control, then the reconciliation engine can immediately take corrective action, such as undo the change or notify you. Oracle Identity Manager also helps you to detect and map existing accounts in target resources. This helps in the creation of an organization-wide identity and access profile for each employee, partner, or customer user.

Rogue and Orphan Account Management

A *rogue account* is an account created "out of process" or beyond the control of the provisioning system. An *orphan account* is an operational account without a valid owner. These accounts represent serious security risks to an organization. Oracle Identity Manager can monitor rogue and orphan accounts continuously. By combining denial access policies, workflows, and reconciliation, an organization can perform the required corrective actions when such accounts are discovered, in accordance with security and governance policies.

Service Accounts

Oracle Identity Manager can also manage the life cycle of special *service accounts*, also known as administrator accounts. These accounts have special life cycle requirements that extend beyond the life cycle of an assigned user and across the life cycles of multiple assigned users. Proper management of service accounts can help to eliminate another source of potential orphan accounts.

Comprehensive Reporting and Auditing

Oracle Identity Manager reports on both the history and the current state of the provisioning environment. Some of the identity data captured by Oracle Identity Manager includes user identity profile history, role membership history, user resource access, and fine-grained entitlement history. Oracle Identity Manager also captures

data generated by its workflow, policy, and reconciliation engines. By combining this data along with identity data, an organization has all the required data to address any identity and access-related audit inquiry.

Attestation

Attestation, also referred to as recertification, is a key part of Sarbanes-Oxley compliance and a highly recommended security best practice. Organizations meet these attestation requirements mostly through manual processes based on spreadsheet reports and e-mails. These manual processes tend to be fragmented, are difficult and expensive to manage, and have little data integrity and auditability.

Oracle Identity Manager offers an attestation feature that can be deployed quickly to enable an organization-wide attestation process that provides automated report generation, delivery, and notification. Attestation reviewers can review fine-grained access reports within an interactive user interface that supports fine-grained *certify*, *reject*, *decline*, and *delegate* actions. All report data and reviewer actions are captured for future auditing needs. Reviewer actions can optionally trigger corrective action by configuring the workflow engine of Oracle Identity Manager.

Note: The attestation feature has been deprecated in Oracle Identity Manager 11*g* Release 2 (11.1.2.2.0). Attestation has been replaced by identity certification. See the following sections for information about identity certification:

- "Managing Tasks" and "Using Identity Certification" in the *Oracle Fusion MIddleware User's Guide for Oracle Identity Manager*
- "Managing Identity Certification" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager*

However, the attestation feature is available if you upgrade from Oracle Identity Manager Release 9.x or 11g Release 1 (11.1.1) or 11g Release 2 (11.1.2).

Identity Certification

Identity certification enables Oracle Identity Manager to offer the following features in a single product:

- Access-request: Request privileges from a catalog, obtain approvals
- Fulfillment: Automated or manual provisioning
- Access-review: Certification of entities in Oracle Identity Manager

Identity certification also provides the following:

- Delegation of individual line-items within a certification allows a reviewer to spread the work among several people (who can work in parallel). This allows users who are responsible for reviewing access within an enterprise to spread the workload, and complete the work quickly.
- Two-phased review allows a single certification to combine the two key perspectives that inform access-review, which are:
  - The business reviewer, typically a Line-of-Business (LOB) manager
  - A technical reviewer, typically an IT expert or an application owner

1.1.14 Integration Solutions

A scalable and flexible integration architecture is critical for the successful deployment of organization provisioning solutions. Oracle Identity Manager offers a proven integration architecture and predefined connectors for fast and low-cost deployments.

Adapter Factory

Integrating most provisioning systems with managed resources is not easy. Connecting to proprietary systems might be difficult. The Adapter Factory eliminates the complexity associated with creating and maintaining these connections. The Adapter Factory provided by Oracle Identity Manager is a code-generation tool that enables you to create Java classes.

The Adapter Factory provides rapid integration with commercial or custom systems. Users can create or modify integrations by using the graphical user interface of the Adapter Factory, without programming or scripting. When connectors are created, Oracle Identity Manager repository maintains their definitions, creating self-documenting views. You use these views to extend, maintain, and upgrade connectors.

Predefined Connectors

Oracle Identity Manager offers an extensive library of predefined connectors for commercial applications and other identity-aware systems that are used widely. By using these connectors, an organization can get a head start on application integration. Each connector supports a wide range of identity management functions. These connectors use the most appropriate integration technology recommended for the target resource, whether it is proprietary or based on open standards. These connectors enable out-of-the-box integration between a set of heterogeneous target systems and Oracle Identity Manager. Because the connectors provide a set of components that were originally developed by using the Adapter Factory, you can further modify them with the Adapter Factory to enable the unique integration requirements of each organization.

Generic Technology Connectors

If you do not need the customization features of the Adapter Factory to create your custom connector, you can use the Generic Technology Connector (GTC) feature of Oracle Identity Manager to create the connector.

Identity Connectors

The Identity Connector Framework (ICF) decouples the connectors from Oracle Identity Manager. As a result, connectors can be used with any product. Identity connectors are designed to separate the implementation of an application from the dependencies of the system that the application is attempting to connect to.

1.1.15 User Provisioning

Provisioning provides outward flow of user information from Oracle Identity Manager to a target system. Provisioning is the process by which an action to create, modify, or delete user information in a resource is started from Oracle Identity Manager and passed into the resource. The provisioning system communicates with the resource and specifies changes to be made to the account.

Provisioning includes the following:

- Automated user identity and account provisioning: This manages user identities and accounts in multiple systems and applications. For example, when an

employee working in the payroll department is created in the human resources system, accounts are also automatically created for this user in the e-mail, telephone, accounting, and payroll reports systems.

- Workflow and policy management: This enables identity provisioning. Administrators can use interfaces provided by provisioning tools to create provisioning processes based on security policies.
- Reporting and auditing: This enables creating documentation of provisioning processes and their enforcement. This documentation is essential for audit, regulatory, and compliance purposes.
- Attestation: This enables administrators to confirm users' access rights on a periodic basis.
- Access deprovisioning: When the access for a user is no longer required or valid in an organization, Oracle Identity Manager revokes access on demand or automatically, as dictated by role or attribute-based access policies. This ensures that a user's access is promptly terminated where is it no longer required. This is done to minimize security risks and prevent paying for access to costly resources, such as data services.

1.2 System Requirements and Certification

Before deploying and using Oracle Identity Manager, you must ensure that your environment meets the minimum installation requirements.

The following URL contains information about supported installation types, platforms, operating systems, databases, JDKs, and third-party products for Oracle Fusion Middleware:

http://www.oracle.com/technetwork/middleware/ias/downloads/fusion-certification-100350.html

2

Product Architecture

The architecture of Oracle Identity Manager provides a number of compelling technical benefits for deploying a provisioning solution as part of identity and access management. Oracle Identity Manager has a flexible architecture that can handle IT and business requirements without requiring changes to existing infrastructure, policies, or procedures.

Oracle Identity Manager architecture is described in the following sections:

- How Oracle Identity Manager Works: The Tiers of Oracle Identity Manager
- System Components

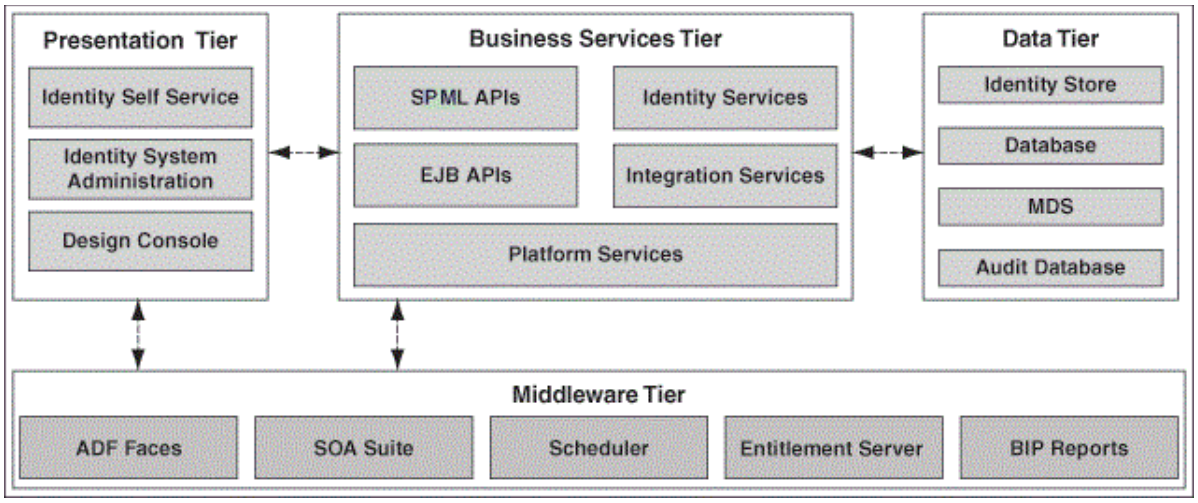
2.1 How Oracle Identity Manager Works: The Tiers of Oracle Identity Manager

Oracle Identity Manager is based on the n-tier J2EE application architecture. Oracle Identity Manager architecture contains the following tiers:

- Presentation Tier
- Business Services Tier
- Middleware Tier
- The Data Tier

Figure 2–1 illustrates Oracle Identity Manager architecture.

Figure 2–1 Oracle Identity Manager Architecture



2.1.1 Presentation Tier

The Presentation tier consists of three clients: Oracle Identity Self Service, Oracle Identity System Administration, and Oracle Identity Manager Design Console.

Oracle Identity Self Service is a Web-based thin client that can be accessed from any Web browser. This Web client provides user self-service and delegated administration features in a single interface that serve most of the users of Oracle Identity Manager.

Oracle Identity System Administration is a Web-based thin client that provides administrative and system management functions.

Oracle Identity Manager Design Console provides system configuration and development capabilities, including form, workflow design, and adapter creation and management. The Design Console is implemented as a Java Swing client that communicates directly with the Business Services layer in the application. You can access Oracle Identity Manager Design Console by using a desktop Java client.

Oracle Identity manager interfaces support a highly sophisticated delegated administration model, guaranteeing that users can only work on those parts of the application configuration for which they are authorized.

In many enterprises, there is a requirement for the provisioning system to support a custom developed client. Some of the requirements that drive this are:

- Integration of the client into an existing enterprise portal and adherence to enterprise portal standards
- Creation of custom flows for user interaction
- Creation of custom pages built around unique requirements from the provisioning system

To support customization, Oracle Identity Manager exposes the bulk of the necessary functionality via its published public APIs. The client environment for Oracle Identity Manager is customizable via Java APIs.

2.1.2 Business Services Tier

The Business Services Tier is implemented as an Enterprise JavaBeans (EJB) application. The core functionality for Oracle Identity Manager platform is

implemented in Java using a highly modular, object-oriented methodology. This makes Oracle Identity Manager flexible and extensible. The Business Services Tier for Oracle Identity Manager includes the following services and capabilities:

- The Core Services that comprise the core of the business features offered by Oracle Identity Manager, such as the User Management Service, the Policy Management Services, and the Provisioning and Reconciliation Services.
- The API Services that describe the APIs supported by Oracle Identity Manager that allow custom clients to integrate with Oracle Identity Manager. This includes a rich set of APIs that expose the business functionality of Oracle Identity Manager for use by custom clients, in product customization, and in plug-in and adapter development.
- The Integration Services based on the Adapter Factory and Connector Framework, which dynamically generates integration code based on the metadata definition of the adapters.
- The Platform Services that are crucial to the business features offered by Oracle Identity Manager, such as the Request Management Service, the Entity Manager Service, and the Scheduler Service.

The Business Services Tier is described in the following sections:

- The API Services
- Integration Services
- Platform Services

2.1.2.1 The API Services

The API Services describe the APIs supported by Oracle Identity Manager that allow custom clients to integrate with Oracle Identity Manager. This includes a rich set of APIs that expose the business functionality of Oracle Identity Manager for use by custom clients, in product customization, and in plug-in and adapter development.

The API Services consist of:

- **SPML APIs:** Service Provisioning Markup Language (SPML) is a standard for managing the provisioning and allocation of identity information and system resources within and between organizations. Oracle Identity Manager supports a set of SPML-based Web services that expose identity administration functionality to the clients. The APIs provide support for:
  - Adding, modifying, and deleting identities
  - Adding, modifying, and deleting roles
  - Adding and deleting role memberships

  These APIs support requests coming into Oracle Identity Manager for administration purposes, which is distinct and separate from SPML as the protocol used to integrate with provisioning targets.
- **EJB APIs:** Highly granular access to the functionality of the platform is via a set of EJB. These session beans are the basis for functionality implemented in Oracle Identity Manager Web application clients. It is also the interface that custom clients can use to access Oracle Identity Manager capabilities.

2.1.2.2 Integration Services

A scalable and flexible integration architecture is critical for the successful deployment of provisioning solutions. Oracle Identity Manager offers an integration architecture for fast and low-cost deployments.

Oracle Identity Manager integration services provide all the components required to support the development, deployment, and maintenance of connectors. The integration services includes:

- Connector Framework
- Identity Connectors
- Adapter Factory
- Generic Technology Connector
- Remote Manager

2.1.2.2.1 Connector Framework

Oracle Identity Manager connectors are packaged solutions that are used to integrate with target applications for the purposes of managing identities in those applications. Examples of such target applications are Microsoft Active Directory or Oracle E-Business Suite. A connector can be predefined by Oracle for particular target systems or can be custom developed.

Because a predefined connector is designed specifically for the target application, it offers the quickest integration method. These connectors support popular business applications such as Oracle eBusiness Suite, PeopleSoft, Siebel, JD Edward and SAP, as well as technology applications such as Active Directory, Java Directory Server, UNIX, databases, and RSA ClearTrust. Predefined connectors offer the quickest integration alternative because they are designed specifically for the target application. They use integration technologies recommended by target and are preconfigured with application specific attributes.

If predefined connectors does not use integration technologies recommended by target, then a custom connector can be developed. The Adapter Factory tool in Oracle Identity Manager Design Console provides a definitional user interface that facilitates such custom development efforts without coding or scripting.

A connector contains:

- Multiple connector-specific Oracle Identity Manager entities such as resource objects, data forms, provisioning workflows, and adapters
- Target-specific Java libraries that provide the underlying functions such as connectivity, authentication and user account management
- Event triggers that wire provisioning operations to both identity profile changes and policy operations

The connector framework combines all of these components together into a functional connector that is run at appropriate times, either manually based on user interaction or based on system triggering. It defines the various operational triggers, policy triggers, and hooks that allow the connector operation to be tailored to specific requirements.

2.1.2.2.2 Identity Connectors

Connectors are deployed with Oracle Identity Manager, which affects the portability of the connectors across various Oracle Identity Manager releases. The Identity Connector Framework (ICF) decouples the connectors from Oracle Identity Manager.

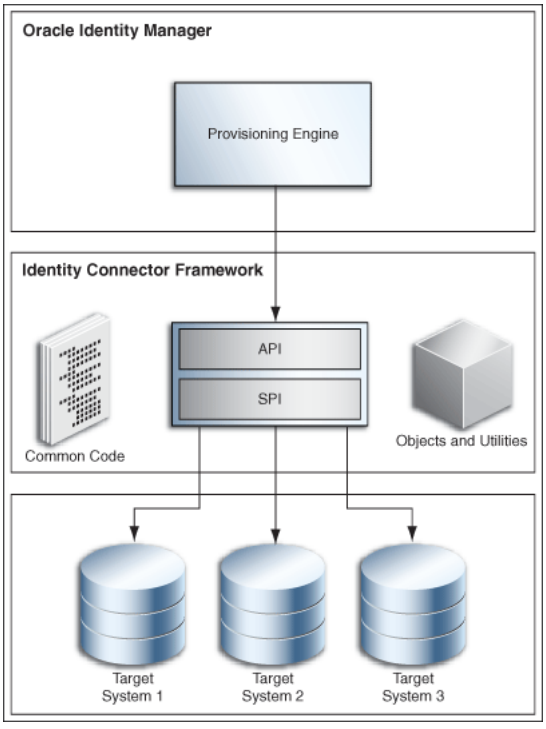
As a result, connectors can be used with any product. Identity connectors are designed to separate the implementation of an application from the dependencies of the system that the application is attempting to connect to.

Identity connectors have the following components:

- **The identity connector framework:** Provides a container that separates the connector bundle from the application. The framework provides many common features that developers would otherwise need to implement on their own. For example, the framework can provide connection pooling, buffering, timeouts, and filtering. The identity connector framework is separated into two parts:
  - The API: Applications use the API to call connectors
  - The SPI: Developers can create connectors by implementing the SPI
- **Identity connector bundle:** The specific implementation for a given resource target
- **The connector server (optional):** Allows an application to remotely run one or more connector bundles that are deployed on another system. Connector servers are available in both Java™ and .NET. The .NET connector server is needed only if you are using .NET connector bundles, whereas the Java connector server is available for connector bundles written in Java.

Figure 2–2 shows the ICF architecture.

Figure 2–2 ICF Architecture



Connector SPI

Connector SPI interfaces represent operations supported on a connector. A connector developer can choose to implement one or more operation interfaces for framing target system calls. Extension on existing interfaces or creating new interfaces is not supported. The SPI is broken up into required interfaces, feature-based interfaces, and operation interfaces such as create, update, delete, and search.

- The required interfaces include the org.identityconnectors.framework.spi.Connector interface and the org.identityconnectors.framework.spi.Configuration interface. These interfaces must be implemented in order for the API to understand which class contains the implementation of the configuration and which contains the implementation of the operations.
- The feature-based interfaces are the org.identityconnectors.framework.spi.AttributeNormalizer and org.identityconnectors.framework.spi.PoolableConnector interfaces.
- The operation interfaces determine the features that the connector supports such as create, delete, or search. See *Oracle Fusion Middleware Java API Reference for Oracle Identity Manager* for details.

Connector API

The connector API is responsible for presenting a consistent view of a connector regardless of the operations it has implemented. For the convenience of the SPI developer, there are several common features that are provided by default. For most of these features there is no need for the application developer to handle the APIs, only configure them. Following is a list of API features:

- Provide connection pooling to those connectors that require it and avoid the need for the API to see it, because not all connectors have connections. In addition, if the connector uses connection pooling, it is not the responsibility of the API developer to handle the connections, nor dispose of them during error conditions.
- Provide timeouts to all operations. The API consumer should only configure the appropriate timeout if the default is unacceptable. Each SPI developer should not have to implement such a common service and, for this reason, it is implemented in the framework.
- Provide search filtering by way of a simple interface that accepts a large variety of filters. The connector developer only needs to implement whichever filters the resource natively supports. The rest is handled by the framework.
- Provide search/sync buffering, allowing queries and updates to be handled in chunks if need be. The application need not worry about this, as it is handled within the framework.
- Provide scripting via Groovy and Boo .NET for connectors. This allows for great flexibility within a connector, because the framework can run scripts both on the connector and on the target resource (if supported).
- The SPI developer has the ability to choose different implementations of an operation. For instance there are two types of updates. This is hidden from the API consumer because there is no need for the application developer to call two different operations that essentially do the same thing. Instead the framework will figure out which operation the connector supports and make the appropriate calls.

2.1.2.2.3 Adapter Factory

The Adapter Factory is a code-generation tool provided by Oracle Identity Manager. It enables an Oracle Identity Manager application developer to create Java classes, known as adapters.

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 with the help of 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.
- It can be maintained easily because all the definitions for the adapter are stored in a repository. This repository can be edited through a GUI.
- A user in Oracle Identity Manager can retain the domain knowledge about the integration, while another user can maintain the adapter.
- It can be modified and upgraded.

The Adapter Factory provides rapid integration with commercial or custom systems. Users can create or modify integrations by using the graphical user interface of the Adapter Factory, without programming or scripting. When connectors are created, Oracle Identity Manager repository maintains the definitions and creates self-documenting views. You use these views to extend, maintain, and upgrade connectors.

2.1.2.2.4 Generic Technology Connector

Predefined Oracle Identity Manager connectors are designed for commonly used target systems such as Microsoft Active Directory and PeopleSoft Enterprise Applications. The architecture of a predefined connector 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.

The use of a predefined connector is the recommended integration method when such a connector is available for the target system. However, in some instances you might want to integrate Oracle Identity Manager with a target system that has no corresponding predefined connector. For example, XYZ Travels Inc. owns a custom Web-based application that its customers use to request airline fare quotes. Agents, who are also employees of XYZ Travels, respond to these requests by using the same application. Customers register themselves to create accounts in this application. However, XYZ 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. There is no predefined connector available to integrate the custom application with Oracle Identity Manager. Therefore, you must create the custom connectors that call the Java APIs exposed by the target application.

To integrate Oracle Identity Manager with a target system that has no corresponding predefined connector, you can create a custom connector to link the target system and Oracle Identity Manager. If you do not need the customization features of the Adapter Factory, then you can create the connector by using the Generic Technology Connector (GTC) feature of Oracle Identity Manager.

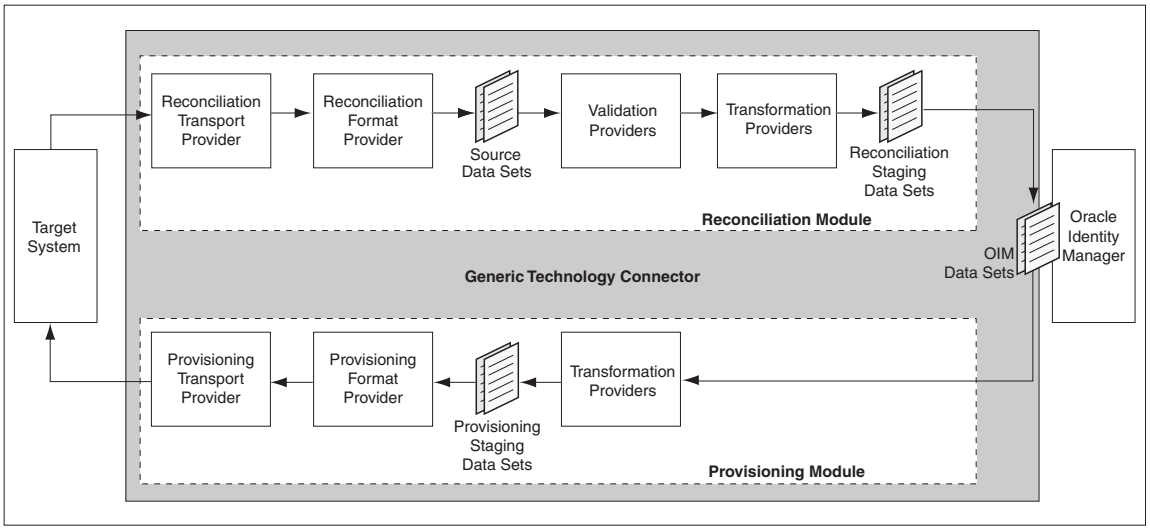
You can quickly and easily build a basic connector without advanced features and customized behavior by using generic connectivity technologies such as SPML and JDBC. GTC is a wizard that provides an alternative environment for connector development to rapidly create all the necessary functional components that make up a target system connector in Oracle Identity Manager.

The GTC framework provides basic components that are used to rapidly assemble a custom connector. The reconciliation and provisioning modules of a generic technology connector are composed of these reusable components that you select. Each component performs a specific function during provisioning or reconciliation. The components are:

- Reconciliation:
  - Reconciliation Transport Provider: This provider is responsible for moving the reconciled data from the target system to Oracle Identity Manager.
  - Reconciliation Format Provider: This provider parses the message received from the target system, which contains the reconciled data, into a data structure that can be interpreted by the reconciliation engine in Oracle identity Manager.
  - Validation Provider: This provider validates any data received before passing it on to the reconciliation engine.
- Provisioning:
  - Provisioning Format Provider: This provider converts Oracle identity Manager provisioning data into a format that is supported by the target system.
  - Provisioning Transport Provider: This provider carries the provisioning message received from the Provisioning Format Provider to the target system.

Figure 2–3 shows the functional architecture of a generic technology connector.

Figure 2–3 Functional Architecture of a Generic Technology Connector



Generic technology connectors have the following features:

- Features specific to the reconciliation module are:
  - **Generic technology connector in 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, then it creates corresponding OIM Users. If the reconciliation engine detects changes to existing target system accounts, then the same changes are made in the corresponding OIM Users.

  - **Generic technology connector in 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, then you can use the predefined Translation Transformation Provider to implement account status reconciliation.
  - **Generic technology connector in full or incremental reconciliation:** While creating a generic technology connector, you can specify that you want to use the connector for full or incremental reconciliation. In incremental reconciliation, only target system records that have changed after the last reconciliation run are reconciled (stored) into Oracle Identity Manager. In full reconciliation, all the reconciliation records are extracted from the target system.
  - **Generic technology connector for batched reconciliation:** To exercise more control over the reconciliation process, you can use the generic technology connector to 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.
  - **Generic technology connector in 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.
  - **Generic technology connector in 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.

- Other features of generic technology connectors are:
  - **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, then you can create custom providers.
  - **Multilanguage Support:** Generic technology connectors can handle both ASCII and non-ASCII user data.
  - **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 and the format in which date values must be sent to the target system during provisioning.
  - **Propagation of Changes in OIM User Attributes to Target Systems:** While creating a generic technology connector, you can enable the automatic propagation of changes in OIM User attributes to the target system.

2.1.2.2.5 Remote Manager

When your adapter uses Java tasks, you must configure Oracle Identity Manager to find the appropriate Java APIs. The Java APIs are located in JAR files in the Meta Data Store (MDS). 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:

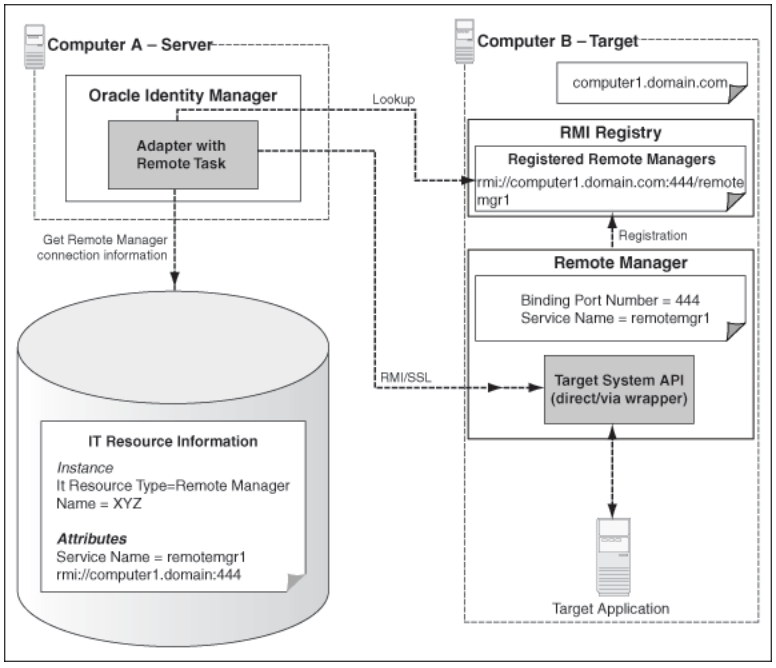
- Invoking nonremotable APIs through Oracle Identity Manager
- Invoking APIs that do not support Secure Sockets Layer (SSL) over secure connections

The Remote Manager is an Oracle Identity Manager server component that runs on a target system computer. It provides the network and security layer required to integrate with applications that do not have network-aware APIs or do not provide security. It is built as a lightweight Remote Method Invocation (RMI) server. The communication protocol is RMI tunneled through Hypertext Transfer Protocol/Secure (HTTP/S).

The J2EE RMI framework enables the creation of virtually transparent, distributed services and applications. RMI-based applications consist of Java objects making method calls to one another, regardless of their location. This enables one Java object to call methods on another Java object residing on another virtual computer in the same manner in which methods are called on a Java object residing on the same virtual computer.

Figure 2–4 shows an overview of the Remote Manager architecture.

Figure 2–4 Remote Manager Architecture



See Also: "Installing and Configuring a Remote Manager" for information about the Remote Manager and its configuration in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager*

2.1.2.3 Platform Services

The Platform Services include:

- Plug-In Framework
- SoD Engine Framework

2.1.2.3.1 Plug-In Framework

The Plug-in Framework allows customers to easily extend and customize the capabilities of the out-of-the-box Oracle Identity Manager features. The features expose specific plug-in points in the business logic where extensibility can be provided. An interface definition accompanies each such point and is called the plug-in interface. Customers can create code that extends these plug-in interfaces and defines customizations based on their business needs. These plug-ins are deployed and registered with Oracle Identity Manager by using the Plug-in Manager. Oracle Identity Manager then incorporates the plug-ins into the feature processing from that point onward.

Feature developers do not have to keep a track of where the custom implementations are stored and how they are loaded. The plug-in framework supports loading plug-ins from the classpath, from the file system, and from the database.

2.1.2.3.2 SoD Engine Framework

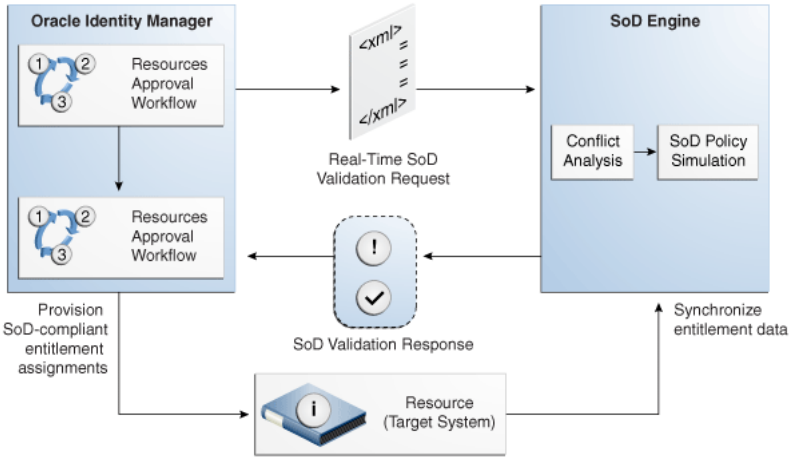
An attempt to enforce good compliance practices is through the definition of Segregation of Duties (SoD) policies. SoD is broadly defined as a way of preventing a user from acquiring a conflicting set of entitlements. This conflicting set is also referred to as a toxic combination. An example of a toxic combination is that a person should not have the ability to create and approve the same purchase order. Enterprises often have business application-specific SoD engines that define and enforce SoD policies on the entitlements users have within those business applications. Examples of such SoD engines are OAACG and SAP GRC.

The SoD Engine Framework allows customers to integrate Oracle Identity Manager with their choice of SoD Engine to enable SoD checks at appropriate points in the request and provisioning process. Oracle Identity Manager can send a request for an SoD check to the SoD Engine through the SoD Invocation Library (SIL). SIL provides a common service interface to all supported SoD engines. The common service interface provides an abstraction on the business components within Oracle Identity Manager. As a result, SoD checks do not have to take care of the correct data formats required by the SoD Engine and also the interpretation of the results returned.

SoD checks can be run at various times in the provisioning lifecycle, such as during an access request, during the approval workflow execution, or during the provisioning execution. If a violation is detected, then the request or resource is marked as being in violation, and the approver or administrator is responsible for deciding whether to proceed or not. If violations are detected during request processing, then various approval workflows can be invoked that allow for higher levels of approval.

Figure 2–5 shows the flow of data during the SoD validation process.

Figure 2–5 SoD Validation Process in Oracle Identity Manager



See Also: Chapter 22, "Using Segregation of Duties (SoD)" for detailed information about SoD

2.1.3 Middleware Tier

The Middleware Tier in Oracle Identity Manager architecture consists of the following:

- Request Service and Approval Workflow
- Authorization Service
- UI Customization Framework
- Scheduler Service
- Reporting

2.1.3.1 Request Service and Approval Workflow

Oracle Identity Manager architecture includes a request service that allows you to configure approval workflows. To deliver this functionality, Oracle Identity Manager uses Oracle Service Oriented Architecture (SOA) Suite.

Oracle SOA Suite enables you to build service-oriented applications and deploy them to your choice of middleware platform. It consists of a number of components, but for the purposes of delivering comprehensive workflow capabilities, Oracle Identity Manager relies on the following components:

- **BPEL Process Manager:** Oracle BPEL Process Manager provides a comprehensive solution for creating, deploying, and managing cross-application business processes with both automated and human workflow steps. It also provides audit trails for both completed and running processes, and process history that enables process improvement.
- **Human Request Service:** Although the BPEL standard does not cover manual tasks, it supports asynchronous services. Therefore, the Oracle SOA Suite supports the Human Request Service, which is a manual task service, so that manual steps can be included in standard BPEL processes. Oracle Identity Self Service includes a task list that allows users to view and interact with assigned tasks being managed within the Human Request Service.

- **BPEL Designer:** The Oracle BPEL Designer is available as a plug-in for JDeveloper and offers a visual design paradigm for creating and deploying BPEL-based processes.

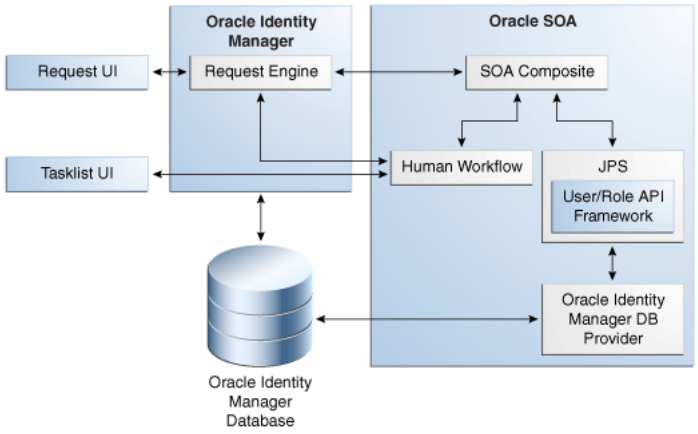
Oracle Identity Manager embeds SOA task list in the UI. This optimizes and simplifies the interaction of users with the SOA suite. The approvers can approve requests originating in Oracle Identity Manager by using the embedded SOA task list.

Embedded SOA task lists in Oracle Identity Manager enable making relevant data about the Oracle Identity Manager entities available to the SOA instance as Oracle Business Rules (OBR) facts on top of the default Web services. Therefore, writing complex JAVA code to resolve approver and approval routing in SOA workflows can be avoided. The following data is available in the SOA composites as OBR facts:

- User attribute for requestor and beneficiary
- All metadata associated with the base entity of requested item
- All metadata associated with the catalog entry of the requested item

Figure 2–6 shows the integration between the request service and SOA.

Figure 2–6 Request Service and SOA Integration



The request service also provides the services used to raise and track requests in Oracle Identity Manager. A request allows a user to ask that an action be taken after obtaining the necessary approvals, and that a tracking record of the entire process and its status be maintained. The request can be for various types of actions that are defined as request types. The request types can be:

- Creating, modifying, or deleting an entity
- Enabling or disabling an entity
- Adding or removing an identity as a member of a role
- Granting and revoking entitlements
- Provisioning, deprovisioning, enabling, disabling, and modifying application instances

> **Note:** Application instance is a new entity introduced in Oracle Identity Manager 11*g* Release 2 (11.1.2.2.0). For information about application instances, see "Managing Application Instances" in *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager*.

The request service supports various types of requests and has the ability to accommodate multiple request types. Oracle Identity Manager provides a number of predefined request types that cover the most common use cases.

The request service also provides support for heterogeneous requests that enable requesting for multiple types of entities in the same request. For example, assigning roles, provisioning application instances, and granting entitlements are supported in a single request.

The request service defines the flow models by which data provided in a request flows through the various services in Oracle Identity Manager. This includes invoking approval workflows at the correct time, monitoring the status of the workflows, and running the request if approval is received.

Both transaction data and history data for requests is maintained, which supports audit and compliance requirements.

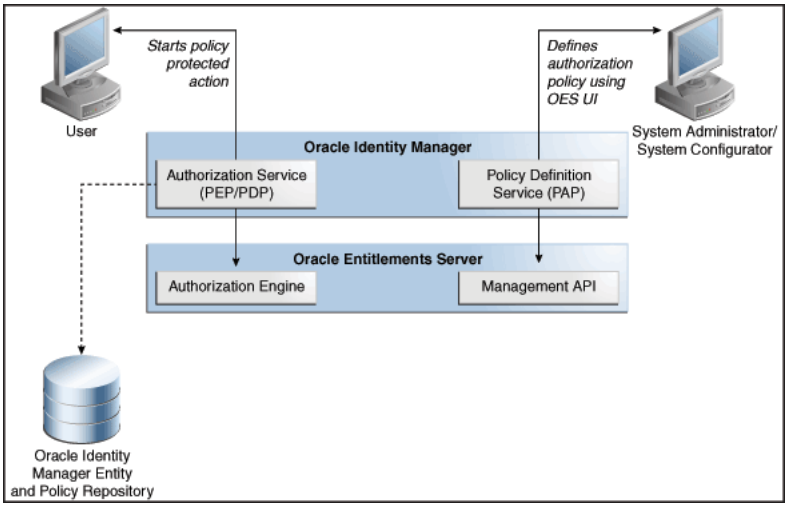
2.1.3.2 Authorization Service

Oracle Identity Manager requires a strong level of access control over what users can view and change in the application. To meet this requirement, Oracle Identity Manager lets you define authorization policies that determine at run time whether or not a particular action is allowed. This is controlled by the authorization service that uses Oracle Entitlements Server (OES) embedded within Oracle Identity Manager. OES is an authorization product and enables centralized management of entitlements and authorization policies to granularly determine access to both application components and application business objects.

The OES architecture is made up of two major components. The administration application acts as the policy administration point (PAP) and is used to manage policy, configuration, roles, and entitlements. The second major component is the use of one or more Security Modules (SMs) that are stored in the application container. The SMs evaluate fine-grain access control polices at the policy decision point (PDP) and enforce it at the policy enforcement point (PEP).

Figure 2–7 shows the architecture of OES-based authorization service:

Figure 2–7 OES-Based Authorization Service



Each time a privilege check is requested, the following takes place:

- Oracle Identity Manager connects to the authorization service to prepare access decision for the operations performed on protected entities.
- The service then finds and evaluates the policy or policies that apply to the resource.
- All information required to evaluate a policy is collected by the Security Modules at run time.
- If the policy references subject by role, all roles are evaluated and the access decision is made.

Oracle Identity Manager provides an abstraction service on top of OES that optimizes and simplifies the definition of policies in Oracle Identity Manager. This service includes a policy definition UI that allows the definition of authorization policies that are feature specific and support fine-grained controls for attributes and functions on entities such as users and roles.

2.1.3.3 UI Customization Framework

Oracle Identity Manager leverages Application Development Framework (ADF) to provide simple and business-friendly UI customization without the need to write code. The components used for customization are:

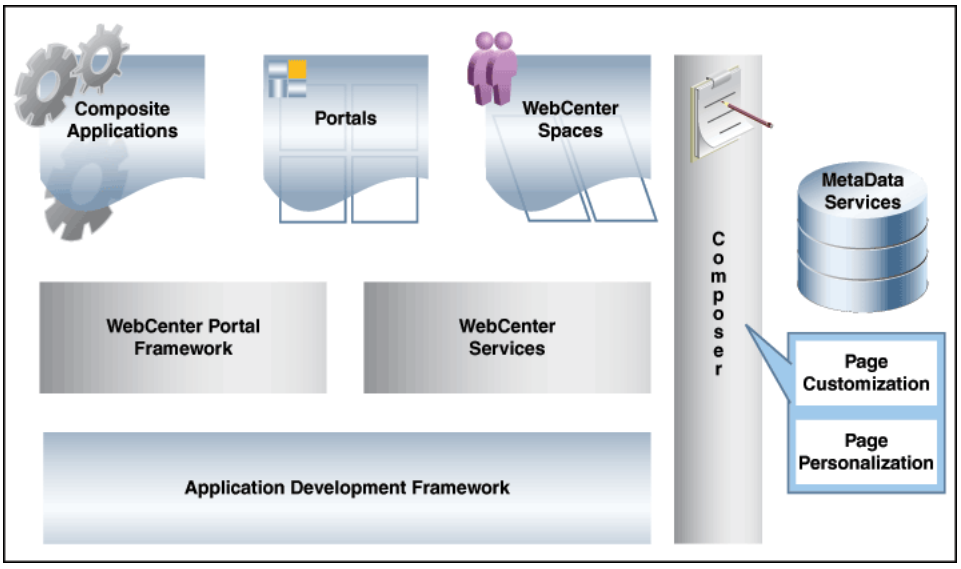
- **Oracle Web Composer:** Oracle Identity Manager supports Web browser-based UI customization and not IDE or JDeveloper-based customizations. JDeveloper is used to create pages, page fragments, regions, tabs and other UI artifacts. After these artifacts are deployed in Oracle Identity Manager, additional customizations can be done on this new content by using Oracle Web Composer.

  Note: If you want to use JDeveloper, then you can use APIs and build pages from scratch rather than customize predefined pages. There will be no interoperability between IDE and Web browser-based UI customizations.

  Oracle Web Composer enables you perform browser-based run-time customization or personalization of pages, Changing page layouts, adding items to pages, and performing branding. Customizations can be temporarily saved, reviewed, and then finalized as deployment specific, tenant specific, or user specific. This provides durable customizations across patches and upgrade because UI customizations are preserved separately from the code and UI metadata.
- **ADF Business Editor:** Using the ADF Business Editor, you can extend or add custom attributes to user, role, organization, catalog, and application instance entities. In addition, you can configure request datasets for users, resource objects, and application instances.
- **Meta Data Store (MDS):** Customizations and personalizations are stored in file system or database by using Oracle MDS.

Figure 2–8 illustrates the components of the UI customization framework.

Figure 2–8 UI Customization Framework



2.1.3.4 Scheduler Service

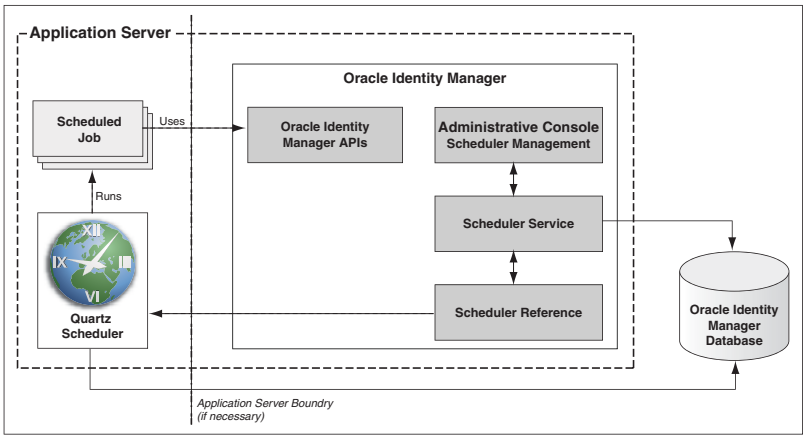
Business systems frequently make use of scheduling systems, which are configured to run other programs at specified times. Scheduling systems run applications that generate reports, reformat data, or perform audits at regular intervals of time. Scheduling systems often run batch jobs or scheduled jobs that perform routine work automatically at a prescribed time.

Scheduling systems are an integral part of any enterprise provisioning solution. Provisioning often involves tasks to be performed in a time-based manner. Some examples are:

- Running a nightly job to reconcile all changes made directly on a managed application
- Do escalations of assigned tasks that have not been handled within a specified time period
- Execute requests at a specific time

Oracle Identity Manager platform includes the Scheduler to provide the scheduling capabilities necessary for enterprise provisioning requirements. This Service is managed as part of Oracle Identity Manager platform and not as an independent product. Figure 2–9 provides an overview of Oracle Identity Manager Scheduler architecture.

Figure 2–9 Oracle Identity Manager Scheduler Architecture



Key capabilities provided by the Scheduler service are:

- The ability to create simple or complex schedules for running thousands of jobs
- The ability to run the scheduling service as a clustered service to provide the necessary high availability capabilities including fail-over and load balancing
- The ability to persist the job definitions for management and fail-over support
- The ability to create, modify, enable, disable, and delete jobs and manage individual job runs by using an administrative UI
- The ability to run a job in an ad-hoc fashion outside of regularly scheduled runs
- The ability to manage errors and failures
- The ability to maintain history of job runs, including statistics and results of these runs
- The ability to manage the Scheduler service itself

2.1.3.5 Reporting

The rich set of data stored in Oracle Identity Manager repository can be viewed through detailed reports that support management and compliance requirements. Oracle Identity Manager provides support for data reporting through the use of Oracle BI Publisher, which is an 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.

Oracle Identity Manager provides a set of standard Oracle BI Publisher report templates. However, you can customize each template to change its look and feel. in addition, you can create your own custom reports by leveraging Oracle Identity Manager database schema.

2.1.4 The Data Tier

Oracle Identity Manager is driven by data and metadata, which provides flexibility and adaptability to Oracle Identity Manager functionalities. Oracle Identity Manager data tier consists of Oracle Identity Manager repository or database, which manages and stores Oracle Identity Manager data and metadata in an ANSI SQL 92-compliant relational database, and an optional LDAP Identity Store.

This section describes the data tier in the following topics:

- Oracle Identity Manager Database
- The Metadata Store
- The Identity Store
- Integration Between LDAP Identity Store and Oracle Identity Manager

2.1.4.1 Oracle Identity Manager Database

Oracle Identity Manager repository is the authoritative store for the *Who Has What, When, How, and Why* data that is the core value of the identity administration and provisioning system. The data stored in Oracle Identity Manager database falls into the following broad categories:

- Entity Data: Users, organizations, roles, role memberships, resources, provisioned resources
- Transactional Data: Requests, approval and provisioning workflow instances, human tasks
- Audit Data: Request history, user profile history

High Availability

The database provides a scalable and redundant data layer to avoid downtime and performance issues. Reliability, recoverability, timely error detection, and continuous operations are primary characteristics of a highly available solution.

Oracle Identity Manager architecture relies on the corresponding capabilities provided by the Database Management System that is used with the product. These capabilities must:

- Encompass redundancy across all components
- Provide protection and tolerance from computer failures, storage failures, human errors, data corruption, lost writes, system hangs or slowdown, and site disasters
- Recover from outages as quickly and transparently as possible
- Provide solutions to eliminate or reduce planned downtime
- Provide consistent high performance
- Be easy to deploy, manage, and scale
- Achieve Service Level Agreements (SLAs) at the lowest possible total cost of ownership

A broad range of high availability and business continuity solutions are available. You can find out more about maximizing database availability by using technologies such as Oracle Real Application Clusters (Oracle RAC) and Oracle Data Guard at the following Web site:

`http://www.oracle.com/technetwork/database/features/availability/maa-090890.html`

2.1.4.2 The Metadata Store

The logic underlying Oracle Identity Manager is metadata driven. The structural and behavioral aspects are described by using metadata. Oracle Identity Manager architecture relies on Oracle Metadata Services (MDS) to provide a unified store for metadata. This ensures consistent and reliable access to the metadata for Oracle

Identity Manager and for the other Fusion Middleware components that it is built on. The same metadata that is used during the design phase of an application is used at application runtime through the metadata services layer. This ensures consistency through the lifecycle of Oracle Identity Manager. MDS also provides common administrative tooling for the metadata that can be used across various types of metadata stored in the common repository.

Key features and architectural principles of the MDS include:

- Simplified resource management through a single, unified repository for all artifacts used by various Fusion Middleware components
- Management of the metadata lifecycle for each artifact as it moves through the various stages of development, testing, staging, and production
- Sharing and reuse of metadata across components
- Categorization and reuse of artifacts, encouraging reuse, and promoting consistency
- Visioning capabilities, which form the basis for various features
- An upgrade-safe and layered customization mechanism through which metadata and application logic can be tailored per usage of the metadata
- Advanced caching and assembling techniques coupled with configurable tuning options to optimize performance

Metadata accessed and managed via MDS can be in a file-based repository or a database-based repository. In Oracle Identity Manager architecture, the metadata is in Oracle Identity Manager database to take advantage of some of the advanced performance and availability features that this mode provides.

MDS provides features using which you can create applications to meet customization requirements, such as modifying applications to suit the requirements of a specific business group, customizing applications to suit the individual preferences of a user, and creating applications that are customizable at run time. For more information about customizing Oracle Identity Manager by using MDS features, see "Customizing the Interface" on page 30-1.

2.1.4.3 The Identity Store

Oracle Identity Manager provides the ability to integrate an LDAP-based identity store into Oracle Identity Manager architecture. You can connect and manage an LDAP-based identity store directly from Oracle Identity Manager. Using this feature, you can use advanced user management capabilities of Oracle Identity Manager, including request-based creation and management of identities, to manage the identities within the corporate identity store.

In this deployment architecture, user identity information is stored in Oracle Identity Manager database to support the relational functionality necessary for Oracle Identity Manager to function, as well as in the LDAP store. All data is kept in sync transparently without the need for provisioning actions and setting up policies and rules. Identity operations started within Oracle Identity Manager, such as user creation or modification, are run on both the stores in a manner that maintains transactional integrity. In addition, any changes in the LDAP store made outside of Oracle Identity Manager is pulled into Oracle Identity Manager and made available as a part of the identity context.

2.1.4.4 Integration Between LDAP Identity Store and Oracle Identity Manager

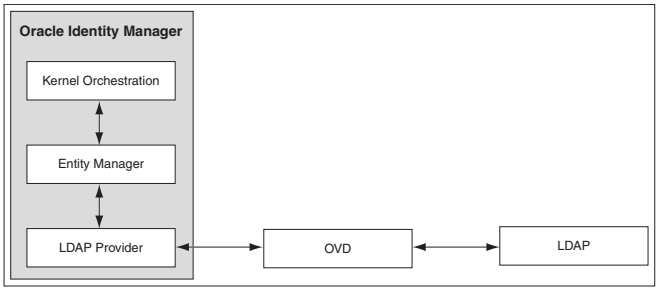
Oracle Identity Manager users and roles are stored in Oracle Identity Manager database. However, when a user, role, or role membership change takes place in Oracle Identity Manager, this information is propagated to LDAP identity store. If user, role, or role membership change takes place in LDAP directly, then these changes are synchronized into Oracle Identity Manager. The synchronization involves:

- User creation, modification, deletion, change in enable or disable states, and password change are made in LDAP in addition to the internal Oracle Identity Manager tables.
- Role creation, modification, and deletion actions update the LDAP groups, including membership changes.
- Initial load of users, roles, and role memberships are synchronized.
- Direct changes to user profile in LDAP are reconciled to Oracle Identity Manager. This does not include password changes.
- Direct changes to roles and role memberships in LDAP are reconciled to Oracle Identity Manager.

When changes are made in the user and role data, the actual operation is performed with the help of the kernel handlers. These handlers go through an orchestration lifecycle of various stages, such as validation, preprocessing, action, and postprocessing. For more information about the various stages of kernel orchestration, see "Developing Event Handlers" on page 28-1.

Oracle Identity Manager kernel orchestration connects to the Entity Manager, which in turn connects to the LDAP provider. The LDAP provider connects to Oracle Virtual Directory (OVD). The OVD is an interface to various directory systems, such as Oracle Internet Directory, iPlanet, and Active Directory. The LDAP provider reaches the LDAP data by using OVD. Figure 2–10 shows the communication between Oracle Identity Manager and LDAP.

Figure 2–10 Oracle Identity Manager and LDAP



The integration configuration and synchronization of data between Oracle Identity Manager and the LDAP identity store are described in the following sections:

- Configuring the Integration with LDAP
- Provisioning Data From Oracle Identity Manager to LDAP Identity Store
- Reconciliation From LDAP Identity Store to Oracle Identity Manager

2.1.4.4.1 Configuring the Integration with LDAP Configuring the integration between Oracle Identity Manager and LDAP is performed while installing Oracle Identity Manager. You can choose to install Oracle Identity Manager with or without LDAP. If you install Oracle Identity Manager with LDAP, then you must install OVD and

Oracle Internet Directory, create a container to store reserved users, create a new user in Oracle Identity Manager to perform Oracle Identity Manager operations, and configure OVD and Oracle Internet Directory for Oracle Identity Manager. For information about how to perform these configuration steps, see "Setting Up LDAP Synchronization" in the *Oracle Fusion Middleware Installation Guide for Oracle Identity and Access Management*.

After installing Oracle Identity Manager with LDAP enabled, you must open the following scheduled jobs and update their Last Change Number parameter with the last changelog number value of Oracle Internet Directory:

- LDAP User Create and Update Reconciliation
- LDAP User Delete Reconciliation
- LDAP Role Membership Reconciliation
- LDAP Role Hierarchy Reconciliation
- LDAP Role Create and Update Reconciliation
- LDAP Role Delete Reconciliation

In addition, you must enable these scheduled jobs after updating the Last Change Number parameter. To do so, see "Disabling and Enabling Jobs" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager*.

> **See Also:** "Managing Scheduled Tasks" for detailed information about scheduled jobs in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager*.

2.1.4.4.2 Provisioning Data From Oracle Identity Manager to LDAP Identity Store Oracle Identity Manager database stores the user and role information. When the user and role information is updated in Oracle Identity Manager, then the external repositories, such as the LDAP directory, must also be updated.

The LDAP changes are performed before Oracle Identity Manager changes. If Oracle Identity Manager changes fail, then the LDAP changes must be reverted to the original state. This is achieved by correcting an enable operation with a disable operation, a create operation with a delete operation, and a modification operation with another modification operation with the original values.

For instance, when a user is created, the validation processes are performed in the validation stage, such as password or any other policy validation. In the preprocessing stage, the user is created in LDAP first. Then, in the action stage, the user is to be created in Oracle Identity Manager. If there is an error in creating the user in Oracle Identity Manager, then the user must be deleted from LDAP because the corresponding user could not be created in Oracle Identity Manager. The operation to revert the change made is provided by the kernel handlers through the compensation method, which is predefined in Oracle Identity Manager.

> **Note:** Each handler has predefined execute and compensate methods. The execute method runs any operation, such as creating a user. The compensate method is called when an error occurs to revert the operation performed by the execute method.

To synchronize date from Oracle Identity Manager to LDAP, the location of the LDAP must be known to Oracle Identity Manager. The information about the LDAP location is stored in Oracle Identity Manager as the Directory Server IT resource. This is a

default IT resource provided by Oracle Identity Manager. The various parameters of this IT resource, which you can specify while installing Oracle Identity Manager, allows the connection between Oracle Identity Manager and LDAP.

In order to identify the same entry in Oracle Identity Manager and LDAP, the Distinguished Name (DN) and GUID attributes are used. Each entry has the DN attribute in LDAP, which indicates the unique location of an entry in LDAP. The GUID attribute is an unique ID to identify the entry. The DN and GUID for users and roles are stored in columns in the users and role tables in Oracle Identity Manager database. For information about how to synchronize user-defined fields between Oracle Identity Manager and LDAP, refer "Synchronizing User-Defined Fields Between Oracle Identity Manager and LDAP" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager*.

This section describes the following topics:

- Managing Users
- Managing Roles

2.1.4.4.3 Managing Users The following user operations can be performed to synchronize data from Oracle Identity Manager to LDAP:

- Create user
- Update user
- Delete user
- Enable user
- Disable user
- Lock user
- Unlock user
- Add role member
- Delete role member
- Change password

2.1.4.4.4 Managing Roles The following role operations can be performed to synchronize data from Oracle Identity Manager to LDAP:

- Create role
- Update role
- Delete role
- Add role to a member
- Add and Update role
- Remove role from a member
- Add role hierarchy
- Remove role hierarchy

2.1.4.4.5 Reconciliation From LDAP Identity Store to Oracle Identity Manager When changes in the identities are made directly in the LDAP identity store, the changes must be replicated to Oracle Identity Manager through authoritative source reconciliation. The identities include users and roles.

Reconciling users from LDAP to Oracle Identity Manager works with the general configuration of reconciliation, which includes the scheduled tasks for reconciliation.

> **See Also:** "Managing Scheduled Tasks" for information about scheduler and scheduled tasks in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager*.

> **Note:** Instead of using LDAP synchronization reconciliation jobs to reconcile users from LDAP to Oracle Identity Manager, if the Bulk Load utility is used, then subsequent operation on these users might fail if LDAP synchronization is enabled. To avoid this, all the users that are loaded in Oracle Identity Manager must be updated with correct GUID and DN values, and all these users in LDAP must be updated with an object class called orclIDXPerson.
>
> For detailed information about the Bulk Load utility, see "Using the Bulk Load Utility" on page 24-1.

The role reconciliation works only with the LDAP groups. Role reconciliation supports creation, updation, and deletion of roles. Role membership reconciliation supports creation and deletion of role memberships being driven from changes in an external LDAP directory.

Without roles and users being present in Oracle Identity Manager, role membership reconciliation will fail. Therefore, configure the LDAP synchronization scheduled jobs to run in the following order:

1. Fusion Applications Role Category Seeding

   > **Note:** Fusion Applications Role Category Seeding is a predefined scheduled task that is generated only when LDAP synchronization is enabled, along with other LDAP synchronization scheduled jobs. This job gets all distinct business categories in LDAP and creates them as OIM role categories.
   >
   > For a list of the predefined scheduled jobs, see "Predefined Scheduled Tasks" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager*.

2. LDAP Role Create and Update Reconciliation
3. LDAP Role Hierarchy Reconciliation
4. LDAP User Create and Update Reconciliation
5. LDAP Role Membership Reconciliation

For each of these jobs, except Fusion Applications Role Category Seeding, there is a parallel job to do the full reconciliation. All these jobs, except Fusion Applications Role Category Seeding, perform the reconciliation based on change logs, whereas full reconciliation jobs use the search base to do the reconciliation.

2.1.4.4.6 Consolidated LDAP Sync Full Reconciliation The LDAP Consolidated Full Reconciliation scheduled job runs the following jobs in order:

1. LDAP User Create and Update Full Reconciliation

2. LDAP Role Create and Update Full Reconciliation
3. LDAP Role Membership Full Reconciliation
4. LDAP Role Hierarchy Full Reconciliation

> **See Also:** "LDAP Scheduled Tasks" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager* for information about the LDAP Consolidated Full Reconciliation scheduled job

When you run the LDAP Consolidated Full Reconciliation scheduled job, the job status of the previous job and all event status for that particular job are checked because the next job must be run in a particular order. If any job fails to run, then the automatic run of the jobs stop, and error messages are logged in the diagnostic log.

> **Note:** The LDAP User Delete Full Reconciliation and LDAP Role Delete Full Reconciliation jobs are not part of LDAP Consolidated Full Reconciliation. These scheduled jobs are disabled by default. They can be enabled by selecting the radio buttons and can be run individually.

You can also run the individual jobs by selecting the radio buttons on the LDAP Sync Consolidated Full Reconciliation job details page. The job details contain all the common parameters for the four full reconciliation jobs. In addition, you can specify the values for the following parameters of the LDAP Sync Consolidated Full Reconciliation scheduled job:

- **Reconciliation Search Base:** Search base for the full reconciliation of users or roles. This defines the location in the LDAP directory from which the LDAP search begins.
- **Reconciliation Role Search Filter:** Search filter for full reconciliation of roles. This filter allows certain role/group entries in the subtree of the LDAP directory and excludes others.
- **Reconciliation User Search Filter:** Search filter for full reconciliation of users. This filter allows certain user entries in the subtree of the LDAP directory and excludes others.

Based on the values entered for the Reconciliation Search Base and/or Reconciliation User Search Filter and Reconciliation Role Search Filter parameters, the user and role accounts are pulled into Oracle Identity Manager from LDAP when the LDAP Sync Consolidated Full Reconciliation job is run. As a result of this full reconciliation, the delete happens in the Oracle Identity Manager database for the deleted entries in LDAP from that particular node.

The Reconciliation Search Base and Reconciliation Search Filter parameters support the following:

- **Reconciling the user or role account from LDAP to Oracle Identity Manager database:**

  This provides the option to perform fine-grained reconciliation of a particular user or role. The value of the Reconciliation Search Base parameter is:

  ```
  "cn=cokeuser1,cn=users,cn=subrealm1,dc=us,dc=oracle,dc=com"
  ```

- **All users and roles or groups under the node is reconciled:**

  The value of Reconciliation Search Base is:

```
"cn=tenant1,dc=us,dc=oracle,dc=com"
```

Here, the user full reconciliation and role full reconciliation are triggered. Therefore, all the users and roles or groups under the tenant1 node are reconciled.

- **All users under the node is reconciled:**

  The value of Reconciliation Search Base is:

  ```
  "cn=users,cn=tenant1,dc=us,dc=oracle,dc=com"
  ```

  Here, all the users under the tenant1 node are reconciled.

- **All roles or groups under the node is reconciled:**

  The value of the Reconciliation Search Base parameter is:

  ```
  "cn=groups,cn=tenant1,dc=us,dc=oracle,dc=com"
  ```

  Here, all roles or groups under the tenant1 node are reconciled.

The Reconciliation Search Base and Reconciliation Search Filter parameters are not bound together for LDAPSync Full reconciliation. Reconciliation Search Filter can be empty. Search Base can be used for provisioning or pushing entries from Oracle Identity Manager to LDAP, while Reconciliation Search Base can be used to perform full reconciliation from LDAP to Oracle Identity Manager database. If a value is not provided for Reconciliation Search Base, then the value for Search Base from the 'Directory Server' IT resource configuration is used for both provisioning and full reconciliation.

Sample values for the Reconciliation Search Base parameter:

```
"cn=tenant1,dc=us,dc=oracle,dc=com"
```

Sample values for the Search Base parameter:

```
"dc=us,dc=oracle,dc=com"
```

Sample values for the Reconciliation User Search Filter and Reconciliation Role Search Filter parameters:

```
(objectclass=orclAPPIDPerson)
(title=foobar)
```

Messages Logged For the LDAP Sync Consolidated Full Reconciliation Scheduled Job

The following is a list of messages for the LDAP Sync Consolidated Full Reconciliation scheduled job that are logged in the Oracle Identity Manager diagnostic log files:

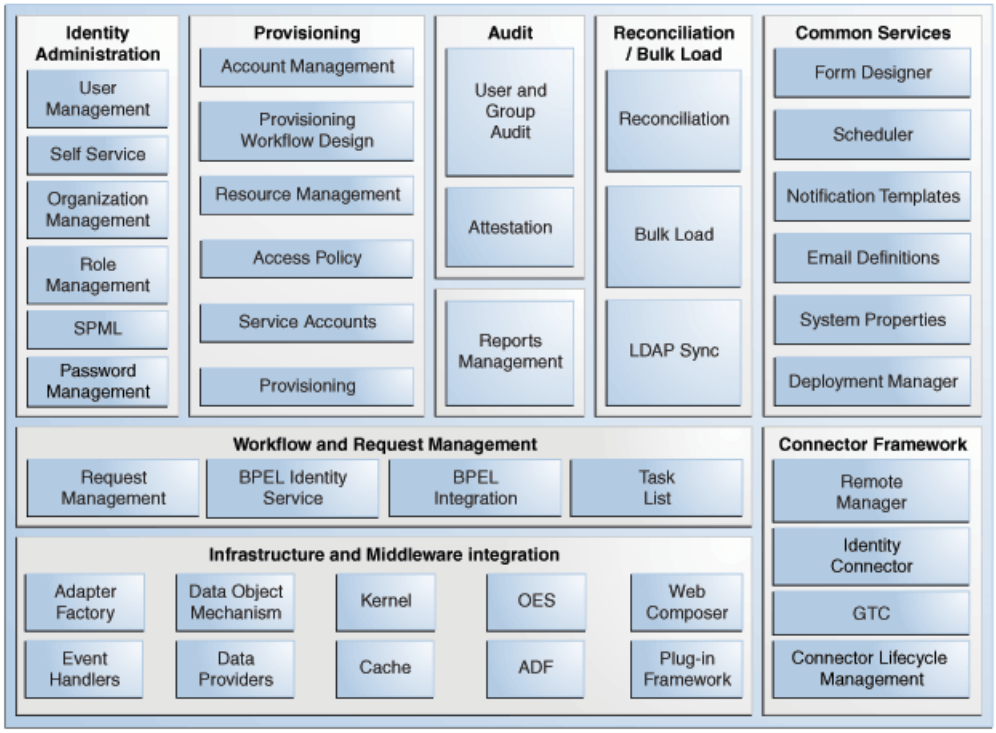
```
LDAP Sync Full Reconciliation Scheduler job {0} is currently Running.
LDAP Sync Full Reconciliation Scheduler job {0} is not currently Running. It has
Stopped.
LDAP Sync Full Reconciliation Scheduler job {0} is currently being Interrupted
while running.
LDAP Sync Full Reconciliation Scheduler job {0} is not currently Running. It has
Failed.
Error occurred while running the LDAP Sync User Full Reconciliation scheduler job.
Please refer to the OIM Server logs for more details.
LDAP Sync Full Reconciliation Scheduler job {0} is not currently Running. It has
been Shutdown.
LDAP Sync Full Reconciliation Scheduler job {0} is not currently Running.
SQLException has occurred.
```

```
All LDAPSync Full Reconciliation jobs ran successfully and Stopped.
```

2.2 System Components

Oracle Identity Manager is built on an enterprise-class, modular architecture that is both open and scalable. Each module plays a critical role in the overall functionality of the system. Figure 2–11 illustrates the system components of Oracle Identity Manager.

Figure 2–11 System Components of Oracle Identity Manager



Oracle Identity Manager user interfaces define and administer the provisioning environment. Oracle Identity Manager offers two user interfaces to satisfy both administrator and user requirements:

- Powerful Java-based Oracle Identity Manager Design Console for developers and system administrators

  Note: Only the users belonging to the SYSTEM ADMINISTRATORS group of Oracle Identity Manager can log in to Design Console.

- Web-based Administration and Oracle Identity Manager Self Service interfaces for identity administrators and users respectively

This section describes the following Oracle Identity Manager components:

- Identity Administration
- Provisioning
- Audit and Reports
- Reconciliation and Bulk Load

- Common Services
- Workflow and Request Management
- Infrastructure and Middleware Integration
- Connector Framework

Identity Administration

Identity administration includes creation and management of identities in Oracle Identity Manager. Identities include users, organizations, and roles. Identity administration also enables password management and user Oracle Identity Manager Self Service operations. Identity administration is performed by using Oracle Identity Manager Administration and Oracle Identity Manager Self Service Web clients, and the SPML Web service.

Provisioning

The provisioning transactions are assembled and modified in the provisioning module. This module maintains the "who" and "what" of provisioning. User profiles, access policies, and resources are defined in the provisioning module, as are business process workflows and business rules.

The Provisioning Server is the run-time engine for Oracle Identity Manager. It runs the provisioning process transactions as defined through Oracle Identity Manager Administration and Oracle Identity Manager Design Console and maintained within the provisioning module.

Audit and Reports

The audit and compliance functions include evaluating a person, organization, system, process, project, or product. This occurs by capturing data generated by the suite's workflow, policy, and reconciliation engines. By combining this data with identity data, an enterprise has all the information it requires to address any identity and to access a related audit inquiry. Audits are performed to ascertain the validity and reliability of information, and also provide an assessment of a system's internal control.

Reporting is the process of generating a formal document, which is created as a result of an audit. The report is subsequently provided to a user, such as an individual, a group of persons, a company, a government, or even the general public, as an assurance service so that the user can make decisions, based on the results of the audit. An enterprise can create reports on both the history and the current state of its provisioning environment. Some captured identity data includes user identity profile history, role membership history, user resource access, and fine-grained entitlement history.

Reconciliation and Bulk Load

The reconciliation engine ensures consistency between the provisioning environment of Oracle Identity Manager and Oracle Identity Manager managed resources within the organization. The reconciliation engine discovers illegal accounts created outside Oracle Identity Manager. The reconciliation engine also synchronizes business roles located inside and outside the provisioning system to ensure consistency.

If you want to load a large amount of data from other repositories in your organization into Oracle Identity Manager, then you can use the Bulk Load utility. The Bulk Load utility reduces the downtime in loading the data. In addition, Bulk Load utility import Oracle Identity Manager users, roles, role memberships, and accounts provisioned to users.

Common Services

Various services are grouped together that are shared and used by other Oracle Identity Manager components. These services are:

- **Form Designer:** A form that allows you to create process and resource object forms that do not come packaged with Oracle Identity Manager.
- **Scheduler:** A service that provides the capability to run specific jobs at specific schedules. This service can be used by users, application developers, connector developer, and administrators to create and configure a Job to be run at specified intervals. In addition, this service provides administrative capabilities to manage the functionality of jobs and their schedules.
- **Notification Templates:** A common notification service is used by other functional components to send notifications to interested parties about events occurring in Oracle Identity Manager. In addition, this service provides the administrative capabilities for notification template management. A notification template is used for sending the outgoing notifications. These templates typically contain the variables that refer to the available data to provide more contextual content.
- **System Properties:** A system property is an entity that controls the configuration aspect of an application. In addition, to the default system properties, you can create and manage system properties in Oracle Identity Manager.
- **Deployment Manager:** The Deployment Manager is a tool for exporting and importing Oracle Identity Manager configurations. The Deployment Manager enables you to export the objects that make up your Oracle Identity Manager configuration.

Workflow and Request Management

Various operations in Oracle Identity Manager cannot be performed directly. Instead, the operations must be requested. The request management service provides a mechanism to create, approve, and manage requests. A request is an entity created by the users or administrators who want to perform a specific action, which requires a discretionary permission to be obtained from someone or some process before the action can be performed. For example, a user can create a request to gain access to a laptop computer, a manager can approve the request and create an open requisition, and an IT resource administrator can approve the request.

The primary goal of a provisioning solution is to manage requests and provision resources. Request service provides an abstraction layer on the Business Process Execution Language (BPEL) 11*g* workflow engine. Functional components such as request, provisioning, and certification interacts with the workflow engine for human approvals. Request service caters to the various functional components in Oracle Identity Manager by managing workflow instances and categories, and provides an abstraction layer on BPEL.

Infrastructure and Middleware Integration

The Adapter Factory, Kernel Orchestration mechanism, Context Manager, and Plug-in Framework are designed to eliminate the need for hard-coding integrations with these systems.

Connector Framework

The integration solution strategy of Oracle Identity Manager provides connectors to various heterogeneous identity-aware IT systems. This strategy is designed to minimize custom development, maximize the reuse of code, and reduce deployment time. The tiers of the integration solution are:

- Out-of-the box integration using predefined connectors and predefined generic technology connector providers
- Identity connectors that are designed to separate the implementation of an application from the dependencies of the system that the application is attempting to connect to
- Connectors based on custom generic technology connector providers
- Custom connectors using the Adapter Factory

3

Security Architecture

Oracle Identity Manager controls access to the application by the users to allow or prevent the users to perform various operations in the application. This is controlled by the authorization engine embedded in Oracle Identity Manager with the help of authorization policies. The purpose of authorization policies is to control user's access to Oracle Identity Manager application, which includes data, UI, and API. The authorization policies determine at runtime whether or not a particular action is allowed. Authorization policies can be defined that satisfy the authorization requirements within Oracle Identity Manager.

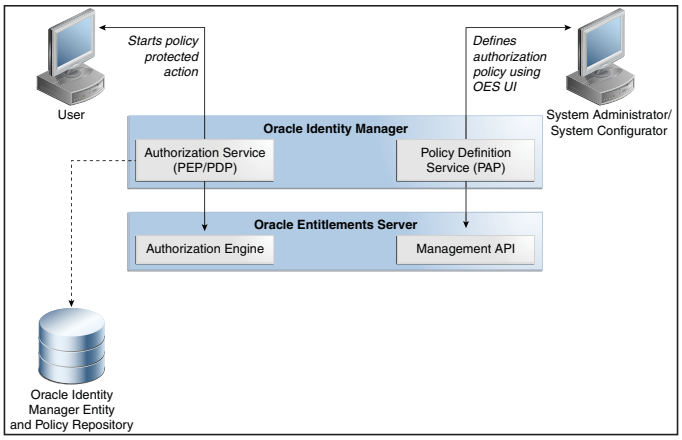
In Oracle Identity Manager, authorization policy management is centralized as an administrative feature. Oracle Identity Manager's authorization policy management and enforcement engine is based on an embedded version of Oracle Entitlements Server (OES), which is Oracle's entitlements administration product. These authorization policies secure access control to the Oracle Identity Manager application, thereby defining 'who can do what on what data' inside the application.

Oracle Identity Manager supports the following:

- Use standard ADF security model for functional security and use OES best practices for data security.
- Use a consistent architecture that supports delegated administration of various entities in Oracle Identity Manager, such as roles, organizations, entitlements, application instances, and LDAP groups.
- Use a consistent architecture that lets backend make various security decisions, for example, who can request what, who can have what, and who needs to go through approval. This architecture facilitates the security of catalog-based request module and of converged UI and backend of self service and delegated-administration.
- Support for a scoping mechanism for delegated administration and data security of various entities. All entities are scoped by the organization structure defined as Oracle Identity Manager metadata.

Figure 3–1 shows the architecture of OES-based authorization service.

Figure 3–1 OES-Based Authorization Service



The authorization and security model is described in the following sections:

- Security Model
- Functional and Data Security Mapping
- Publishing Entities to Organizations
- Managing OES Policies
- Enforcing Functional Security

3.1 Security Model

The security model is described in the following sections:

- Admin Role Assignment
- Attribute-Level Security for the User Attributes
- Policy Obligations

3.1.1 Admin Role Assignment

The new authorization model works on the basis of the admin role assignment to a user. There are two types of admin roles, global and scoped. Global admin roles, such as System Administrator, System Configuration Administrator, Catalog System Administrator, SPML Admin, Certification Administrator, and Certification Viewer, can only be assigned in the context of the Top organization only. Scoped admin roles can be assigned in the context of both Top as well as other organizations.

The Top organization is at the root of the organization hierarchy in Oracle Identity Manager. Authorization policies are created according to the admin roles. Admin roles are predefined in Oracle Identity Manager, and new admin roles cannot be added. Admin roles cannot be created, updated, deleted, searched, or requested.

Admin roles are predefined for each entity type. The entity type admin roles are scoped because entity management is performed by delegated administrators. Each entity has the following admin roles defined for it:

- **Entity Administrator:** Can manage the entire lifecycle of the entity and perform any operation on the entity.
- **Entity Viewer:** Can view the entity in the catalog or request profile and request for the entity.

- **Entity Authorizer:** Can view the entity in the catalog or request profiles and request for it, but does not require approval. There is no authorizer on the organization entity because organization membership cannot be requested. Similarly, there is no authorizer for the user. The user admin and user authorizer are the same.

Note:

- Entity refers to role, user, organization, entitlement, and application instance.
- See "Admin Roles" in the *Oracle Fusion Middleware User's Guide for Oracle Identity Manager* for a description of each admin role

Admin roles have no hierarchy. However, admin role membership organization scoping is hierarchy-aware, and can be cascaded downwards to the child organizations. Admin role membership is always given in an organization scope, and can only be assigned by the System Administrator or the Organization Administrator. The Organization Administrator can assign the admin role for that organization for which it is the Organization Administrator. System Configuration Administrators cannot assign admin roles. Admin roles do not have autogroup membership or role membership rules.

Note:

- Admin roles are stored only in Oracle Identity Manager database and are not stored or synchronized in LDAP data store.
- The admin roles cannot be requested and are never exposed to the users.

The System Administrator and System Configuration Administrator admin roles are available only to the Top organization. Therefore, only System Administrators can assign System Administrator and System Configuration Administrator roles because they have access to the Top organization.

The permissions a user has on a entity can be of the following types:

- **Inherent permissions:** The organization to which a user is a member is referred as the Home organization for that user. A user has certain implicit permissions on the entities available to the Home organization. These permissions are automatically assigned to a user. For example, a Role Administrator does not need explicit Role Viewer privileges to view and request for roles available to the Home organization. However, to view and request for roles in another organization, Role Viewer privileges must be explicitly assigned to the same Role Administrator.
- **Management hierarchy:** If User A is the manager of User B and User C, then User A has implicit permissions on User B and User C. If User B and User C are in a different organization, then User A has implicit permissions on User B and User C. User A does not need explicit privileges on the direct reports, irrespective of which organization the direct reports belong. Privileges through management hierarchy is applicable globally, and every manager is able to perform user administration operations on their reports.

Each admin role in Oracle Identity Manager has one-to-one mapping to the application roles in the OES. The application roles have associated policies that govern what permissions are allowed for users who belong to this role. If you want to change

the functional and data constraints on these policies, then you must open the respective policy in Authorization Policy Management (APM) UI in OES, and modify the policy.

Table 3–1 lists the organization-scoped admin roles in Oracle Identity Manager and the corresponding permissions provided by the admin roles.

Note: In Table 3–1 and Table 3–2, you will come across implicit permissions called org basic info, role basic info, entitlement basic info, and appinstance basic info. The basic-info permission gives the permission only to view-search the given entity. Consider the following examples:

- View Org permission provides all the permissions defined for the Organization Viewer admin role, but org basic info provides the permissions only to search and view the organization attributes.
- The User Viewer admin role provides the basic info permission on roles, organizations, application instances, and entitlements in that scoped organization.

Table 3–1 Organization-Scoped Admin Roles and Permissions

Admin Role in Oracle Identity Manager	Implicit Permissions	Organization Scoped Permissions	Request or Direct Operation
User Administrator	Organization Viewer	Search User (attribute-level security)	NA
	Role Viewer	View User (attribute-level security)	NA
	Entitlement Viewer	Create User	Direct
	AppInstance Viewer	Delete User	Direct
		Modify User (attribute-level security)	Direct
		Lock User	NA
		Unlock User	NA
		Enable User	Direct
		Disable User	Direct
		Grant Role	Direct
		Revoke Role	Direct
		Grant Accounts	Direct
		Revoke Accounts	Direct
		Grant Entitlements	Direct
		Revoke Entitlements	Direct
		Change User Password	NA
		Change Account Passwords	NA
		Modify User Account	Direct
		Enable User Account	Direct

Table 3–1 (Cont.) Organization-Scoped Admin Roles and Permissions

Admin Role in Oracle Identity Manager	Implicit Permissions	Organization Scoped Permissions	Request or Direct Operation
		Disable User Account	Direct
		View Org	NA
		View Role	NA
		View Entitlements	NA
		View Application Instance	NA
		View Requests	NA
		View Admin Role Memberships	NA
		View Role Memberships	NA
		View User Accounts	NA
		View User Entitlements	NA
		View Proxy	NA
		Add Proxy	Direct
		Delete Proxy	Direct
Help Desk	Org Basic Info	Search User (attribute-level security)	NA
	Role Basic Info	View User (attribute-level security)	NA
	Entitlement Basic Info	Enable User	Request
	AppInstance Basic Info	Disable User	Request
		Unlock User ONLY IF locked out due to failed logins	Direct
		Change User Password	Direct
		View Org	NA
		View Role	NA
		View Entitlements	NA
		View Application Instance	NA
		View Requests	NA
		View Role Memberships	NA
		View Proxy	NA
		View User Accounts	NA
		View User Entitlements	NA
User Viewer	Organization Viewer	Create User	Request
	Role Viewer	Delete User	Request

Table 3–1 (Cont.) Organization-Scoped Admin Roles and Permissions

Admin Role in Oracle Identity Manager	Implicit Permissions	Organization Scoped Permissions	Request or Direct Operation
	Entitlement Viewer	Modify User (attribute-level security)	Request
	AppInstance Viewer	Search User (attribute-level security)	NA
		View User (attribute-level security)	NA
		Enable User	Request
		Disable User	Request
		Grant Role	Request
		Revoke Role	Request
		Grant Accounts	Request
		Revoke Accounts	Request
		Grant Entitlements	Request
		Revoke Entitlements	Request
		Modify User Account	Request
		View Org	NA
		View Role	NA
		View Entitlements	NA
		View Application Instance	NA
		View Requests	NA
		View Role Memberships	NA
		View Proxy	NA
		Enable User Account	Request
		Disable User Account	Request
		View Admin Role Memberships	NA
		Add Admin roles	NA
		Delete Admin roles	NA
		Modify Admin Role membership	NA
		View User Accounts	NA
		View User Entitlements	NA
Role Viewer	Org Basic Info	Grant Role	Request
	User Basic Info	Revoke Role	Request
		View Org	NA
		View Role	NA
		View Users	NA

Table 3–1 (Cont.) Organization-Scoped Admin Roles and Permissions

Admin Role in Oracle Identity Manager	Implicit Permissions	Organization Scoped Permissions	Request or Direct Operation
		View Role Memberships	NA
Organization Viewer	Org Basic Info	Search Org	NA
	User Basic Info	View Org	NA
	AppInstance Info	View Users	NA
	Entitlement Info	View Role	NA
		View AppInstance	NA
		View Entitlement	NA
		View All Publications	NA
		View All Org Members	NA
		View Admin Role & memberships	NA
		View Accounts Provisioned to Org	NA
Application Instance Viewer	User Basic Info	Search Application Instance	NA
	Org Basic Info	View Application Instance (excluding passwords)	NA
	Entitlement Info	Grant Account	Request
		Revoke Accounts	Request
		Modify User Account	Request
		Enable User Account	Request
		Disable User Account	Request
		View Org	NA
		View User	NA
		View AppInstance	NA
		View Entitlements	NA
		View User Accounts	NA
		View User Entitlements	NA
Entitlement Viewer	User Basic Info	Search Entitlement	NA
	Org Basic Info	View Entitlement	NA
	AppInstance Basic Info	Grant Entitlement	Request
		Revoke Entitlement	Request
		View Orgs	NA
		View Users	NA
		View AppInstance	NA
		View User Accounts	NA

Table 3–1 (Cont.) Organization-Scoped Admin Roles and Permissions

Admin Role in Oracle Identity Manager	Implicit Permissions	Organization Scoped Permissions	Request or Direct Operation
		View User Entitlements	NA
Role Administrator **Note:** The Role Administrator admin role can only manage the lifecycle of roles within their organization scope but does not have the permissions to grant/revoke roles to/from any user. If Role administrator needs this functionality, either assign the Role Viewer admin role if request needs to be approved, or the Role Authorizer admin role if request needs no approval, within the scope of the organizations in which they need this functionality.	User Basic Info	Search Role	NA
	Org Basic Info	View Role	NA
		Create Role	Direct
		Modify Role	Direct
		Delete Role	Direct
		View Role Members	NA
		Manage Role Hierarchy	Direct
		Publish role (only to allowed orgs)	Direct
		Unpublish role (only to allowed orgs)	Direct
		Manage Role Membership Rules	Direct
		Create Role Category	Direct
		Update Role Category	Direct
		Delete Role Category	Direct
		View Users	NA
		View Orgs	NA
		View Role Memberships	NA

Table 3–1 (Cont.) Organization-Scoped Admin Roles and Permissions

Admin Role in Oracle Identity Manager	Implicit Permissions	Organization Scoped Permissions	Request or Direct Operation
Application Instance Administrator **Note:** The Application Instance Administrator admin role can only manage the lifecycle of application instances within their organization scope but does not have the permissions to grant/revoke application instances to/from any user. If Application Instance administrator needs this functionality, either assign the Application Instance Viewer admin role if request needs to be approved, or the Application Instance Authorizer admin role if request needs no approval, within the scope of the organizations in which they need this functionality.	User Basic Info	Create Application instance	Direct
	Org Basic Info	Modify Application instance	Direct
	Entitlement Administrator	Delete Application instance	Direct
		Search Application Instance	NA
		View Application Instance	NA
		Publish Application Instance (only to allowed orgs)	Direct
		Unpublish Application Instance (only to allowed orgs)	Direct
		Publish Entitlements (only to allowed orgs)	Direct
		Unpublish Entitlements (only to allowed orgs)	Direct
		Access Advanced UI	NA
		View accounts	NA
		View Users	NA

Table 3–1 (Cont.) Organization-Scoped Admin Roles and Permissions

Admin Role in Oracle Identity Manager	Implicit Permissions	Organization Scoped Permissions	Request or Direct Operation
		View Orgs	NA
		View User Accounts	NA
		View User Entitlements	NA
Organization Administrator	User Basic Info	Search Org	NA
	AppInstance Basic Info	View Org	NA
	Entitlement Basic Info	Create Organization	Direct
	Role Basic Info	Modify Organization	Direct
		Delete Organization	Direct
		All Role Admin Privileges for Admin Roles.	Direct
		Update Organization Hierarchy (for a specific organization)	Direct
		Associate password policy	Direct
		View members	NA
		View roles published	NA
		View app instances published	NA
		View entitlements published	NA
		View accounts (provisioned to org) **Note:** Provisioning resources to organization is allowed only to the System Administrator.	NA

Table 3–1 (Cont.) Organization-Scoped Admin Roles and Permissions

Admin Role in Oracle Identity Manager	Implicit Permissions	Organization Scoped Permissions	Request or Direct Operation
Entitlement Administrator **Note:** The Entitlement Administrator admin role can only manage the lifecycle of entitlements within their organization scope but does not have the permissions to grant/revoke entitlements to/from any user. If Entitlement administrator needs this functionality, either assign the Entitlement Viewer admin role if request needs to be approved, or the Entitlement Authorizer admin role if request needs no approval, within the scope of the organizations in which they need this functionality.	User Basic Info	Search Entitlements	NA
	AppInstance Basic Info	View Entitlements	NA
	Org Basic Info	add Entitlements (API)	Direct
		delete Entitlements (API)	Direct
		update Entitlements (API)	Direct
		Publish Entitlement (only to allowed orgs)	Direct
		Unpublish Entitlement (only from allowed orgs)	Direct
		View orgs	NA
		View User	NA
		View app instance	NA
		View accounts	NA
		View Entitlement Members	NA
		View Published Entitlements (API) org data security applies	NA
Catalog System Administrator	AppInstance Basic Info	Edit Catalog metadata	Direct

Table 3–1 (Cont.) Organization-Scoped Admin Roles and Permissions

Admin Role in Oracle Identity Manager	Implicit Permissions	Organization Scoped Permissions	Request or Direct Operation
	Entitlement Basic Info	Create Request Profiles	Direct
	Role Basic Info	Modify Request Profiles	Direct
		Delete Request Profiles	Direct
		View application instances	NA
		View entitlements	NA
		View roles	NA
Role Authorizer	User Basic Info	View Role	NA
	Org Basic Info	Grant Role	Direct
		Revoke Role	Direct
		View Orgs	NA
		View Users	NA
		View Role Memberships	NA
Application Instance Authorizer	User Basic Info	Search Application Instance	NA
	Org Basic Info	View Application Instance (excluding passwords)	NA
		Grant account	Direct
		Revoke account	Direct
		Modify account	Direct
		Enable account	Direct
		Disable account	Direct
		View Org	NA
		View Entitlements	NA
		View Users	NA
		View User Accounts	NA
		View User Entitlements	NA
Entitlement Authorizer	User Basic Info	Search Entitlement	NA
	Org Basic Info	View Entitlement	NA
	AppInstance Basic Info	Grant Entitlement	Direct
		Revoke Entitlement	Direct
		View Users	NA
		View Orgs	NA

Table 3–1 (Cont.) Organization-Scoped Admin Roles and Permissions

Admin Role in Oracle Identity Manager	Implicit Permissions	Organization Scoped Permissions	Request or Direct Operation
		View Application Instance	NA
		View User Accounts	NA
		View User Entitlements	NA

Table 3–2 lists the global admin roles in Oracle Identity Manager and the corresponding permissions provided by the admin roles. These admin roles are assigned through the Top organization.

Note: The System Administrator admin role is not listed in Table 3–2. The System Administrator admin role has global permissions for all operations.

Table 3–2 Global Admin Roles and Permissions

Admin Role in Oracle Identity Manager	Implicit Permissions	Explicit Permissions	Request or Direct Operation
Catalog System Administrator	App Instance Basic Info	Edit Catalog metadata	Direct
	Entitlement Basic Info	Create Request Profiles	Direct
	Role Basic Info	Modify Request Profiles	Direct
		Delete Request Profiles	Direct
		View Application Instances	NA
		ViewRentitlements	NA
		View Roles	NA
System Configuration Administrator	Role Basic Info	View Forms	NA
	Org Basic Info	Create Forms	NA
	Application Instance Basic Info	Modify Forms	NA
	Entitlement Basic Info	Delete Forms	NA
		Import Connector	NA
		Export Connector	NA
		View Resource Object	NA
		Create Resource Object	NA
		Modify Resource Object	NA
		Delete Resource Object	NA

Table 3–2 (Cont.) Global Admin Roles and Permissions

Admin Role in Oracle Identity Manager	Implicit Permissions	Explicit Permissions	Request or Direct Operation
		View Application Instance	NA
		Create Application Instance	NA
		Modify Application Instance	NA
		Delete Application Instance	NA
		Publish Application Instance	NA
		View Entitlement	NA
		Publish Entitlement	NA
		Delete Entitlement (using APIs)	NA
		Modify Entitlement (using APIs)	NA
		Add Entitlement (using APIs)	NA
		View Approval Policies	NA
		Create Approval Policies	NA
		Modify Approval Policies	NA
		Delete Approval Policies	NA
		Access Advanced UI	NA
		View Password Policy	NA
		Create Password Policy	NA
		Modify Password Policy	NA
		Delete Password Policy	NA
		View Notification	NA
		Create Notification	NA
		Delete Notification	NA
		Modify Notification	NA
		Add Locale to Notification	NA
		Remove Locale To Notification	NA

Table 3–2 (Cont.) Global Admin Roles and Permissions

Admin Role in Oracle Identity Manager	Implicit Permissions	Explicit Permissions	Request or Direct Operation
		Complete Async Event Handlers	NA
		Orchestration Operation	NA
		Register Plugin	NA
		Unregister Plugin	NA
		View scheduled Jobs	NA
		Search Scheduled Jobs	
		Start Scheduler	NA
		Stop Scheduler	NA
		Add Task	NA
		Modify Task	NA
		Delete Task	NA
		Create Trigger	NA
		Delete Trigger	NA
		Modify Trigger	NA
		View Jobs	NA
		Create Jobs	NA
		Modify Jobs	NA
		Delete Jobs	NA
		Enable Jobs	NA
		Disable Jobs	NA
		Run-now Jobs	NA
		Pause Jobs	NA
		Resume Jobs	NA
		Stop Jobs	NA
		Reset Status	NA
		View System Properties	NA
		Create System Properties	NA
		Modify System Properties	NA
		Delete System Properties	NA
		View Attributes	NA
		Add Attributes	NA
		Modify Attributes	NA

Table 3–2 (Cont.) Global Admin Roles and Permissions

Admin Role in Oracle Identity Manager	Implicit Permissions	Explicit Permissions	Request or Direct Operation
		Delete Attributes	NA
		Add Derived Attributes	NA
SPML Admin		Create, modify, and delete users	Request
		Search users on all the attributes	NA
		Enable user status	Request
		Disable user status	Request
		Add role memberships	Request
		Delete role memberships	Request
		Search roles on all the attributes	NA
		Create, modify, and delete roles	Request
Certification Administrator	Certification Viewer	View Proxy User	NA
	User Basic Info	ViewUser Admin Role	NA
	Role Basic Info	View User Entitlements	NA
	Organization Basic Info	View Requests	NA
	Application Instance Basic Info	View User Accounts	NA
	Entitlement Basic Info	View User Roles	NA
		View Certification Configuration	NA
		Update Certification Configuration	NA
		Update Certification	NA
		Access Advanced UI	NA
		View scheduled Jobs	NA
		Search Scheduled Jobs	NA
		Start Scheduler	NA
		Stop Scheduler	NA
		Add Task	NA
		Modify Task	NA
		Delete Task	NA
		Create Trigger	NA
		Delete Trigger	NA

Table 3–2 (Cont.) Global Admin Roles and Permissions

Admin Role in Oracle Identity Manager	Implicit Permissions	Explicit Permissions	Request or Direct Operation
		Modify Trigger	NA
		View Jobs	NA
		Create Jobs	NA
		Modify Jobs	NA
		Delete Jobs	NA
		Enable Jobs	NA
		Disable Jobs	NA
		Run-now Jobs	NA
		Pause Jobs	NA
		Resume Jobs	NA
		Stop Jobs	NA
Certification Viewer **Note:** The only permission explicitly granted to the Certification Viewer admin role is View Certification. Permissions to view other entities are dynamically granted and scoped to those entities referenced in a certification.		View Certification	NA

Note: The following permissions in Oracle Identity Manager are not governed by OES policies:

- Create / Update / Delete Access Policies
- Add / Modify / Remove Lookup
- Import / Export using the Deployment Manager
- Attestation Administration

3.1.2 Attribute-Level Security for the User Attributes

Oracle Identity Manager supports attribute-level security only for user attributes. The security for all other entities is supported at the entity-instance level.

Oracle Identity Manager contains the default User Viewer, User Administrator, and User HelpDesk admin roles along with the corresponding default authorization policies in OES. The default policies allow the User Viewer and User Administrator to view and modify all the user attributes including the attributes that are added as user-defined fields (UDFs), without requiring any changes to the default policies.

The User Viewer policy has the default constraint set as the `deniedattributes` obligation in the policy, and by default, it contains NULL list for the attributes.

Therefore, all users belonging to the User Viewer role are allowed to view all user attributes by default.

The User Administrator policy has the default constraint set as the `deniedattributes` obligation in the policy with a NULL list of attributes, and all users belonging to the User Administrator role are allowed to view and modify all user attributes by default. The User HelpDesk policy also has the default constraint set as the `deniedattributes` obligation in the policy with a NULL list of attributes.

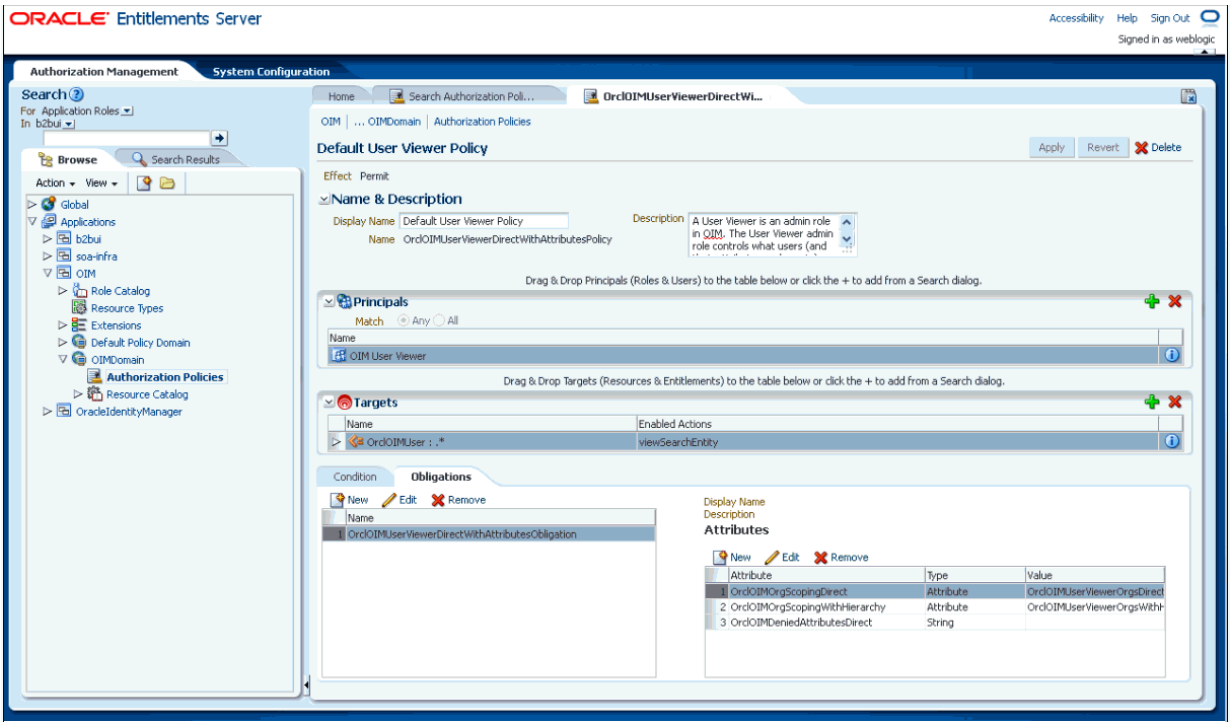
When you add a new UDF, there is no need to change the User Viewer policy. This is because, this policy has default constraint set as deniedattributes, and by default a NULL list for the attributes. This automatically enables the users belonging to the User Viewer role to view the UDFs. There is no need to change the User Administrator policy because the constraint to view and modify all attributes automatically enables the users belonging to the User Administrator role to view and modify these UDFs.

Only if you want to restrict certain attributes to be viewed or modified, then you can change the policies in OES to include such attributes in the deny list. When you want to restrict the list of attributes to be viewed by the User Viewer role or restrict the list of attributes to be viewed and modified by the User Administrator role, you must open the respective policy in the APM UI in OES, and include the list of attributes to be restricted in the deny attribute list of the policy. For example, if you want to restrict the Salary user attribute to be available only for the User Administrator role and not for the User Viewer role, then use the APM UI and modify the User Viewer role to include the Salary attribute in the deny list. When Oracle Identity Manager queries OES to provide a list of attributes for the User Viewer role, OES provides all user attributes but excludes the attributes specified in the Deny List, which is the Salary attribute in this example. Here, there are no changes required for User Admin policy because the 'View and Modify All Attributes' returns the Salary information to be viewed and modified by the users belonging to the User Administrator role.

To change the denied attributes:

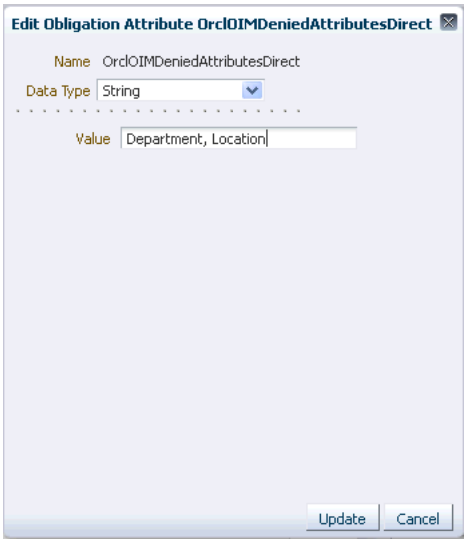
1. Open the required OES policy in APM UI. In this example, an OES policy by name OrclOIMUserViewerDirectWithAttributesPolicy has been opened that gives the permission to view-search user for the User Viewer admin role, as shown in Figure 3–2:

Figure 3–2 The OrclOIMUserViewerDirectWithObligationPolicy



2. You can click the OrclOIMDeniedAttributesDirect attribute in edit mode, and then provide the denied attributes, separated by commas, as shown in Figure 3–3:

Figure 3–3 The Edit Obligation Attribute Dialog Box



3. Click **Update**.
4. In the authorization policy details page, click **Apply** at the top-right corner.

Note: The following deviations are required in APM to deny the listed attributes:

- To deny the User Manager, the attribute must be usr_manager_ key instead of User Manager.
- To deny the User Type, the attribute must be Role instead of User Type.
- Organization cannot be denied.

3.1.2.1 Using Plug-ins to Pass Attributes for Policy Evaluation

You can use the **oracle.iam.platform.authopss.plugin.AttributeResolver** plug-in point to pass the attributes to OES for policy evaluation. To add a new attribute to be used in policies (condition), you must add the attributes in a Map by using the following methods:

To resolve the attributes of the target entity on which the logged-in user is working:

```
public Map<String, Object> resolveResourceAttributes(String subjectId,
PolicyConstants.Resources resourceType, String resourceId) throws Exception;
```

To resolve the attributes related to logged-in user:

```
public Map<String, Object> resolveSubjectAttributes(String subjectId, Policy
Constants.Resources resourceType) throws Exception;
```

Note: See Chapter 27, "Developing Plug-ins" for information about developing and using plug-ins.

The following is an example plug-in class:

```
import java.util.HashMap;
import java.util.HashSet;
import java.util.Map;
import java.util.Set;

import oracle.iam.identity.usermgmt.api.UserManager;
import oracle.iam.identity.usermgmt.vo.User;

import oracle.iam.platform.Platform;
import oracle.iam.platform.authopss.api.PolicyConstants;
import oracle.iam.platform.authopss.plugin.AttributeResolver;

public class ResolveResourceUserTypeAttribute implements AttributeResolver {
    public Map<String, Object> resolveResourceAttributes(String subjectId,
                                                         PolicyConstants.Resources
resourceType,
                                                         String resourceId) {
        if(resourceType!=PolicyConstants.Resources.USER) return null;



        Map<String, Object> env = new HashMap<String, Object>();

        try {
            Set<String> searchAttributes = new HashSet<String>();
```

```
            searchAttributes.add(oracle.iam.identity.utils.Constants.EMPTYPE);
            searchAttributes.add(oracle.iam.identity.utils.Constants.USERID);
            searchAttributes.add(oracle.iam.identity.utils.Constants.USERKEY);

            String empType;

            UserManager userManager = Platform.getService(UserManager.class);
            User user;
            user = userManager.getDetails(resourceId, searchAttributes, false);
            empType =
(String)user.getAttribute(oracle.iam.identity.utils.Constants.EMPTYPE);
            if(empType.equalsIgnoreCase("Full-Time"))
                env.put("OrclOIMResourceUserAttributeType", "EMPLOYEE");
            else env.put("OrclOIMResourceUserAttributeType", "NON-EMPLOYEE");
        } catch (Exception e) {
            e.printStackTrace();
        }
        return env;
    }

    public Map<String, Object> resolveSubjectAttributes(String subjectId,
                                                        PolicyConstants.Resources
resourceType) {
        return null;
    }
}
Added the plugin.xml as:
<?xml version="1.0" encoding="UTF-8"?>
<oimplugins xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://www.oracle.com/schema/oim/plugin
plugin.xsd">

    <plugins pluginpoint="oracle.iam.platform.authopss.plugin.AttributeResolver">
        <plugin
pluginclass="oracle.iam.platform.authopss.plugin.impl.ResolveResourceUserTypeAttri
bute"
            version="1.0" name="ResolveResourceUserTypeAttribute">
         </plugin>
    </plugins>

</oimplugins>
```

3.1.3 Policy Obligations

If a user has multiple roles that have different authorization policies applicable in the same context, then the user's access rights are the cumulative rights across those policies. For example, the authorization check for the permission to search for users returns a list of obligations. This is a list of obligations from each applicable authorization policy. These obligations from multiple policies are combined to get a unified search result.

The following types of obligations are returned as a result of multiple authorization policies:

- **OrclOIMOrgScopingDirect:** This is used to search the given entity for the intent-based search. This is supported only for view-search.
- **OrclOIMOrgScopingWithHierarchy:** This considers the hierarchy of the Admin Role organization scoping, and it can search entities in down hierarchy. This

allows users to view and modify user profiles without approval as applicable for the organization in which the user has the appropriate admin role, and its suborganizations. This is controlled by the Hierarchy Aware data constraint.

- **OrclOIMNeedApproval:** This obligation defines if the authorization policies are applicable, then the operation requires approval or not. If the value of this flag is true, then a request is created. If the value is false, then it is a direct operation.
- **OrclOIMUserManagementScoping:** This is used for making the search criteria to search in the management chain of the user.
- **OrclOIMDeniedAttributesWithoutApproval:** This defines the obligation for the user attributes that are denied for modification without a request approval.
- **OrclOIMDeniedAttributesDirect:** This defined the obligation for the user attributes that are denied for the view user operation as a direct operation.
- **OrclOIMDeniedAttributesWithApproval:** This defines the obligation for the user attributes that are denied for modification with a request approval.

The following are examples of policy obligations returned as a result of multiple authorization policies:

- The user with role viewer admin role for an organization need approval to grant a role to the user. The role viewer can view all users in the organization with hierarchy as a result of OrgScopingWithHierarchy policy obligation. For the same organization, granting a role to a user is a direct operation for a user with the role authorizer admin role.
- Suppose there are two admin roles assigned to a user in the same organization scoping, User Viewer and User Administrator. When both the users try to modify a user, the first admin role policy returns approval-required, and other policy returns that approval is not required. As a result, no request would be raised, and the cumulative effect of two approval-required obligations is NO-approval required.
- As a result of the OrgScopingDirect policy obligation, a user with the role authorizer admin role can view all users in an organization. The same user with role authorizer admin role can be denied modifying a few attributes by the DeniedAttributesWithApproval policy obligation, and as a result, the attributes are not displayed to the user.
- Suppose a user is a Role Viewer in Org1 and Role Authorizer in Org2. Then if the user searches for the roles, then the obligation returned from policy1 is OrgScopingDirect = org1 and OrgScopingDirect = org2. Therefore, roles will be returned from both the organizations.

3.2 Functional and Data Security Mapping

Table 3–3 lists the admin roles and the corresponding application roles, default authorization policies, and policy obligations.

Table 3–3 Default Authorization Policies

Admin Role in Oracle Identity Manager	Application Role in OES	Policy Name	Description	Obligation
Authenticated Role	authenticated-role	Role Category View Policy	This Policy controls if authenticated users can view role categories.	
Role Administrator	OIM Role Administrator	OIM RoleCategory RoleAdmin Policy	This policy controls the creation, modification, and deletion of role categories by the Role Administrator admin role.	
Catalog Administrator	OIM Catalog Administrator Role	Catalog Administration Policy	Catalog Administrator is a global admin role. Catalog Administrators are responsible for managing catalog items and their metadata. This Policy specifies the actions that a member of the role can take.	
Organization Administrator	OIM Organization Administrator	Organization Administration Policy	This policy specifies the actions that an Organization Administrator can perform. This policy can also be configured to require an approval.	OrclOIMOrgScopingWithHierarchy=OrclOIMOrganizationAdminOrgsWithHierarchy OrclOIMOrgScopingDirect=OrclOIMOrganizationAdminOrgsDirect
Organization Administrator	OIM Organization Administrator	OIM OrgAdministrator Basic Info Application Instance Direct Policy	This policy specifies the direct view and search permissions on application instances by Organization Administrators.	OrclOIMOrgScopingDirect=OrclOIMOrganizationAdminOrgsDirect OrclOIMOrgScopingWithHierarchy=OrclOIMOrganizationAdminOrgsWithHierarchy
Organization Administrator	OIM Organization Administrator	OIM OrgAdministrator Basic Info IT Resource Entitlement Direct Policy	This policy specifies the direct view and search permissions on entitlements by Organization Administrators.	OrclOIMOrgScopingWithHierarchy=OrclOIMOrganizationAdminOrgsWithHierarchy OrclOIMOrgScopingDirect=OrclOIMOrganizationAdminOrgsDirect
Organization Administrator	OIM Organization Administrator	OIM OrgAdministrator Basic Info Role Direct Policy	This policy specifies the direct view and search permissions on roles by Organization Administrators.	OrclOIMOrgScopingWithHierarchy=OrclOIMOrganizationAdminOrgsWithHierarchy OrclOIMOrgScopingDirect Attribute=OrclOIMOrganizationAdminOrgsDirect

Table 3–3 (Cont.) Default Authorization Policies

Admin Role in Oracle Identity Manager	Application Role in OES	Policy Name	Description	Obligation
Organization Administrator	OIM Organization Administrator	OIM OrgAdministrator Basic Info User Direct WithAttributes Policy	This policy specifies the direct view and search permissions on users and user attributes by Organization Administrators.	OrclOIMOrgScopingDirect=OrclOIMOrganizationAdminOrgsDirect OrclOIMOrgScopingWithHierarchy=OrclOIMOrganizationAdminOrgsWithHierarchy OrclOIMDeniedAttributesDirect=
Organization Viewer	OIM Organization Viewer	Organization Viewer Policy for View Actions	Organization Viewer is an organization-scoped admin role. This policy specifies the actions that members of this role can take, which do not require approval. By default, the policy specifies that all view actions do not require approval.	OrclOIMOrgScopingDirect=OrclOIMOrganizationViewerOrgsDirect OrclOIMOrgScopingWithHierarchy=OrclOIMOrganizationViewerOrgsWithHierarchy
Organization Viewer	OIM Organization Viewer	OIM OrgViewer Basic Info Application Instance Direct Policy	This policy specifies the direct view and search permissions on application instances by Organization Viewers.	OrclOIMOrgScopingDirect=OrclOIMOrganizationViewerOrgsDirect OrclOIMOrgScopingWithHierarchy=OrclOIMOrganizationViewerOrgsWithHierarchy
Organization Viewer	OIM Organization Viewer	OIM OrgViewer Basic Info IT Resource Entitlement Direct Policy	This policy specifies the direct view and search permissions on entitlements by Organization Viewers.	OrclOIMOrgScopingWithHierarchy=OrclOIMOrganizationViewerOrgsWithHierarchy OrclOIMOrgScopingDirect=OrclOIMOrganizationViewerOrgsDirect
Organization Viewer	OIM Organization Viewer	OIM OrgViewer Basic Info Role Direct Policy	This policy specifies the direct view and search permissions on roles by Organization Viewers.	OrclOIMOrgScopingDirect=OrclOIMOrganizationViewerOrgsDirect OrclOIMOrgScopingWithHierarchy=OrclOIMOrganizationViewerOrgsWithHierarchy

Table 3–3 (Cont.) Default Authorization Policies

Admin Role in Oracle Identity Manager	Application Role in OES	Policy Name	Description	Obligation
Organization Viewer	OIM Organization Viewer	OIM OrgViewer Basic Info User Direct WithAttributes Policy	This policy specifies the direct view and search permissions on users and user attributes by Organization Viewers.	OrclOIMOrgScopingDirect=OrclOIMOrganizationViewerOrgsDirect OrclOIMOrgScopingWithHierarchy=OrclOIMOrganizationViewerOrgsWithHierarchy OrclOIMDeniedAttributesDirect=
Application Instance Administrator	OIM Application Instance Administrator Role	Application Instance Administrator Policy	The Application Instance Administrator admin role is an organization-scoped role. This policy controls the actions that members of the role can perform and whether or not the actions require approval.	OrclOIMOrgScopingDirect=OrclOIMApplicationInstanceAdminOrgsDirect OrclOIMOrgScopingWithHierarchy=OrclOIMApplicationInstanceAdminOrgsWithHierarchy
Application Instance Administrator	OIM Application Instance Administrator Role	OIM ApplicationInstanceAdministrator Basic Info User Direct WithAttributes Policy	This policy specifies the direct view and search permissions on users and user attributes by Application Instance Administrators.	OrclOIMOrgScopingWithHierarchy=OrclOIMApplicationInstanceAdminOrgsWithHierarchy OrclOIMOrgScopingDirect=OrclOIMApplicationInstanceAdminOrgsDirect OrclOIMDeniedAttributesDirect=
Application Instance Administrator	OIM Application Instance Administrator Role	OIM ApplicationInstanceAdministrator Basic Info Organization Direct Policy	This policy specifies the direct view and search permissions on organizations by Application Instance Administrators.	OrclOIMOrgScopingDirect=OrclOIMApplicationInstanceAdminOrgsDirect OrclOIMOrgScopingWithHierarchy=OrclOIMApplicationInstanceAdminOrgsWithHierarchy

Table 3–3 (Cont.) Default Authorization Policies

Admin Role in Oracle Identity Manager	Application Role in OES	Policy Name	Description	Obligation
Application Instance Authorizer	OIM Application Instance Authorizer Role	Application Instance Authorizer Policy	An Application Instance Authorizer is an admin role in Oracle Identity Manager. Application Instance Authorizers can grant/revoke/modify application instances to user accounts without approval. This policy controls whether or not an Application Instance Authorizer can view/search application instances and application instance attributes.	OrclOIMOrgScopingWithHierarchy=OrclOIMApplicationInstanceAuthorizerOrgsWithHierarchy OrclOIMOrgScopingDirect=OrclOIMApplicationInstanceAuthorizerOrgsDirect
Application Instance Authorizer	OIM Application Instance Authorizer Role	Application Instance Authorizer Policy	Application Instance Authorizers can grant/revoke/modify application instances to user accounts without approval. This policy controls whether or not an Application Instance Authorizer can view/search application instances and application instance attributes.	OrclOIMOrgScopingWithHierarchy=OrclOIMApplicationInstanceAuthorizerOrgsWithHierarchy OrclOIMOrgScopingDirect=OrclOIMApplicationInstanceAuthorizerOrgsDirect OrclOIMNeedApproval=false
Application Instance Authorizer	OIM Application Instance Authorizer Role	OIM ApplicationInstanceAuthorizer Basic Info User Direct WithAttributes Policy	This policy specifies the direct view and search permissions on users and user attributes by Application Instance Authorizers.	OrclOIMDeniedAttributesDirect= OrclOIMOrgScopingWithHierarchy=OrclOIMApplicationInstanceAuthorizerOrgsWithHierarchy OrclOIMOrgScopingDirect=OrclOIMApplicationInstanceAuthorizerOrgsDirect
Application Instance Authorizer	OIM Application Instance Authorizer Role	OIM ApplicationInstanceAuthorizer Basic Info Organization Direct Policy	This policy specifies the direct view and search permissions on organizations by Application Instance Authorizers.	OrclOIMOrgScopingDirect=OrclOIMApplicationInstanceAuthorizerOrgsDirect OrclOIMOrgScopingWithHierarchy=OrclOIMApplicationInstanceAuthorizerOrgsWithHierarchy

Table 3–3 (Cont.) Default Authorization Policies

Admin Role in Oracle Identity Manager	Application Role in OES	Policy Name	Description	Obligation
Application Instance Viewer	OIM Application Instance Viewer Role	OIM Application Instance Viewer Direct Policy	This policy specifies the operations that Application Instance Viewers can perform directly.	OrclOIMOrgScopingWithHierarchy=OrclOIMApplicationInstanceViewerOrgsWithHierarchy OrclOIMOrgScopingDirect=OrclOIMApplicationInstanceViewerOrgsDirect
Application Instance Viewer	OIM Application Instance Viewer Role	Application Instance Viewer Policy for Request actions	The Application Instance Viewer admin role is an organization-scoped role. This policy controls the actions that members of the role can perform and whether or not the actions require approval.	OrclOIMOrgScopingDirect=OrclOIMApplicationInstanceViewerOrgsDirect OrclOIMNeedApproval=true OrclOIMOrgScopingWithHierarchy=OrclOIMApplicationInstanceViewerOrgsWithHierarchy
Application Instance Viewer	OIM Application Instance Viewer Role	OIM ApplicationInstanceViewer Basic Info IT Resource Entitlement Direct Policy	This policy specifies the direct view and search permissions on entitlements by Application Instance Viewers.	OrclOIMOrgScopingDirect=OrclOIMApplicationInstanceViewerOrgsDirect OrclOIMOrgScopingWithHierarchy=OrclOIMApplicationInstanceViewerOrgsWithHierarchy
Application Instance Viewer	OIM Application Instance Viewer Role	OIM ApplicationInstanceViewer Basic Info User Direct WithAttributes Policy	This policy specifies the direct view and search permissions on users and user attributes by Application Instance Viewers.	OrclOIMDeniedAttributesDirect= OrclOIMOrgScopingWithHierarchy=OrclOIMApplicationInstanceViewerOrgsWithHierarchy OrclOIMOrgScopingDirect=OrclOIMApplicationInstanceViewerOrgsDirect

Table 3–3 (Cont.) Default Authorization Policies

Admin Role in Oracle Identity Manager	Application Role in OES	Policy Name	Description	Obligation
Application Instance Viewer	OIM Application Instance Viewer Role	OIM ApplicationInstanceViewer Basic Info Organization Direct Policy	This policy specifies the direct view and search permissions on organizations by Application Instance Viewers.	OrclOIMOrgScopingWithHierarchy=OrclOIMApplicationInstanceViewerOrgsWithHierarchy OrclOIMOrgScopingDirect=OrclOIMApplicationInstanceViewerOrgsDirect
Authenticated Role	authenticated-role	Home Org Policy for Application Instances	This Policy allows a user to implicitly view the application instances and application instance attributes that have been published to the user's home organization.	OrclOIMOrgScopingDirect=OrclOIMUserHomeOrgs
Authenticated Role	authenticated-role	Application Instance Policy for Home Org	This policy controls the actions that a user can take on accounts in the user's Home Organization and whether these actions require approval. By default, actions by non-User Administrators on accounts in the same Home Organization require approval.	OrclOIMOrgScopingDirect=OrclOIMUserHomeOrgs OrclOIMNeedApproval=true
System Configuration Administrator	OIM System Configurator	Password Policy Management Policy	This policy controls the password policy management actions that members of the System Administrator or System Configuration Administrator can take.	
Organization Administrator	OIM Organization Administrator	OIM Password Policy OrgAdmin ViewSearch Policy	This policy specifies the view and search permissions on password policies by Organization Administrators.	
Entitlement Administrator	OIM Entitlement Administrator	Entitlement Administrator Policy for entitlement management actions	An Entitlement Administrator is an organization scoped admin role in Oracle Identity Manager. This policy controls the actions a member of this role can perform without requiring approval.	OrclOIMOrgScopingDirect=OrclOIMEntitlementAdminOrgsDirect OrclOIMOrgScopingWithHierarchy=OrclOIMEntitlementAdminOrgsWithHierarchy
Entitlement Administrator	OIM Entitlement Administrator	OIM EntitlementAdministrator Basic Info Application Instance Direct Policy	This policy specifies the direct view and search permissions on application instances by Entitlement Administrators.	OrclOIMOrgScopingWithHierarchy=OrclOIMEntitlementAdminOrgsWithHierarchy OrclOIMOrgScopingDirect=OrclOIMEntitlementAdminOrgsDirect

Table 3–3 (Cont.) Default Authorization Policies

Admin Role in Oracle Identity Manager	Application Role in OES	Policy Name	Description	Obligation
Entitlement Administrator	OIM Entitlement Administrator	OIM EntitlementAdministrator Basic Info User Direct WithAttributes Policy	This policy specifies the direct view and search permissions on users and user attributes by Entitlement Administrators.	OrclOIMOrgScopingWithHierarchy=OrclOIMEntitlementAdminOrgsWithHierarchy OrclOIMOrgScopingDirect=OrclOIMEntitlementAdminOrgsDirect OrclOIMDeniedAttributesDirect=
Entitlement Administrator	OIM Entitlement Administrator	OIM EntitlementAdministrator Basic Info Organization Direct Policy	This policy specifies the direct view and search permissions on organizations by Entitlement Administrators.	OrclOIMOrgScopingWithHierarchy=OrclOIMEntitlementAdminOrgsWithHierarchy OrclOIMOrgScopingDirect=OrclOIMEntitlementAdminOrgsDirect
Entitlement Authorizer	OIM Entitlement Authorizer	Entitlement Authorizer Policy for View Actions	An Entitlement Authorizer is an admin role in Oracle Identity Manager. Entitlement Authorizers can grant/revoke/modify entitlements to user accounts without approval. This policy controls whether an Entitlement Authorizer can view/search entitlements and entitlement attributes.	OrclOIMOrgScopingWithHierarchy=OrclOIMEntitlementAuthorizerOrgsWithHierarchy OrclOIMOrgScopingDirect=OrclOIMEntitlementAuthorizerOrgsDirect
Entitlement Authorizer	OIM Entitlement Authorizer	Entitlement Authorizer Policy for Request Actions	Entitlement Authorizers can grant/revoke/modify entitlements to user accounts without approval. This policy controls the actions that can be performed by an Entitlement Authorizer as part of a request. This policy is used by the request engine to determine if a particular action taken by the Entitlement Authorizer is direct or through request.	OrclOIMOrgScopingDirect=OrclOIMEntitlementAuthorizerOrgsDirect OrclOIMOrgScopingWithHierarchy=OrclOIMEntitlementAuthorizerOrgsWithHierarchy OrclOIMNeedApproval=false
Entitlement Authorizer	OIM Entitlement Authorizer	OIM EntitlementAuthorizer Basic Info Application Instance Direct Policy	This policy specifies the direct view and search permissions on application instances by Entitlement Authorizers.	OrclOIMOrgScopingDirect=OrclOIMEntitlementAuthorizerOrgsDirect OrclOIMOrgScopingWithHierarchy=OrclOIMEntitlementAuthorizerOrgsWithHierarchy

Table 3–3 (Cont.) Default Authorization Policies

Admin Role in Oracle Identity Manager	Application Role in OES	Policy Name	Description	Obligation
Entitlement Authorizer	OIM Entitlement Authorizer	OIM EntitlementAuthorizer Basic Info User Direct WithAttributes Policy	This policy specifies the direct view and search permissions on users and user attributes by Entitlement Authorizers.	OrclOIMDeniedAttributesDirect= OrclOIMOrgScopingDirect=OrclOIMEntitlementAuthorizerOrgsDirect OrclOIMOrgScopingWithHierarchy=OrclOIMEntitlementAuthorizerOrgsWithHierarchy
Entitlement Authorizer	OIM Entitlement Authorizer	OIM EntitlementAuthorizer Basic Info Organization Direct Policy	This policy specifies the direct view and search permissions on organizations by Entitlement Authorizers.	OrclOIMOrgScopingDirect=OrclOIMEntitlementAuthorizerOrgsDirect OrclOIMOrgScopingWithHierarchy=OrclOIMEntitlementAuthorizerOrgsWithHierarchy
Entitlement Viewer	OIM Entitlement Viewer	Entitlement Viewer Policy for View Actions	An Entitlement Viewer is an organization-scoped admin role in Oracle Identity Manager. This Policy specifies whether an entitlement viewer can search for entitlements and view its attributes without approval. By default, no approval is required.	OrclOIMOrgScopingDirect=OrclOIMEntitlementViewerOrgsDirect OrclOIMOrgScopingWithHierarchy=OrclOIMEntitlementViewerOrgsWithHierarchy
Entitlement Viewer	OIM Entitlement Viewer	OIM Entitlement Viewer Policy for Request Actions	This policy is an organization-scoped policy, which allows members of the role to request granting, revoking, and modifying entitlements that are published to their organizations. An entitlement grant or revoke by an Entitlement Viewer results in a request.	OrclOIMOrgScopingWithHierarchy=OrclOIMEntitlementViewerOrgsWithHierarchy OrclOIMOrgScopingDirect=OrclOIMEntitlementViewerOrgsDirect OrclOIMNeedApproval=true
Entitlement Viewer	OIM Entitlement Viewer	OIM EntitlementViewer Basic Info Application Instance Direct Policy	This policy specifies the direct view and search permissions on application instances by Entitlement Viewers.	OrclOIMOrgScopingDirect=OrclOIMEntitlementViewerOrgsDirect OrclOIMOrgScopingWithHierarchy=OrclOIMEntitlementViewerOrgsWithHierarchy

Table 3–3 (Cont.) Default Authorization Policies

Admin Role in Oracle Identity Manager	Application Role in OES	Policy Name	Description	Obligation
Entitlement Viewer	OIM Entitlement Viewer	OIM EntitlementViewer Basic Info User Direct WithAttributes Policy	This policy specifies the direct view and search permissions on users and user attributes by Entitlement Viewers.	OrclOIMOrgScopingWithHierarchy=OrclOIMEntitlementViewerOrgsWithHierarchy OrclOIMDeniedAttributesDirect= OrclOIMOrgScopingDirect=OrclOIMEntitlementViewerOrgsDirect
Entitlement Viewer	OIM Entitlement Viewer	OIM EntitlementViewer Basic Info Organization Direct Policy	This policy specifies the direct view and search permissions on organizations by Entitlement Viewers.	OrclOIMOrgScopingDirect=OrclOIMEntitlementViewerOrgsDirect OrclOIMOrgScopingWithHierarchy=OrclOIMEntitlementViewerOrgsWithHierarchy
Authenticated Role	authenticated-role	Home Org Policy for viewing Entitlements	This Policy allows a user to implicitly view the entitlements and entitlement attributes that have been published to the user's home organization.	OrclOIMOrgScopingDirect=OrclOIMUserHomeOrgs
Authenticated Role	authenticated-role	HomeOrg Policy for actions on Entitlements	This policy specifies the actions that a user can take on the entitlements provisioned to another user in the same home organization, and whether these actions require approval. By default, approval is required.	OrclOIMNeedApproval=true OrclOIMOrgScopingDirect=OrclOIMUserHomeOrgs
Catalog Administrator	OIM Catalog Administrator Role	Request Profile Management Policy	This policy controls the actions that a member of the Catalog Administrator role can perform while managing request profiles.	
Authenticated Role	authenticated-role	OIM Request Profile All User ViewSearch Policy	This policy controls the view and search permissions on requests catalogs by all users.	
System Configuration Administrator	OIM System Configurator	OIM Approval Policy Administrator Policy	This policy controls the permissions for approval policy administration by the System Configuration Administrator.	
System Configuration Administrator	OIM System Configurator	Diagnostic Dashboard Administrator Policy	The Diagnostic Dashboard is a diagnostic utility for Oracle Identity Manager. This policy specifies who can access the Diagnostic Dashboard and what actions they can perform.	

Table 3–3 (Cont.) Default Authorization Policies

Admin Role in Oracle Identity Manager	Application Role in OES	Policy Name	Description	Obligation
System Configuration Administrator	OIM System Configurator	OIM resource object administration Policy	This policy controls the permissions for resource object administration by the System Configuration Administrators.	
System Configuration Administrator	OIM System Configurator	Notification Administrator Policy	This policy specifies the actions that a notification administrator can perform.	
System Configuration Administrator	OIM System Configurator	OIM Platform Service Administrator Policy	This policy specifies the actions that a platform service administrator can perform.	
System Configuration Administrator	OIM System Configurator	Plugin Administrator Policy	This policy controls who can register and unregister plug-ins. By default, only members of the System Administrator and System Configuration Administrator admin roles can register and unregister plug-ins.	
System Configuration Administrator	OIM System Configurator	System Configurator Policy for System Admin Console	This policy controls whether members of the System Configuration Administrator admin role can access Oracle Identity System Administration.	
Application Instance Administrator	OIM Application Instance Administrator	OIM UI App Instance Administrator Policy	This policy specifies the actions that an Application Instance Administrator can perform in the UI.	
Entitlement Administrator	OIM Entitlement Administrator	OIM UI Entitlement Administrator Policy	This policy specifies the actions that an Entitlement Administrator can perform in the UI.	
Application Instance Administrator System Configuration Administrator	OIM Application Instance Administrator OIM System Configurator	Request Dataset Policy	This Policy is used to control the actions that members of the System Configuration Administrator role can perform on request datasets.	OrclOIMOrgScopingDirect=OrclOIMSystemConfiguratorOrgsDirect OrclOIMOrgScopingWithHierarchy=OrclOIMSystemConfiguratorOrgsWithHierarchy
System Configuration Administrator	OIM System Configurator	Reconciliation Administrator Policy	A Reconciliation Administrator can perform actions on reconciliation events. This policy controls what actions a Reconciliation Administrator can perform.	
System Configuration Administrator	OIM System Configurator	OIM Scheduler Administrator Policy	A Scheduler Administrator can perform actions on scheduled tasks. This policy controls what actions a Scheduler Administrator can perform.	

Table 3–3 (Cont.) Default Authorization Policies

Admin Role in Oracle Identity Manager	Application Role in OES	Policy Name	Description	Obligation
System Configuration Administrator	OIM System Configurator	System Properties Administration Policy	This policy specifies the actions and determines who can perform them as part of managing the Oracle Identity Manager system properties. The default behavior allows only the System Configuration Administrators to manage the system properties.	
System Configuration Administrator	OIM System Configurator	OIM User Management Configuration Administrator Policy	This policy controls what user configuration capabilities are available to a member of the System Configuration Administrator role.	
Authenticated Role	authenticated-role	Home Org Policy for Organizations	This policy allows a user to implicitly view the application instances, accounts, entitlements and entitlement attributes, and users that have been published to the user's home organization.	OrclOIMOrgScopingDirect=OrclOIMUserHomeOrgs OrclOIMNeedApproval=true
User Administrator	OIM User Admin	User Admin Policy for user modification	User Admin is an organization-scoped admin role. Members of this role manage users, and their actions do not require approval. This policy specifies whether User Administrators can modify user attributes, the attributes they cannot modify, and whether their modification requires approval. By default, members of this role can modify all user attributes, and their actions do not require approval.	OrclOIMDeniedAttributesWithoutApproval= OrclOIMNeedApproval=false OrclOIMOrgScopingDirect=OrclOIMUserAdminOrgsDirect OrclOIMOrgScopingWithHierarchy=OrclOIMUserAdminOrgsWithHierarchy
User Administrator	OIM User Admin	User Administrator Policy for Admin Actions	A User Administrator is an organization-scoped admin role. Members of this role can perform actions on users in their organizations' scope without approval. This policy covers all actions other than view actions. It returns an obligation indicating that approval is not required for the enabled actions.	OrclOIMNeedApproval=false OrclOIMOrgScopingWithHierarchy=OrclOIMUserAdminOrgsWithHierarchy OrclOIMOrgScopingDirect=OrclOIMUserAdminOrgsDirect

Table 3–3 (Cont.) Default Authorization Policies

Admin Role in Oracle Identity Manager	Application Role in OES	Policy Name	Description	Obligation
User Administrator	OIM User Admin	OIM User Admin Policy direct with attributes	This policy controls the direct actions that the User Administrators can perform on users and user attributes.	OrclOIMDeniedAttributesDirect= OrclOIMOrgScopingWithHierarchy=OrclOIMUserAdminOrgsWithHierarchy OrclOIMOrgScopingDirect=OrclOIMUserAdminOrgsDirect
User Administrator	OIM User Admin	User Admin Policy for non-requestable actions	User Administrator is an organization-scoped admin role. Members of this role manage users, and their actions do not require approval. This Policy specifies the actions a member of the role can perform on a user, which do not require approval.	OrclOIMOrgScopingDirect=OrclOIMUserAdminOrgsDirect OrclOIMOrgScopingWithHierarchy=OrclOIMUserAdminOrgsWithHierarchy
User Help Desk	OIM User Password Admin	Help Desk Policy for managing user status	This policy controls the actions that member of the User Help Desk admin role can take as part of managing a user's account status and whether it requires approvals. By default, members of the role can enable/disable a user's status without approval.	OrclOIMOrgScopingWithHierarchy=OrclOIMUserHelpDeskOrgsWithHierarchy OrclOIMOrgScopingDirect=OrclOIMUserHelpDeskOrgsDirect OrclOIMNeedApproval=true
User Help Desk	OIM User Password Admin	OIM User HelpDesk Policy for modify user accounts	This policy controls the actions that a member of the User Help Desk admin role can take as part of modifying a user's account.	OrclOIMNeedApproval=false OrclOIMOrgScopingDirect=OrclOIMUserHelpDeskOrgsDirect OrclOIMOrgScopingWithHierarchy=OrclOIMUserHelpDeskOrgsWithHierarchy
User Help Desk	OIM User Password Admin	Help Desk Admin Policy for User search	User Help Desk is an organization-scoped admin role. Members of this role can search for users, modify user profiles, and change user passwords. This policy specifies whether members of the role can search for users and whether they can view any user attributes. By default, members of this admin role can see all user attributes.	OrclOIMOrgScopingDirect=OrclOIMUserHelpDeskOrgsDirect OrclOIMDeniedAttributesDirect= OrclOIMOrgScopingWithHierarchy=OrclOIMUserHelpDeskOrgsWithHierarchy

Table 3–3 (Cont.) Default Authorization Policies

Admin Role in Oracle Identity Manager	Application Role in OES	Policy Name	Description	Obligation
User Help Desk	OIM User Password Admin	Help Desk User Policy for Password Management	Members of the User Help Desk admin role can search for users, modify user profiles, and change user passwords. This policy specifies whether members of the role can manage user passwords, lock/unlock accounts, and view requests raised by users	OrclOIMOrgScopingDirect=OrclOIMUserHelpDeskOrgsDirect OrclOIMOrgScopingWithHierarchy=OrclOIMUserHelpDeskOrgsWithHierarchy
User Help Desk	OIM User Password Admin	OIM User HelpDesk UnLockUser Policy direct	This policy determines if the User Help Desk can directly unlock a user account.	OrclOIMOrgScopingWithHierarchy=OrclOIMUserHelpDeskOrgsWithHierarchy OrclOIMOrgScopingDirect=OrclOIMUserHelpDeskOrgsDirect OrclOIMAllowOnlyIfLockedByFailLoginAttempts=true
User Help Desk	OIM User Password Admin	OIM HelpDesk Basic Info Application Instance Direct Policy	This policy specifies the direct view and search permissions on application instances by members of the User Help Desk admin role.	OrclOIMOrgScopingDirect=OrclOIMUserHelpDeskOrgsDirect OrclOIMOrgScopingWithHierarchy=OrclOIMUserHelpDeskOrgsWithHierarchy
User Help Desk	OIM User Password Admin	OIM HelpDesk Basic Info IT Resource Entitlement Direct Policy	This policy specifies the direct view and search permissions on IT resource entitlements by members of the User Help Desk admin role.	OrclOIMOrgScopingWithHierarchy=OrclOIMUserHelpDeskOrgsWithHierarchy OrclOIMOrgScopingDirect=OrclOIMUserHelpDeskOrgsDirect
User Help Desk	OIM User Password Admin	OIM HelpDesk Basic Info Role Direct Policy	This policy specifies the direct view and search permissions on roles by members of the User Help Desk admin role.	OrclOIMOrgScopingWithHierarchy=OrclOIMUserHelpDeskOrgsWithHierarchy OrclOIMOrgScopingDirect=OrclOIMUserHelpDeskOrgsDirect

Table 3–3 (Cont.) Default Authorization Policies

Admin Role in Oracle Identity Manager	Application Role in OES	Policy Name	Description	Obligation
User Help Desk	OIM User Password Admin	OIM HelpDesk Basic Info Organization Direct Policy	This policy specifies the direct view and search permissions on organizations by members of the User Help Desk admin role.	OrclOIMOrgScopingWithHierarchy=OrclOIMUserHelpDeskOrgsWithHierarchy OrclOIMOrgScopingDirect=OrclOIMUserHelpDeskOrgsDirect
User Viewer	OIM User Viewer	User Viewer Policy for Request Actions	User Viewer is an organization-scoped admin role. This policy controls whether a member of the admin role can modify a user's profile and whether the action requires approval or not. By default, user modification requests submitted by members of the User Viewer role require approval.	OrclOIMNeedApproval=true OrclOIMDeniedAttributesWithApproval= OrclOIMOrgScopingDirect=OrclOIMUserViewerOrgsDirect OrclOIMOrgScopingWithHierarchy=OrclOIMUserViewerOrgsWithHierarchy
User Viewer	OIM User Viewer	User Viewer Policy for User management	This policy controls what actions can be performed by a member of the User Viewer role, and whether or not those actions require approval.	OrclOIMOrgScopingWithHierarchy=OrclOIMUserViewerOrgsWithHierarchy OrclOIMOrgScopingDirect=OrclOIMUserViewerOrgsDirect OrclOIMNeedApproval=true
User Viewer	OIM User Viewer	Default User Viewer Policy	The User Viewer admin role controls what users and their attributes and grants an authenticated user can search for and view.	OrclOIMOrgScopingDirect=OrclOIMUserViewerOrgsDirect OrclOIMDeniedAttributesDirect= OrclOIMOrgScopingWithHierarchy=OrclOIMUserViewerOrgsWithHierarchy

Table 3–3 (Cont.) Default Authorization Policies

Admin Role in Oracle Identity Manager	Application Role in OES	Policy Name	Description	Obligation
User Viewer	OIM User Viewer	User Viewer Policy	This policy controls the attributes and the relationships of a user that a member of the User Viewer admin role can view.	OrclOIMOrgScopingDirect=OrclOIMUserViewerOrgsDirect OrclOIMOrgScopingWithHierarchy=OrclOIMUserViewerOrgsWithHierarchy
Authenticated Role	authenticated-role	Management Chain Policy for user modification	This policy specifies whether a user can modify another user in the user's management chain and if the action requires approval. The policy also specifies which user attributes do not require approval. By default, modification of any user attribute excluding password requires approval.	OrclOIMUserManagementScoping=OrclOIMUserId OrclOIMNeedApproval=true OrclOIMDeniedAttributesWithApproval=
Authenticated Role	authenticated-role	Management Chain Policy for actions on users	This policy controls what actions a user can perform on other users in their management chain and whether those actions require approval. By default, approval is required.	OrclOIMNeedApproval=true OrclOIMUserManagementScoping=OrclOIMUserId
Authenticated Role	authenticated-role	Management Chain Policy for User search	This policy allows users to search for other users in their management chain and view allowed attributes. By default, users can view all attributes of other users in their management chain.	OrclOIMDeniedAttributesDirect= OrclOIMUserManagementScoping=OrclOIMUserId
Authenticated Role	authenticated-role	Management Chain Policy for Admin Role actions	This policy controls the actions that a user can take on admin roles granted to other users tin their management chain.	OrclOIMUserManagementScoping=OrclOIMUserId
Authenticated Role	authenticated-role	Home Organization Approval Policy	A home organization is the default organization that a user belongs to. This policy controls what actions a user can take in the user's home organization, and it is used by the request engine to determine whether the action requires approval or not.	OrclOIMOrgScopingDirect=OrclOIMUserHomeOrgs OrclOIMNeedApproval=true
Authenticated Role	authenticated-role	Home Organization Approval with Attributes Policy	This policy controls what actions a user can take in the user's home organization, and it is used by the request engine to determine whether the action requires approval or not.	OrclOIMDeniedAttributesWithApproval=USR_PASSWORD OrclOIMNeedApproval=true OrclOIMOrgScopingDirect=OrclOIMUserHomeOrgs

Table 3–3 (Cont.) Default Authorization Policies

Admin Role in Oracle Identity Manager	Application Role in OES	Policy Name	Description	Obligation
Authenticated Role	authenticated-role	Home Org Policy for User attributes	This policy controls the user attributes that are not visible to users when searching for and viewing user profiles of other users in the same home organization. By default, users can view all attributes.	OrclOIMOrgScopingDirect=OrclOIMUserHomeOrgs OrclOIMDeniedAttributesDirect=
Authenticated Role	authenticated-role	Home Org Policy for viewing user access	This policy controls the actions that a user can take while viewing the access of another user in the same home organization.	OrclOIMOrgScopingDirectAttribute OrclOIMUserHomeOrgs
Authenticated Role	authenticated-role	Policy for modification of self user profile	This policy specifies the user attributes that a user can modify in the user's own user profile, and whether the modification needs approval. By default, a user can modify any attribute in the user's own profile, and the modification requires approval.	OrclOIMNeedApproval=true OrclOIMDeniedAttributesWithApproval=
Authenticated Role	authenticated-role	User Self Service Policy for Request Actions	This policy controls the actions authenticated users can take in Identity Self Service, and whether or not approvals are required.	OrclOIMNeedApproval=true
Authenticated Role	authenticated-role	User attribute view Policy for self	This policy specifies whether an authenticated user can view the user's own user attributes, and the attributes that cannot be viewed. By default, all user attributes can be viewed.	OrclOIMDeniedAttributesDirect=
Authenticated Role	authenticated-role	User Self Service Policy for view actions	This policy specifies the actions that a user can take on the user's own profile, which does not initiate a request.	
SPML Admin	OIM SPML Admin	SPML Admin Policy for User updates	SPML Admin is a global admin role. This admin role is used by the SPML web service to carry out user management operations. This policy specifies whether members of the role can modify users and if the action requires approval. By default, user modification by members of the role requires approval.	OrclOIMOrgScopingDirect=OrclOIMSPMLAdminOrgsDirect OrclOIMNeedApproval=true OrclOIMDeniedAttributesWithApproval= OrclOIMOrgScopingWithHierarchy=OrclOIMSPMLAdminOrgsWithHierarchy

Table 3–3 (Cont.) Default Authorization Policies

Admin Role in Oracle Identity Manager	Application Role in OES	Policy Name	Description	Obligation
SPML Admin	OIM SPML Admin	SPML Admin Policy for actions on Users	This policy controls that actions that a member of the SPML Admin role can take while managing users and whether approval is required. By default, user management actions performed by members of this role require approval.	OrclOIMOrgScopingDirect=OrclOIMSPMLAdminOrgsDirect OrclOIMOrgScopingWithHierarchy=OrclOIMSPMLAdminOrgsWithHierarchy OrclOIMNeedApproval=true
SPML Admin	OIM SPML Admin	SPML Administrator Policy	This policy specifies the actions that the SPML Admin can take on users.	OrclOIMDeniedAttributesDirect= OrclOIMOrgScopingDirect=OrclOIMSPMLAdminOrgsDirect OrclOIMOrgScopingWithHierarchy=OrclOIMSPMLAdminOrgsWithHierarchy
SPML Admin	OIM SPML Admin	SPML Admin Policy for role membership actions	This policy controls the role membership actions that a member of the SPML Admin role can perform and whether the actions require approval. By default, the actions require approval.	OrclOIMOrgScopingWithHierarchy=OrclOIMSPMLAdminOrgsWithHierarchy OrclOIMOrgScopingDirect=OrclOIMSPMLAdminOrgsDirect OrclOIMNeedApproval=true
SPML Admin	OIM SPML Admin	OIM Role SPML Admin Policy direct with attributes	This policy specifies the actions that the SPML Admin can directly take on roles and role attributes.	
Role Authorizer	OIM Role Authorizer	Role Authorizer Policy for View actions	The Role Authorizer admin role is an organization-scoped role. This policy controls the actions a Role Authorizer can perform without requiring approval. Actions, such as viewing role memberships and searching for roles, do not require approval. Searching for roles that are organization-scoped and viewing role members do not require approval.	OrclOIMOrgScopingDirect=OrclOIMRoleAuthorizerOrgsDirect OrclOIMOrgScopingWithHierarchy=OrclOIMRoleAuthorizerOrgsWithHierarchy

Table 3–3 (Cont.) Default Authorization Policies

Admin Role in Oracle Identity Manager	Application Role in OES	Policy Name	Description	Obligation
Role Authorizer	OIM Role Authorizer	Role Authorizer Policy for Request actions	This policy controls the actions a Role Authorizer can perform that require approval. By default, granting and revoking of role membership by a member of this role does not require approval.	OrclOIMNeedApproval=false OrclOIMOrgScopingDirect=OrclOIMRoleAuthorizerOrgsDirect OrclOIMOrgScopingWithHierarchy=OrclOIMRoleAuthorizerOrgsWithHierarchy
Role Authorizer	OIM Role Authorizer	OIM RoleAuthorizer Basic Info Organization Direct Policy	This policy specifies the direct view and search permissions on organizations by Role Authorizers.	OrclOIMOrgScopingWithHierarchy=OrclOIMRoleAuthorizerOrgsWithHierarchy OrclOIMOrgScopingDirect=OrclOIMRoleAuthorizerOrgsDirect
Role Authorizer	OIM Role Authorizer	OIM RoleAuthorizer Basic Info User Direct WithAttributes Policy	This policy specifies the direct view and search permissions on users and user attributes by Role Authorizers.	OrclOIMOrgScopingDirect=OrclOIMRoleAuthorizerOrgsDirect OrclOIMDeniedAttributesDirect= OrclOIMOrgScopingWithHierarchy=OrclOIMRoleAuthorizerOrgsWithHierarchy
Role Viewer	OIM Role Viewer	Role Viewer Policy	A Role Viewer is an admin role in Oracle Identity Manager. This policy controls what actions a member of the role can perform. By default, this policy allows a member of this admin role to search for and view roles.	OrclOIMOrgScopingDirect=OrclOIMRoleViewerOrgsDirect OrclOIMOrgScopingWithHierarchy=OrclOIMRoleViewerOrgsWithHierarchy
Role Viewer	OIM Role Viewer	Role Viewer Policy for Role Membership	This policy controls the actions that a role viewer can perform and whether those actions require approval. By default, approval is required.	OrclOIMOrgScopingDirect=OrclOIMRoleViewerOrgsDirect OrclOIMNeedApproval=true OrclOIMOrgScopingWithHierarchy=OrclOIMRoleViewerOrgsWithHierarchy

Table 3–3 (Cont.) Default Authorization Policies

Admin Role in Oracle Identity Manager	Application Role in OES	Policy Name	Description	Obligation
Role Viewer	OIM Role Viewer	OIM RoleViewer Basic Info Organization Direct Policy	This policy specifies the direct view and search permissions on organizations by Role Viewers.	OrclOIMOrgScopingDirect=OrclOIMRoleViewerOrgsDirect OrclOIMOrgScopingWithHierarchy=OrclOIMRoleViewerOrgsWithHierarchy
Role Viewer	OIM Role Viewer	OIM RoleViewer Basic Info User Direct WithAttributes Policy	This policy specifies the direct view and search permissions on users and user attributes by Role Viewers.	OrclOIMDeniedAttributesDirect= OrclOIMOrgScopingDirect=OrclOIMRoleViewerOrgsDirect OrclOIMOrgScopingWithHierarchy=OrclOIMRoleViewerOrgsWithHierarchy
Authenticated Role	authenticated-role	Home Org Policy for Role memberships	This policy controls the grant role membership and revoke role membership actions that a user can perform in the user's home org and whether it requires approval. By default, approval is required.	OrclOIMNeedApproval=true OrclOIMOrgScopingDirect=OrclOIMUserHomeOrgs
Authenticated Role	authenticated-role	Home Org Policy for Roles	This policy allows a user to implicitly view the roles and role attributes that have been published to the user's home organization.	OrclOIMOrgScopingDirect=OrclOIMUserHomeOrgs
Role Administrator	OIM Role Administrator	OIM Role Administrator Policy with approval	Role Administrator is an organization-scoped admin role. This policy specifies the actions that the Role Administrator can perform with approval.	OrclOIMOrgScopingDirect=OrclOIMRoleAdminOrgsDirect OrclOIMNeedApproval=false OrclOIMOrgScopingWithHierarchy=OrclOIMRoleAdminOrgsWithHierarchy
Role Administrator	OIM Role Administrator	Role Administrator Policy	This Policy controls what actions a member of the Role Administrator admin role can perform.	OrclOIMOrgScopingWithHierarchy=OrclOIMRoleAdminOrgsWithHierarchy OrclOIMOrgScopingDirect=OrclOIMRoleAdminOrgsDirect

Table 3–3 (Cont.) Default Authorization Policies

Admin Role in Oracle Identity Manager	Application Role in OES	Policy Name	Description	Obligation
Role Administrator	OIM Role Administrator	OIM RoleAdministrator Basic Info Organization Direct Policy	This policy specifies the direct view and search permissions on organizations by Role Administrators.	OrclOIMOrgScopingWithHierarchy=OrclOIMRoleAdminOrgsWithHierarchy OrclOIMOrgScopingDirect=OrclOIMRoleAdminOrgsDirect
Role Administrator	OIM Role Administrator	OIM RoleAdministrator Basic Info User Direct WithAttributes Policy	This policy specifies the direct view and search permissions on users and user attributes by Role Administrators.	OrclOIMOrgScopingDirect=OrclOIMRoleAdminOrgsDirect OrclOIMDeniedAttributesDirect= OrclOIMOrgScopingWithHierarchy=OrclOIMRoleAdminOrgsWithHierarchy
System Configuration Administrator	OIM System Configurator	System Configurator Policy for OIM entities	The System Configuration Administrator admin role is a global role. This policy controls what actions a member of the role can perform on users, entitlements, roles, organizations, and application instances. Members can manage application instances in the Identity System Administration, but have viewer admin role capabilities in the Identity Self Service.	OrclOIMOrgScopingWithHierarchy=OrclOIMSystemConfiguratorOrgsWithHierarchy OrclOIMOrgScopingDirect=OrclOIMSystemConfiguratorOrgsDirect
System Configuration Administrator	OIM System Configurator	System Configurator Policy	This policy controls the actions that members of the System Configuration Administrator admin role can perform. Members of this admin role carry out post-install product configuration activities, and can perform all configuration activities that a system administrator can. However, members of the System Configuration Administrator admin role do not have the implicit user, role, and application instance administrator capabilities that members of the System Administrator admin role have.	
System Configuration Administrator	OIM System Configurator	System Configurator Policy deny policy for User	This policy controls the actions that a member of the System Configuration Administrator can perform for the user entity.	

Table 3–3 (Cont.) Default Authorization Policies

Admin Role in Oracle Identity Manager	Application Role in OES	Policy Name	Description	Obligation
Catalog Administrator	OIM Catalog Administrator Role	View Policy for Catalog Administrators	This policy controls the view permission on catalog entities for the Catalog Administrator.	OrclOIMOrgScopingDirect=OrclOIMCatalogAdminOrgsDirect OrclOIMOrgScopingWithHierarchy=OrclOIMCatalogAdminOrgsWithHierarchy
Authenticated Role	authenticated-role	OIM Entity Assigned to User Direct Policy	This policy controls the actions that authenticated users can perform on the assigned entities.	
Authenticated Role	authenticated-role	OIM Entity Assigned to User Approval Policy	This policy controls the actions that authenticated users can perform on the assigned entities.	OrclOIMNeedApproval=true
Certification Administrator	OIM Certification Administrator	OIM UI Certification Administrator Policy	This policy grants access to Identity System Administration UI that contains screens for certification configuration and certification definition.	
Certification Administrator	OIM Certification Administrator	Certification Administrator All Entities Search Policy	This policy grants view and search capability to Oracle Identity Manager entities required to design certification definitions. The entities include application instances, entitlements, organizations, users, enterprise roles, and catalog items. This policy is used to construct certification definitions.	OrclOIMOrgScopingDirect=OrclOIMCertificationAdministratorOrgsDirect OrclOIMOrgScopingWithHierarchy=OrclOIMCertificationAdministratorOrgsWithHierarchy
Certification Administrator	OIM Certification Administrator	Certification Administrator Policy	This policy grants update access to certification configuration objects, such as certification configuration and definitions.	

Table 3–3 (Cont.) Default Authorization Policies

Admin Role in Oracle Identity Manager	Application Role in OES	Policy Name	Description	Obligation
Certification Administrator	OIM Certification Administrator	Scheduler Certification Administrator Policy	This policy grants access to the Scheduler. Certifications are produced from certification definitions by running a scheduled job.	
Certification Administrator	OIM Certification Administrator	Certification Certification Administrator Policy	This policy grants update access to certification instances. **Note:** Certification view and update access for reviewers (non-admin users) are granted directly by the certification authorization.	
Certification Viewer	OIM Certification Viewer	Certification Certification Viewer Policy	This policy grants view access to certification instances. **Note:** Certification Administrator has all Certification Viewer privileges.	

There are some application roles in OES that cannot be granted to users in Oracle Identity Manager, and therefore, do not have corresponding admin roles in Oracle Identity Manager. The policies associated with these application roles are used for request-related operations. For example, the policies associated with the OIM Request Approver application role are used to control the operations of the approver of a request. Table 3–4 lists the application roles that do not have corresponding admin roles in Oracle Identity Manager, and the associated policies.

Table 3–4 OES Application Roles and Policies

Application Role in OES	Policy Name	Description	Obligation
OIM Request Approver	OIM Request Approver Role Policy	This policy specifies the permissions to view and search roles by the request approver.	
OIM Request Requestor	OIM Request Requestor Role Policy	This policy specifies the permissions to view and search roles by the requester.	
OIM Request Beneficiary	OIM Request Beneficiary Role Policy	This policy specifies the permissions to view and search roles by the beneficiary of a request.	
OIM Request Approver	OIM Request Approver ApplicationInstance Policy	This policy specifies the permissions to view and search application instances by the request approver.	
OIM Request Requestor	OIM Request Requestor ApplicationInstance Policy	This policy specifies the permissions to view and search application instances by the requester.	
OIM Request Beneficiary	OIM Request Beneficiary ApplicationInstance Policy	This policy specifies the permissions to view and search application instances by the beneficiary of a request.	

Table 3–4 (Cont.) OES Application Roles and Policies

Application Role in OES	Policy Name	Description	Obligation
OIM Request Approver	OIM Request Approver Entitlement Policy	This policy specifies the permissions to view and search entitlements by the request approver.	
OIM Request Requestor	OIM Request Requestor Entitlement Policy	This policy specifies the permissions to view and search entitlements by the requester.	
OIM Request Beneficiary	OIM Request Beneficiary Entitlement Policy	This policy specifies the permissions to view and search entitlements by the beneficiary of a request.	
OIM Request Approver	OIM Request Approver User Policy	This policy specifies the permissions to view and search users by the request approver.	OrclOIMDeniedAttributes Direct=
OIM Request Requestor	OIM Request Requestor User Policy	This policy specifies the permissions to view and search users by the requester.	OrclOIMDeniedAttributes Direct=
OIM Request Beneficiary	OIM Request Beneficiary User Policy	This policy specifies the permissions to view and search users by the beneficiary of a request.	OrclOIMDeniedAttributes Direct=
OIM Request Delegated Admin	OIM Request Delegated Admin Role Policy	This policy specifies the permissions to view and search roles by the delegated administrators.	
OIM Request Target Entity	OIM Request Target Entity Role Policy	This policy specifies the permissions to view and search roles by the target users of a request.	
OIM Request Delegated Admin	OIM Request Delegated Admin User Policy	This policy specifies the permissions to view and search users by the delegated administrators.	
OIM Request Target Entity	OIM Request Target Entity User Policy	This policy specifies the permissions to view and search users by the target users of a request.	
OIM Request Delegated Admin	OIM Request Delegated Admin ITResEntitlement Policy	This policy specifies the permissions to view and search IT resource entitlements by the delegated administrators.	
OIM Request Target Entity	OIM Request Target Entity ITResEntitlement Policy	This policy specifies the permissions to view and search IT resource entitlements by the target users of a request.	
OIM Request Delegated Admin	OIM Request Delegated Admin ApplicationInstance Policy	This policy specifies the permissions to view and search application instances by the delegated administrators.	
OIM Request Target Entity	OIM Request Target Entity ApplicationInstance Policy	This policy specifies the permissions to view and search application instances by the target users of a request.	

Table 3–4 (Cont.) OES Application Roles and Policies

Application Role in OES	Policy Name	Description	Obligation
OIM Request Certification Viewer	OIM Certification Request Certifier Target Entity Role Policy	This policy grants view access to role entities. This policy is granted to Request Certification Viewer, which is dynamically granted to the logged-in user to view the roles referenced in a certification for which the user is a certifier (reviewer).	
OIM Request Certification Viewer	OIM Certification Request Certifier Target Entity User Policy	This policy grants view access to user entities. This policy is granted to Request Certification Viewer, which is dynamically granted to the logged-in user to view the users referenced in a certification for which the user is a certifier (reviewer).	
OIM Request Certification Viewer	OIM Certification Request Certifier Target Entity ITResEntitlement Policy	This policy grants view access to entitlement entities. This policy is granted to Request Certification Viewer, which is dynamically granted to the logged-in user to view the entitlements referenced in a certification for which the user is a certifier (reviewer).	
OIM Request Certification Viewer	OIM Certification Request Certifier Target Entity ApplicationInstance Policy	This policy grants view access to application instance entities. This policy is granted to Request Certification Viewer, which is dynamically granted to the logged-in user to view the application instances referenced in a certification for which the user is a certifier (reviewer).	

Application-role hierarchies for application roles are defined in OES. This means that a user that has been granted an application role on a given organization can perform all actions of application roles present in that given organization hierarchy. For example, if a user has the OrclOIMUserViewer application role (in other words, the User Viewer Admin role) on a given organization, then the user can perform all the actions of the OrclOIMApplicationInstanceViewerRole, OrclOIMEntitlementViewer, OrclOIMOrgViewer, and OrclOIMRoleViewer application roles present in that given organization.

Table 3–5 lists the mapping between an application role and the corresponding application roles in a given organization. Note that a user that has been granted an application role listed in the second column can perform all the actions by the corresponding application role in the first column.

Table 3–5 Application Role Mapping

Application Role	Application Role Mapped To
OrclOIMRoleViewer	OrclOIMUserAdmin, OrclOIMUserViewer
OrclOIMOrgViewer	OrclOIMUserAdmin, OrclOIMUserViewer, OrclOIMSPMLAdmin
OrclOIMEntitlementViewer	OrclOIMUserAdmin, OrclOIMUserViewer
OrclOIMEntitlementAdministrator	OrclOIMApplicationInstanceAdministratorRole

Table 3–5 (Cont.) Application Role Mapping

Application Role	Application Role Mapped To
OrclOIMApplicationInstanceViewerRole	OrclOIMUserAdmin, OrclOIMUserViewer
OrclOIMCertificationViewer	OrclOIMCertificationAdministrator

In Oracle Identity Manager 11*g* Release 2 (11.1.2.2.0), some of the roles from the earlier release have either been removed or replaced with another role. Table 3–6 provides a mapping between the legacy and new roles.

Table 3–6 Mapping Between Legacy and New Roles

Legacy Role	New Role
SCHEDULER ADMINISTRATORS	SYSTEM CONFIGURATORS
DEPLOYMENT MANAGER ADMINISTRATORS	SYSTEM CONFIGURATORS
NOTIFICATION TEMPLATE ADMINISTRATORS	SYSTEM CONFIGURATORS
SOD ADMINISTRATORS	SYSTEM ADMINISTRATORS
GENERATE_USERNAME_ROLE	SYSTEM ADMINISTRATORS
IDENTITY USER ADMINISTRATORS	USER ADMIN
USER CONFIGURATION ADMINISTRATORS	SYSTEM CONFIGURATORS
ACCESS POLICY ADMINISTRATORS	SYSTEM CONFIGURATORS
RECONCILIATION ADMINISTRATORS	SYSTEM ADMINISTRATORS
RESOURCE ADMINISTRATORS	SYSTEM CONFIGURATORS
GENERIC CONNECTOR ADMINISTRATORS	SYSTEM CONFIGURATORS
APPROVAL POLICY ADMINISTRATORS	SYSTEM CONFIGURATORS
REQUEST ADMINISTRATORS	SYSTEM ADMINISTRATORS
REQUEST TEMPLATE ADMINISTRATORS	SYSTEM CONFIGURATORS
PLUGIN ADMINISTRATORS	SYSTEM CONFIGURATORS
ATTESTATION CONFIGURATION ADMINISTRATORS	SYSTEM CONFIGURATORS
ATTESTATION EVENT ADMINISTRATORS	SYSTEM ADMINISTRATORS
ROLE ADMINISTRATORS	ROLE ADMIN
USER NAME ADMINISTRATOR	The legacy role has been removed and there is no corresponding role in the current release. Will rely on Admin roles.
IDENTITY ORGANIZATION ADMINISTRATORS	ORGANIZATION ADMIN
IT RESOURCE ADMINISTRATORS	APPLICATION INSTANCE ADMIN
REPORT ADMINISTRATORS	The legacy role has been removed and there is no corresponding role for the current release because there are no links to reports from Oracle Identity Manager.

Table 3–6 (Cont.) Mapping Between Legacy and New Roles

Legacy Role	New Role
SPML_APP_ROLE	SPML_APP_ROLE There is no change to this enterprise role. However, a corresponding role with the privileges is seeded in OES. **Note:** This role is not used in Oracle Identity Manager.
ALL USERS	ALL USERS This role will remain as an enterprise role. Therefore, there is no corresponding application role in OES. This role is required in Oracle Identity Manager Enterprise Edition for the access policy-based provisioning operations.
SYSTEM CONFIGURATION ADMINISTRATORS	SYSTEM CONFIGURATORS This role has all privileges as the SYSTEM ADMINISTRATORS role, except for the ability to manage users, roles, organizations, and provisioning. This admin role is used for system configuration tasks for which a complete access to the system as the SYSTEM ADMINISTRATORS role is not required.
SYSTEM ADMINISTRATORS	SYSTEM ADMINISTRATORS This role remains as is to provide full privileges on the system. This role allows unrestricted permissions enforced at the code level (no declarative security model for this role). Therefore, there are no corresponding policies in OES for this role.

The policies listed in Table 3–7 provides the createUser permission to logged-in users having the admin-roles as given in the table.

Table 3–7 Policies for Create User

Admin Role	Policy Name	Policy Display Name
User Admin	OrclOIMUserAdminApprovalPolicy	User Admin Approval Policy
Home Organization or Same Organization	OrclOIMUserHomeOrgApprovalPolicy	User Home Org Approval Policy
User Manager or Management Hierarchy	OrclOIMUserManagementChainApprovalPolicy	User Management Chain Approval Policy
SPML Admin	OrclOIMUserSPMLAdminApprovalPolicy	User SPML Admin Approval Policy
User Viewer	OrclOIMUserViewerApprovalPolicy	User Viewer Approval Policy

3.3 Publishing Entities to Organizations

Publishing an entity to an organization is making the entity available to that organization. The enterprise roles, entitlements, or application instances can be published by respective administrators to a list of organizations to enable these to be

granted to the users of those organizations. Enterprise roles, entitlements, and application instances are published to a list of organizations to make these:

- Requestable to users under the list of organizations
- Manageable to the list of organization administrators to manage these roles

When an entity administrator creates an entity, then that entity is automatically made available to all the organizations for which the administrator has entity admin role. For example, when a user with Role Administrator privilege creates an enterprise role, the newly created role is automatically made available to all the organizations on which the user is the Role Administrator. This avoids the need to create and then publish the entities for administrators in their respective organizations (or organization hierarchies). However, if the entity is required to be published to other organizations, then the entity must be manually published.

Entity administrators can publish the entities to organizations by using the entity detail pages. For example, publishing a role to a set of organizations is done from the Organizations tab of the Role Details page.

For information about how to publish the following entities to organizations:

- **Publishing a role to an organization:** See "Publishing Roles to an Organization" in the *Oracle Fusion Middleware User's Guide for Oracle Identity Manager*.
- **Publishing an application instance with or without entitlements to an organization:** See "Publishing an Application Instance to Organizations" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager*.

3.4 Managing OES Policies

As listed in Table 3–3, each admin role in Oracle Identity Manager has a one-to-one mapping with a policy role in OES, which has a corresponding OES policy. To customize the default authorization policies, you can modify the OES policies by using the Authorization Policy Management (APM) UI. For example, to restrict the list of attributes to be viewed by a specific admin role, you can update the OrclOIMDeniedAttributes policy obligation in APM in the corresponding OES policies. Similarly, to restrict the list of attributes to be edited by a specific admin role, you can update the OrclOIMDeniedAttributesWithApproval obligation.

For information about managing OES policies by using the APM UI, see "Managing Policies and Policy Objects" in the *Oracle Fusion Middleware Oracle Authorization Policy Manager Administrator's Guide (Oracle Fusion Applications Edition)*.

3.4.1 Customizing the Authorization Policies

The default functionality of authorization security in Oracle Identity Manager can be customized in one or more of the following ways:

- Changing the existing policy's allowed actions.
- Changing the obligations. This includes:
  - Changing the denied-attributes list
  - Changing the approval-required Boolean flag from true/false
  - Changing the scoping attributes
- Changing the policy conditions. This includes:
  - Adding a FALSE condition to disable the policy

  - Changing the default condition in the policy
  - Adding a resource/dynamic attribute, and then using those attributes in the existing policies
- Deleting the default policies

The following sections provide the mapping between admin roles and authorization policies that can be modified to customize the default authorization:

- Controlling Who can View Which Users
- Controlling Who can Modify Which Users
- Controlling Who can View Which Links
- Controlling Who can Request an Account in an Application Instance
- Controlling Who can Modify an Account
- Controlling Who can Manage an Application Instance
- Controlling Who can Change User Password
- Controlling Who can Change Account Password
- Controlling Which Operations Are Direct or Request-Based
- Controlling the Denied Attributes for Self
- Controlling Modify User Operation for Self

3.4.1.1 Controlling Who can View Which Users

Controlling who can view which users can be controlled by using the admin roles listed in Table 3–8. These admin roles provide permissions to users to view or search users in scoped organizations.

Table 3–8 Admin Roles to View or Search Users in Scoped Organizations

Admin Role	Associated Authorization Policy Name
ApplicationInstanceAdministratorRole	ApplicationInstanceAdministratorBasicInfoUserDirectWithAttributesPolicy
ApplicationInstanceAuthorizerRole	ApplicationInstanceAuthorizerBasicInfoUserDirectWithAttributesPolicy
ApplicationInstanceViewerRole	ApplicationInstanceViewerBasicInfoUserDirectWithAttributesPolicy
EntitlementAdministrator	EntitlementAdministratorBasicInfoUserDirectWithAttributesPolicy
EntitlementAuthorizer	EntitlementAuthorizerBasicInfoUserDirectWithAttributesPolicy
EntitlementViewer	EntitlementViewerBasicInfoUserDirectWithAttributesPolicy
OrgAdministrator	OrgAdministratorBasicInfoUserDirectWithAttributesPolicy
OrgViewer	OrgViewerBasicInfoUserDirectWithAttributesPolicy
RoleAdministrator	RoleAdministratorBasicInfoUserDirectWithAttributesPolicy
RoleAuthorizer	RoleAuthorizerBasicInfoUserDirectWithAttributesPolicy
RoleViewer	RoleViewerBasicInfoUserDirectWithAttributesPolicy
UserAdmin	UserAdminDirectWithAttributesPolicy
UserHelpDesk	UserHelpDeskDirectWithAttributesPolicy

Table 3–8 (Cont.) Admin Roles to View or Search Users in Scoped Organizations

Admin Role	Associated Authorization Policy Name
No Admin role, but home organization	UserHomeOrgDirectWithAttributesPolicy
No Admin role, but for management hierarchy	UserManagementChainDirectWithAttributesPolicy
SPMLAdmin	UserSPMLAdminDirectWithAttributesPolicy
No Admin role, but for self	UserSelfServiceDirectWithAttributesPolicy
UserViewer	UserViewerDirectWithAttributesPolicy

To control the users that any user can view, update the admin roles assigned, as listed in Table 3–8. If you want to limit the users from home organizations, then you must change the authorization policies.

3.4.1.2 Controlling Who can Modify Which Users

Table 3–9 lists the admin roles and associated authorization policies that control who can modify which users.

Table 3–9 Admin Roles to Modify Users in Scoped Organizations

Admin Role	Associated Authorization Policy Name
UserAdmin	UserAdminApprovalWithAttributesPolicy
No Admin role, but home organization	UserHomeOrgApprovalWithAttributesPolicy
No Admin role, but for management hierarchy	UserManagementChainApprovalWithAttributesPolicy
SPMLAdmin	UserSPMLAdminApprovalWithAttributesPolicy
No Admin role, but for self	UserSelfServiceApprovalWithAttributesPolicy
UserViewer	UserViewerApprovalWithAttributesPolicy

To control the users that any user can modify, update the admin roles assigned, as listed in Table 3–9. If you want to limit the users from home organizations, then you must change the associated authorization policies.

3.4.1.3 Controlling Who can View Which Links

Table 3–10 lists the admin roles, associated authorization policies, and the corresponding links that are enabled because of the policies.

Table 3–10 Admin Roles to Control the View of Links

Admin Role	Associated Authorization Policy Name	Links Enabled Because of the Policy
UserAdmin	UserAdminApprovalPolicy	enableUserStatus, modifyUserAccounts, deleteUserAccounts, addUserEntitlements, disableUserStatus, addUserRoles, createUser, disableUserAccount, deleteUserRoles, deleteUser, deleteUserEntitlements, enableUserAccount, addUserAccounts
UserHelpDesk	UserHelpDeskApprovalPolicy	enableUserStatus, disableUserStatus
UserHelpDesk	UserHelpDeskUserAccountsPolicy	modifyUserAccounts
No Admin role, but home organization	UserHomeOrgApprovalPolicy	enableUserStatus, modifyUserAccounts, deleteUserAccounts, addUserEntitlements, disableUserStatus, createUser, addUserRoles, disableUserAccount, deleteUserRoles, deleteUser, deleteUserEntitlements, enableUserAccount, addUserAccounts
No Admin role, but for management hierarchy	UserManagementChainApprovalPolicy	enableUserStatus, modifyUserAccounts, deleteUserAccounts, addUserEntitlements, disableUserStatus, addUserRoles, createUser, disableUserAccount, deleteUserRoles, deleteUser, deleteUserEntitlements, enableUserAccount, addUserAccounts
No Admin role, but for self	UserSelfServiceApprovalPolicy	addUserRoles, deleteUserRoles, deleteUserEntitlements, deleteUserAccounts, addUserEntitlements, addUserAccounts
SPMLAdmin	UserSPMLAdminApprovalPolicy	enableUserStatus, disableUserStatus, createUser, addUserRoles, deleteUserRoles, deleteUser

Table 3–10 (Cont.) Admin Roles to Control the View of Links

Admin Role	Associated Authorization Policy Name	Links Enabled Because of the Policy
UserViewer	UserViewerApprovalPolicy	enableUserStatus, modifyUserAccounts, deleteUserAccounts, addUserEntitlements, disableUserStatus, addUserRoles, createUser, disableUserAccount, deleteUserRoles, deleteUser, deleteUserEntitlements, enableUserAccount, addUserAccounts

As listed in Table 3–10, the admin roles and associated policies provide the permissions to enable the links according to the allowed actions. You can update the authorization policies and change the enabled links with the policies, or assign the admin roles accordingly.

3.4.1.4 Controlling Who can Request an Account in an Application Instance

Table 3–11 lists the admin roles and associated authorization policies that control which user can request an account in an application instance.

Table 3–11 Admin Roles for Requesting an Account in an Application Instance

Admin Role	Associated Authorization Policy Name
ApplicationInstanceAuthorizerRole	ApplicationInstanceAuthorizerApprovalPolicy
No Admin role, but app-instance published in user's home organization	ApplicationInstanceHomeOrgApprovalPolicy
ApplicationInstanceViewerRole	ApplicationInstanceViewerApprovalPolicy

3.4.1.5 Controlling Who can Modify an Account

Table 3–12 lists the admin roles and associated authorization policies that control who can modify an account.

Table 3–12 Admin Roles for Modifying an Account

Admin Role	Associated Authorization Policy Name
ApplicationInstanceAuthorizerRole	ApplicationInstanceAuthorizerApprovalPolicy
No Admin role, but app-instance published in user's home organization	ApplicationInstanceHomeOrgApprovalPolicy
ApplicationInstanceViewerRole	ApplicationInstanceViewerApprovalPolicy
No admin-role, The app-instance provisioned to the user only	EntityUserAssignmentApprovalPolicy

3.4.1.6 Controlling Who can Manage an Application Instance

The Application Instance Administrator admin role and the associated ApplicationInstanceAdministratorDirectPolicy authorization policy can be used to control who can manage an application instance.

3.4.1.7 Controlling Who can Change User Password

Table 3–13 lists admin roles and associated authorization policies that control who can change user's password.

Table 3–13 Admin Roles for Changing User Password

Admin Role	Associated Authorization Policy Name
UserAdmin	UserAdminDirectPolicy
UserHelpDesk	UserHelpDeskDirectPolicy
No Admin role, but for self	UserSelfServiceDirectPolicy

3.4.1.8 Controlling Who can Change Account Password

Table 3–14 lists the permission on the selected users for the change account password.

Table 3–14 Admin Roles for Permissions on Selected Users

Admin Roles	Associated Authorization Policy Name
UserAdmin	UserAdminDirectPolicy
UserHelpDesk	UserHelpDeskDirectPolicy
No Admin role, but for self	UserSelfServiceDirectPolicy

Table 3–15 is for the selected accounts on which you have the permission to change account password.

Table 3–15 Admin Roles for Permissions on Selected Accounts

Admin Roles	Associated Authorization Policy Name
ApplicationInstanceAuthorizerRole	ApplicationInstanceAuthorizerDirectPolicy
No Admin role, but app-instance published in user's home organization	ApplicationInstanceHomeOrgDirectPolicy
ApplicationInstanceViewerRole	ApplicationInstanceViewerDirectPolicy
No admin-role, The app-instance provisioned to the user only	EntityUserAssignmentDirectPolicy

3.4.1.9 Controlling Which Operations Are Direct or Request-Based

The operations listed in Table 3–16 and the associated authorization policies enable the operations to be request bound. You can change the approvalrequired obligation in the associated authorization policies to make them direct operation.

Table 3–16 Request-Based Operations

Request-Based Operation	Associated Authorization Policy Name
enablenApplicationInstance	ApplicationInstanceHomeOrgApprovalPolicy
revokeApplicationInstance	ApplicationInstanceHomeOrgApprovalPolicy
modifyAccountApplicationInstance	ApplicationInstanceHomeOrgApprovalPolicy
disableApplicationInstance	ApplicationInstanceHomeOrgApprovalPolicy
provisionApplicationInstance	ApplicationInstanceHomeOrgApprovalPolicy
enablenApplicationInstance	ApplicationInstanceViewerApprovalPolicy
revokeApplicationInstance	ApplicationInstanceViewerApprovalPolicy
modifyAccountApplicationInstance	ApplicationInstanceViewerApprovalPolicy
disableApplicationInstance	ApplicationInstanceViewerApprovalPolicy
provisionApplicationInstance	ApplicationInstanceViewerApprovalPolicy
deleteRoleMemberships	EntityUserAssignmentApprovalPolicy
enablenApplicationInstance	EntityUserAssignmentApprovalPolicy
revokeApplicationInstance	EntityUserAssignmentApprovalPolicy
modifyAccountApplicationInstance	EntityUserAssignmentApprovalPolicy
disableApplicationInstance	EntityUserAssignmentApprovalPolicy
revokeITResourceEntitlement	EntityUserAssignmentApprovalPolicy
modifyProvisionedEntitlement	EntityUserAssignmentApprovalPolicy
grantITResourceEntitlement	EntitlementHomeOrgApprovalPolicy
bulkRequestForEntitlements	EntitlementHomeOrgApprovalPolicy
revokeITResourceEntitlement	EntitlementHomeOrgApprovalPolicy
modifyProvisionedEntitlement	EntitlementHomeOrgApprovalPolicy
grantITResourceEntitlement	EntitlementViewerApprovalPolicy
bulkRequestForEntitlements	EntitlementViewerApprovalPolicy
revokeITResourceEntitlement	EntitlementViewerApprovalPolicy
modifyProvisionedEntitlement	EntitlementViewerApprovalPolicy
deleteRoleMemberships	EntityUserAssignmentApprovalPolicy
enablenApplicationInstance	EntityUserAssignmentApprovalPolicy
revokeApplicationInstance	EntityUserAssignmentApprovalPolicy
modifyAccountApplicationInstance	EntityUserAssignmentApprovalPolicy
disableApplicationInstance	EntityUserAssignmentApprovalPolicy

Table 3–16 (Cont.) Request-Based Operations

Request-Based Operation	Associated Authorization Policy Name
revokeITResourceEntitlement	EntityUserAssignmentApprovalPolicy
modifyProvisionedEntitlement	EntityUserAssignmentApprovalPolicy
viewPublishedAccounts	OrganizationHomeOrgDirectPolicy
viewProvisionedAccounts	OrganizationHomeOrgDirectPolicy
viewSearchEntity	OrganizationHomeOrgDirectPolicy
viewPublishedEntitlements	OrganizationHomeOrgDirectPolicy
deleteRoleMemberships	EntityUserAssignmentApprovalPolicy
enablenApplicationInstance	EntityUserAssignmentApprovalPolicy
revokeApplicationInstance	EntityUserAssignmentApprovalPolicy
modifyAccountApplicationInstance	EntityUserAssignmentApprovalPolicy
disableApplicationInstance	EntityUserAssignmentApprovalPolicy
revokeITResourceEntitlement	EntityUserAssignmentApprovalPolicy
modifyProvisionedEntitlement	EntityUserAssignmentApprovalPolicy
deleteRoleMemberships	RoleHomeOrgApprovalPolicy
addRoleMemberships	RoleHomeOrgApprovalPolicy
deleteRoleMemberships	RoleSPMLAdminApprovalPolicy
deleteRole	RoleSPMLAdminApprovalPolicy
addRoleMemberships	RoleSPMLAdminApprovalPolicy
createRole	RoleSPMLAdminApprovalPolicy
modifyRole	RoleSPMLAdminApprovalPolicy
deleteRoleMemberships	RoleViewerApprovalPolicy
addRoleMemberships	RoleViewerApprovalPolicy
enableUserStatus	UserHelpDeskApprovalPolicy
disableUserStatus	UserHelpDeskApprovalPolicy
enableUserStatus	UserHomeOrgApprovalPolicy
modifyUserAccounts	UserHomeOrgApprovalPolicy
deleteUserAccounts	UserHomeOrgApprovalPolicy
addUserEntitlements	UserHomeOrgApprovalPolicy
disableUserStatus	UserHomeOrgApprovalPolicy
createUser	UserHomeOrgApprovalPolicy
addUserRoles	UserHomeOrgApprovalPolicy
disableUserAccount	UserHomeOrgApprovalPolicy
deleteUserRoles	UserHomeOrgApprovalPolicy
deleteUser	UserHomeOrgApprovalPolicy

Table 3–16 (Cont.) Request-Based Operations

Request-Based Operation	Associated Authorization Policy Name
deleteUserEntitlements	UserHomeOrgApprovalPolicy
enableUserAccount	UserHomeOrgApprovalPolicy
addUserAccounts	UserHomeOrgApprovalPolicy
modifyUser	UserHomeOrgApprovalWithAttributesPolicy
enableUserStatus	UserManagementChainApprovalPolicy
modifyUserAccounts	UserManagementChainApprovalPolicy
deleteUserAccounts	UserManagementChainApprovalPolicy
addUserEntitlements	UserManagementChainApprovalPolicy
disableUserStatus	UserManagementChainApprovalPolicy
addUserRoles	UserManagementChainApprovalPolicy
createUser	UserManagementChainApprovalPolicy
disableUserAccount	UserManagementChainApprovalPolicy
deleteUserRoles	UserManagementChainApprovalPolicy
deleteUser	UserManagementChainApprovalPolicy
deleteUserEntitlements	UserManagementChainApprovalPolicy
enableUserAccount	UserManagementChainApprovalPolicy
addUserAccounts	UserManagementChainApprovalPolicy
modifyUser	UserManagementChainApprovalWithAttributesPolicy
enableUserStatus	UserSPMLAdminApprovalPolicy
disableUserStatus	UserSPMLAdminApprovalPolicy
createUser	UserSPMLAdminApprovalPolicy
addUserRoles	UserSPMLAdminApprovalPolicy
deleteUserRoles	UserSPMLAdminApprovalPolicy
deleteUser	UserSPMLAdminApprovalPolicy
modifyUser	UserSPMLAdminApprovalWithAttributesPolicy
addUserRoles	UserSelfServiceApprovalPolicy
deleteUserRoles	UserSelfServiceApprovalPolicy
deleteUserEntitlements	UserSelfServiceApprovalPolicy
deleteUserAccounts	UserSelfServiceApprovalPolicy
addUserEntitlements	UserSelfServiceApprovalPolicy
addUserAccounts	UserSelfServiceApprovalPolicy
modifyUser	UserSelfServiceApprovalWithAttributesPolicy
enableUserStatus	UserViewerApprovalPolicy
modifyUserAccounts	UserViewerApprovalPolicy
deleteUserAccounts	UserViewerApprovalPolicy
addUserEntitlements	UserViewerApprovalPolicy
disableUserStatus	UserViewerApprovalPolicy

Table 3–16 (Cont.) Request-Based Operations

Request-Based Operation	Associated Authorization Policy Name
addUserRoles	UserViewerApprovalPolicy
createUser	UserViewerApprovalPolicy
disableUserAccount	UserViewerApprovalPolicy
deleteUserRoles	UserViewerApprovalPolicy
deleteUser	UserViewerApprovalPolicy
deleteUserEntitlements	UserViewerApprovalPolicy
enableUserAccount	UserViewerApprovalPolicy
addUserAccounts	UserViewerApprovalPolicy
modifyUser	UserViewerApprovalWithAttributesPolicy

To disable the users to search/raise-request for the user's peers except direct reportees, perform the following steps:

1. Disable/deactivate/delete the home-org policies for the user to disallow peer permissioning. These policies are as follows:
   - User Home Org Approval Policy
   - User Home Org Direct Policy
   - User Home Org Direct With Attributes Policy
2. To disallow users to search, view, and raise requests for indirect reportees:
   a. By default, Oracle Identity Manager allows searching, viewing, and raising requests for direct and indirect reportess. To remove the permissioning from indirect reportees, create an authorization plug-in, pass an attribute as isDirectReportee, and return its value as TRUE/FALSE.
   b. Update the following user policies to use the attribute in policy condition:
      – User Management Chain Approval Policy
      – User Management Chain Approval With Attributes Policy
      – User Management Chain Direct With Attributes Policy

3.4.1.10 Controlling the Denied Attributes for Self

To control the denied attributes for self profile, modify policy obligations for the following authorization policies by using APM:

- OrclOIMUserHomeOrgDirectWithAttributesPolicy
- OrclOIMUserSelfServiceDirectWithAttributesPolicy

3.4.1.11 Controlling Modify User Operation for Self

To control the modify user operation for self for the System Administrator:

1. customize the following self service authorization policies:
   - OrclOIMUserViewerApprovalWithAttributesPolicy
   - OrclOIMUserManagementChainApprovalWithAttributesPolicy
   - OrclOIMUserHomeOrgApprovalWithAttributesPolicy

- OrclOIMUserAdminApprovalWithAttributesPolicy

2. Add the following condition in the AND condition along with other checks:

   if OrclOIMUserId == OrclOIMTargetEntity is TRUE

3.4.2 Authorization Policy Customization Use Cases

Authorization policy customization is described with the help of the following use cases:

- Controlling the Attributes an Help-desk Administrator Can View/Modify
- Controlling the Permissions Based on Default Home Organization

3.4.2.1 Controlling the Attributes an Help-desk Administrator Can View/Modify

To control the user attributes that an Help-Desk Administrator can view or modify for an user of type Employee or Non-employee:

1. Using APM, edit the attribute definition to add the following attribute:

```
<attribute>
    <name>OrclOIMResourceUserAttributeType</name>
    <display-name>OIM resource user type.</display-name>
    <description>OIM resource user type.</description>
    <attribute-category>DYNAMIC</attribute-category>

<attribute-type>oracle.security.jps.service.policystore.info.OpssString</attrib
ute-type>
    <is-single-valued>true</is-single-valued>
</attribute>
```

   The possible values for this attribute is EMPLOYEE and NON-EMPLOYEE.

2. Change the authorization policy required for the View User permission:

   a. In the OrclOIMUserHelpDeskDirectWithAttributesPolicy policy, change the condition as follows:

```
<Condition>
  <F nm="OR" exp="bool" params="*">
    <F nm="BOOLEAN_EQUAL" exp="bool" params="*">
      <F nm="ATTRIBUTE_HAS_VALUE" exp="bool" params="*">
        <Par tp="str" val="OrclOIMTargetEntity"/>
      </F>
      <Par tp="bool" val="false"/>
    </F>
    <F nm="OR" exp="bool" params="*">
      <F nm="AND" exp="bool" params="*">
        <F nm="BOOLEAN_EQUAL" exp="bool" params="*">
          <F nm="ATTRIBUTE_HAS_VALUE" exp="bool" params="*">
            <Par tp="str" val="OrclOIMUserOrganizations"/>
          </F>
          <Par tp="bool" val="true"/>
        </F>
        <F nm="BOOLEAN_EQUAL" exp="bool" params="*">
          <F nm="ATTRIBUTE_HAS_VALUE" exp="bool" params="*">
            <Par tp="str" val="OrclOIMUserHelpDeskOrgsDirect"/>
          </F>
          <Par tp="bool" val="true"/>
        </F>
        <F nm="BOOLEAN_EQUAL" exp="bool" params="*">
```

```
            <F nm="STRING_AT_LEAST_ONE_MEMBER_OF" exp="bool" params="*">
              <Par nm="OrclOIMUserHelpDeskOrgsDirect"/>
              <Par nm="OrclOIMUserOrganizations"/>
            </F>
            <Par tp="bool" val="true"/>
          </F>
            <F nm="AND" exp="bool" params="*">
              <F nm="BOOLEAN_EQUAL" exp="bool" params="*">
                <F nm="ATTRIBUTE_HAS_VALUE" exp="bool" params="*">
                  <Par tp="str" val="OrclOIMResourceUserAttributeType"/>
                </F>
                <Par tp="bool" val="true"/>
              </F>
              <F nm="STRING_EQUAL" exp="str" params="2">
                <Par nm="OrclOIMResourceUserAttributeType"/>
                <Par tp="str" val="EMPLOYEE"/>
              </F>
            </F>

        </F>
        <F nm="AND" exp="bool" params="*">
          <F nm="BOOLEAN_EQUAL" exp="bool" params="*">
            <F nm="ATTRIBUTE_HAS_VALUE" exp="bool" params="*">
              <Par tp="str" val="OrclOIMUserOrganizationsWithParents"/>
            </F>
            <Par tp="bool" val="true"/>
          </F>
          <F nm="BOOLEAN_EQUAL" exp="bool" params="*">
            <F nm="ATTRIBUTE_HAS_VALUE" exp="bool" params="*">
              <Par tp="str" val="OrclOIMUserHelpDeskOrgsWithHierarchy"/>
            </F>
            <Par tp="bool" val="true"/>
          </F>
          <F nm="BOOLEAN_EQUAL" exp="bool" params="*">
            <F nm="STRING_AT_LEAST_ONE_MEMBER_OF" exp="bool" params="*">
              <Par nm="OrclOIMUserHelpDeskOrgsWithHierarchy"/>
              <Par nm="OrclOIMUserOrganizationsWithParents"/>
            </F>
            <Par tp="bool" val="true"/>
          </F>
            <F nm="AND" exp="bool" params="*">
              <F nm="BOOLEAN_EQUAL" exp="bool" params="*">
                <F nm="ATTRIBUTE_HAS_VALUE" exp="bool" params="*">
                  <Par tp="str" val="OrclOIMResourceUserAttributeType"/>
                </F>
                <Par tp="bool" val="true"/>
              </F>
              <F nm="STRING_EQUAL" exp="str" params="2">
                <Par nm="OrclOIMResourceUserAttributeType"/>
                <Par tp="str" val="EMPLOYEE"/>
              </F>
            </F>
        </F>
      </F>
    </F>
  </F>
</Condition>
```

Change the denied-attribute obligation and add the attributes denied for the EMPLOYEE user.

b. Duplicate the OrclOIMUserHelpDeskDirectWithAttributesPolicy policy changed in step 1a, and change the condition as follows:

```
<Condition>
  <F nm="OR" exp="bool" params="*">
    <F nm="BOOLEAN_EQUAL" exp="bool" params="*">
      <F nm="ATTRIBUTE_HAS_VALUE" exp="bool" params="*">
        <Par tp="str" val="OrclOIMTargetEntity"/>
      </F>
      <Par tp="bool" val="false"/>
    </F>
    <F nm="OR" exp="bool" params="*">
      <F nm="AND" exp="bool" params="*">
        <F nm="BOOLEAN_EQUAL" exp="bool" params="*">
          <F nm="ATTRIBUTE_HAS_VALUE" exp="bool" params="*">
            <Par tp="str" val="OrclOIMUserOrganizations"/>
          </F>
          <Par tp="bool" val="true"/>
        </F>
        <F nm="BOOLEAN_EQUAL" exp="bool" params="*">
          <F nm="ATTRIBUTE_HAS_VALUE" exp="bool" params="*">
            <Par tp="str" val="OrclOIMUserHelpDeskOrgsDirect"/>
          </F>
          <Par tp="bool" val="true"/>
        </F>
        <F nm="BOOLEAN_EQUAL" exp="bool" params="*">
          <F nm="STRING_AT_LEAST_ONE_MEMBER_OF" exp="bool" params="*">
            <Par nm="OrclOIMUserHelpDeskOrgsDirect"/>
            <Par nm="OrclOIMUserOrganizations"/>
          </F>
          <Par tp="bool" val="true"/>
        </F>
          <F nm="AND" exp="bool" params="*">
            <F nm="BOOLEAN_EQUAL" exp="bool" params="*">
              <F nm="ATTRIBUTE_HAS_VALUE" exp="bool" params="*">
                <Par tp="str" val="OrclOIMResourceUserAttributeType"/>
              </F>
              <Par tp="bool" val="true"/>
            </F>
            <F nm="STRING_EQUAL" exp="str" params="2">
              <Par nm="OrclOIMResourceUserAttributeType"/>
              <Par tp="str" val="NON-EMPLOYEE"/>
            </F>
          </F>

      </F>
      <F nm="AND" exp="bool" params="*">
        <F nm="BOOLEAN_EQUAL" exp="bool" params="*">
          <F nm="ATTRIBUTE_HAS_VALUE" exp="bool" params="*">
            <Par tp="str" val="OrclOIMUserOrganizationsWithParents"/>
          </F>
          <Par tp="bool" val="true"/>
        </F>
        <F nm="BOOLEAN_EQUAL" exp="bool" params="*">
          <F nm="ATTRIBUTE_HAS_VALUE" exp="bool" params="*">
            <Par tp="str" val="OrclOIMUserHelpDeskOrgsWithHierarchy"/>
          </F>
          <Par tp="bool" val="true"/>
        </F>
        <F nm="BOOLEAN_EQUAL" exp="bool" params="*">
          <F nm="STRING_AT_LEAST_ONE_MEMBER_OF" exp="bool" params="*">
```

```
            <Par nm="OrclOIMUserHelpDeskOrgsWithHierarchy"/>
            <Par nm="OrclOIMUserOrganizationsWithParents"/>
          </F>
          <Par tp="bool" val="true"/>
        </F>
          <F nm="AND" exp="bool" params="*">
            <F nm="BOOLEAN_EQUAL" exp="bool" params="*">
              <F nm="ATTRIBUTE_HAS_VALUE" exp="bool" params="*">
                <Par tp="str" val="OrclOIMResourceUserAttributeType"/>
              </F>
              <Par tp="bool" val="true"/>
            </F>
            <F nm="STRING_EQUAL" exp="str" params="2">
              <Par nm="OrclOIMResourceUserAttributeType"/>
              <Par tp="str" val="NON-EMPLOYEE"/>
            </F>
          </F>
      </F>
    </F>
  </F>
</Condition>
```

Change the denied-attribute obligation and add the attributes denied for the NON-EMPLOYEE user.

3. Change the authorization policy required for the Modify User permission:

   a. Duplicate the OrclOIMUserHelpDeskApprovalPolicy policy, name it as OrclOIMUserHelpDeskApprovalWithAttributesPolicy, and put the permission as modifyUser. Change the condition as follows:

```
<Condition>
  <F nm="OR" exp="bool" params="*">
    <F nm="BOOLEAN_EQUAL" exp="bool" params="*">
      <F nm="ATTRIBUTE_HAS_VALUE" exp="bool" params="*">
        <Par tp="str" val="OrclOIMTargetEntity"/>
      </F>
      <Par tp="bool" val="false"/>
    </F>
    <F nm="OR" exp="bool" params="*">
      <F nm="AND" exp="bool" params="*">
        <F nm="BOOLEAN_EQUAL" exp="bool" params="*">
          <F nm="ATTRIBUTE_HAS_VALUE" exp="bool" params="*">
            <Par tp="str" val="OrclOIMUserOrganizations"/>
          </F>
          <Par tp="bool" val="true"/>
        </F>
        <F nm="BOOLEAN_EQUAL" exp="bool" params="*">
          <F nm="ATTRIBUTE_HAS_VALUE" exp="bool" params="*">
            <Par tp="str" val="OrclOIMUserHelpDeskOrgsDirect"/>
          </F>
          <Par tp="bool" val="true"/>
        </F>
        <F nm="BOOLEAN_EQUAL" exp="bool" params="*">
          <F nm="STRING_AT_LEAST_ONE_MEMBER_OF" exp="bool" params="*">
            <Par nm="OrclOIMUserHelpDeskOrgsDirect"/>
            <Par nm="OrclOIMUserOrganizations"/>
          </F>
          <Par tp="bool" val="true"/>
        </F>
          <F nm="AND" exp="bool" params="*">
```

```
                <F nm="BOOLEAN_EQUAL" exp="bool" params="*">
                  <F nm="ATTRIBUTE_HAS_VALUE" exp="bool" params="*">
                    <Par tp="str" val="OrclOIMResourceUserAttributeType"/>
                  </F>
                  <Par tp="bool" val="true"/>
                </F>
                <F nm="STRING_EQUAL" exp="str" params="2">
                  <Par nm="OrclOIMResourceUserAttributeType"/>
                  <Par tp="str" val="EMPLOYEE"/>
                </F>
              </F>

            </F>
            <F nm="AND" exp="bool" params="*">
              <F nm="BOOLEAN_EQUAL" exp="bool" params="*">
                <F nm="ATTRIBUTE_HAS_VALUE" exp="bool" params="*">
                  <Par tp="str" val="OrclOIMUserOrganizationsWithParents"/>
                </F>
                <Par tp="bool" val="true"/>
              </F>
              <F nm="BOOLEAN_EQUAL" exp="bool" params="*">
                <F nm="ATTRIBUTE_HAS_VALUE" exp="bool" params="*">
                  <Par tp="str" val="OrclOIMUserHelpDeskOrgsWithHierarchy"/>
                </F>
                <Par tp="bool" val="true"/>
              </F>
              <F nm="BOOLEAN_EQUAL" exp="bool" params="*">
                <F nm="STRING_AT_LEAST_ONE_MEMBER_OF" exp="bool" params="*">
                  <Par nm="OrclOIMUserHelpDeskOrgsWithHierarchy"/>
                  <Par nm="OrclOIMUserOrganizationsWithParents"/>
                </F>
                <Par tp="bool" val="true"/>
              </F>
                <F nm="AND" exp="bool" params="*">
                  <F nm="BOOLEAN_EQUAL" exp="bool" params="*">
                    <F nm="ATTRIBUTE_HAS_VALUE" exp="bool" params="*">
                      <Par tp="str" val="OrclOIMResourceUserAttributeType"/>
                    </F>
                    <Par tp="bool" val="true"/>
                  </F>
                  <F nm="STRING_EQUAL" exp="str" params="2">
                    <Par nm="OrclOIMResourceUserAttributeType"/>
                    <Par tp="str" val="EMPLOYEE"/>
                  </F>
                </F>
            </F>
          </F>
        </F>
      </Condition>
```

Add the denied attribute obligation as OrclOIMDeniedAttributesWithApproval and add the attributes denied for the EMPLOYEE user.

b. Duplicate the OrclOIMUserHelpDeskApprovalWithAttributesPolicy created in step 2a, and change the condition as follows:

```
<Condition>
  <F nm="OR" exp="bool" params="*">
    <F nm="BOOLEAN_EQUAL" exp="bool" params="*">
      <F nm="ATTRIBUTE_HAS_VALUE" exp="bool" params="*">
```

```
    <Par tp="str" val="OrclOIMTargetEntity"/>
  </F>
  <Par tp="bool" val="false"/>
</F>
<F nm="OR" exp="bool" params="*">
  <F nm="AND" exp="bool" params="*">
    <F nm="BOOLEAN_EQUAL" exp="bool" params="*">
      <F nm="ATTRIBUTE_HAS_VALUE" exp="bool" params="*">
        <Par tp="str" val="OrclOIMUserOrganizations"/>
      </F>
      <Par tp="bool" val="true"/>
    </F>
    <F nm="BOOLEAN_EQUAL" exp="bool" params="*">
      <F nm="ATTRIBUTE_HAS_VALUE" exp="bool" params="*">
        <Par tp="str" val="OrclOIMUserHelpDeskOrgsDirect"/>
      </F>
      <Par tp="bool" val="true"/>
    </F>
    <F nm="BOOLEAN_EQUAL" exp="bool" params="*">
      <F nm="STRING_AT_LEAST_ONE_MEMBER_OF" exp="bool" params="*">
        <Par nm="OrclOIMUserHelpDeskOrgsDirect"/>
        <Par nm="OrclOIMUserOrganizations"/>
      </F>
      <Par tp="bool" val="true"/>
    </F>
      <F nm="AND" exp="bool" params="*">
        <F nm="BOOLEAN_EQUAL" exp="bool" params="*">
          <F nm="ATTRIBUTE_HAS_VALUE" exp="bool" params="*">
            <Par tp="str" val="OrclOIMResourceUserAttributeType"/>
          </F>
          <Par tp="bool" val="true"/>
        </F>
        <F nm="STRING_EQUAL" exp="str" params="2">
          <Par nm="OrclOIMResourceUserAttributeType"/>
          <Par tp="str" val="NON-EMPLOYEE"/>
        </F>
      </F>

  </F>
  <F nm="AND" exp="bool" params="*">
    <F nm="BOOLEAN_EQUAL" exp="bool" params="*">
      <F nm="ATTRIBUTE_HAS_VALUE" exp="bool" params="*">
        <Par tp="str" val="OrclOIMUserOrganizationsWithParents"/>
      </F>
      <Par tp="bool" val="true"/>
    </F>
    <F nm="BOOLEAN_EQUAL" exp="bool" params="*">
      <F nm="ATTRIBUTE_HAS_VALUE" exp="bool" params="*">
        <Par tp="str" val="OrclOIMUserHelpDeskOrgsWithHierarchy"/>
      </F>
      <Par tp="bool" val="true"/>
    </F>
    <F nm="BOOLEAN_EQUAL" exp="bool" params="*">
      <F nm="STRING_AT_LEAST_ONE_MEMBER_OF" exp="bool" params="*">
        <Par nm="OrclOIMUserHelpDeskOrgsWithHierarchy"/>
        <Par nm="OrclOIMUserOrganizationsWithParents"/>
      </F>
      <Par tp="bool" val="true"/>
    </F>
      <F nm="AND" exp="bool" params="*">
```

```
            <F nm="BOOLEAN_EQUAL" exp="bool" params="*">
              <F nm="ATTRIBUTE_HAS_VALUE" exp="bool" params="*">
                <Par tp="str" val="OrclOIMResourceUserAttributeType"/>
              </F>
              <Par tp="bool" val="true"/>
            </F>
            <F nm="STRING_EQUAL" exp="str" params="2">
              <Par nm="OrclOIMResourceUserAttributeType"/>
              <Par tp="str" val="NON-EMPLOYEE"/>
            </F>
          </F>
      </F>
    </F>
  </F>
</Condition>
```

Change the denied-attribute obligation and add the attributes denied for the NON-EMPLOYEE user.

c. Add the plug-in class as follows:

```
public class ResolveResourceUserTypeAttribute implements AttributeResolver
{
    public Map<String, Object> resolveResourceAttributes(String subjectId,

PolicyConstants.Resources resourceType,
                                                          String resourceId)
{
        if(resourceType!=PolicyConstants.Resources.USER) return null;



        Map<String, Object> env = new HashMap<String, Object>();

        try {
            Set<String> searchAttributes = new HashSet<String>();

searchAttributes.add(oracle.iam.identity.utils.Constants.EMPTYPE);

searchAttributes.add(oracle.iam.identity.utils.Constants.USERID);

searchAttributes.add(oracle.iam.identity.utils.Constants.USERKEY);

            String empType;

            UserManager userManager =
Platform.getService(UserManager.class);
            User user;
            user = userManager.getDetails(resourceId, searchAttributes,
false);
            empType =
(String)user.getAttribute(oracle.iam.identity.utils.Constants.EMPTYPE);
            if(empType.equalsIgnoreCase("Full-Time"))
                env.put("OrclOIMResourceUserAttributeType", "EMPLOYEE");
            else env.put("OrclOIMResourceUserAttributeType",
"NON-EMPLOYEE");
        } catch (Exception e) {
            e.printStackTrace();
        }
        return env;
    }
```

```
    public Map<String, Object> resolveSubjectAttributes(String subjectId,

PolicyConstants.Resources resourceType) {
        return null;
    }
}
Added the plugin.xml as:
<?xml version="1.0" encoding="UTF-8"?>
<oimplugins xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://www.oracle.com/schema/oim/plugin
plugin.xsd">

    <plugins
pluginpoint="oracle.iam.platform.authopss.plugin.AttributeResolver">
        <plugin
pluginclass="oracle.iam.platform.authopss.plugin.impl.ResolveResourceUserTy
peAttribute"
            version="1.0" name="ResolveResourceUserTypeAttribute">
         </plugin>
    </plugins>

</oimplugins>
```

A logged-in Help Desk Administrator user can view some attributes of the EMPLOYEE users and some attributes for the NON-EMPLOYEE users. In addition, the Help Desk Administrator user can modify different attributes for EMPLOYEE users and different attributes for NON-EMPLOYEE users.

3.4.2.2 Controlling the Permissions Based on Default Home Organization

To control the permissions based on the default Home organization and use a flag to enable/disable these policies:

1. Using APM, edit the attribute definition of the user to add the following attribute:

```
<attribute>
          <name>OrclOIMHomeOrganizationPoliciesEffective</name>
          <display-name>OIM flag for putting the OES policies for home-org in
effect.</display-name>
          <description>OIM flag for putting the OES policies for home-org in
effect.</description>
          <attribute-category>DYNAMIC</attribute-category>
<attribute-type>oracle.security.jps.service.policystore.info.OpssBoolean</attri
bute-type>
          <is-single-valued>true</is-single-valued>
</attribute>
```

2. Change the policy condition as follows:

```
<Condition>
  <F nm="AND" exp="bool" params="*">
    <F nm="OR" exp="bool" params="*">
      <F nm="BOOLEAN_EQUAL" exp="bool" params="*">
        <F nm="ATTRIBUTE_HAS_VALUE" exp="bool" params="*">
          <Par tp="str" val="OrclOIMHomeOrganizationPoliciesEffective"/>
        </F>
        <Par tp="bool" val="false"/>
      </F>
      <F nm="AND" exp="bool" params="*">
        <F nm="BOOLEAN_EQUAL" exp="bool" params="*">
```

```
            <F nm="ATTRIBUTE_HAS_VALUE" exp="bool" params="*">
              <Par tp="str" val="OrclOIMHomeOrganizationPoliciesEffective"/>
            </F>
            <Par tp="bool" val="true"/>
          </F>
          <F nm="BOOLEAN_EQUAL" exp="str" params="2">
            <Par nm="OrclOIMHomeOrganizationPoliciesEffective"/>
            <Par tp="bool" val="true"/>
          </F>
        </F>
      </F>
      <F nm="OR" exp="bool" params="*">
        <F nm="BOOLEAN_EQUAL" exp="bool" params="*">
          <F nm="ATTRIBUTE_HAS_VALUE" exp="bool" params="*">
            <Par tp="str" val="OrclOIMTargetEntity"/>
          </F>
          <Par tp="bool" val="false"/>
        </F>
        <F nm="AND" exp="bool" params="*">
          <F nm="BOOLEAN_EQUAL" exp="bool" params="*">
            <F nm="ATTRIBUTE_HAS_VALUE" exp="bool" params="*">
              <Par tp="str" val="OrclOIMUserHomeOrgs"/>
            </F>
            <Par tp="bool" val="true"/>
          </F>
          <F nm="BOOLEAN_EQUAL" exp="bool" params="*">
            <F nm="ATTRIBUTE_HAS_VALUE" exp="bool" params="*">
              <Par tp="str" val="OrclOIMUserOrganizationsWithParents"/>
            </F>
            <Par tp="bool" val="true"/>
          </F>
          <F nm="BOOLEAN_EQUAL" exp="bool" params="*">
            <F nm="STRING_AT_LEAST_ONE_MEMBER_OF" exp="bool" params="*">
              <Par nm="OrclOIMUserHomeOrgs"/>
              <Par nm="OrclOIMUserOrganizationsWithParents"/>
            </F>
            <Par tp="bool" val="true"/>
          </F>
        </F>
      </F>
    </F>
  </Condition>
```

3. Add the plug-in class as follows:

```
import java.util.Map;

import oracle.iam.platform.authopss.api.PolicyConstants;
import oracle.iam.platform.authopss.plugin.AttributeResolver;


public class RemoveHomeOrgUserPolicies implements AttributeResolver {
    public Map<String, Object> resolveResourceAttributes(String subjectId,

PolicyConstants.Resources resourceType,
                                                          String resourceId) {
        return null;
    }

    public Map<String, Object> resolveSubjectAttributes(String subjectId,
```

```
PolicyConstants.Resources resourceType) {
        Map<String, Object> env = new HashMap<String, Object>();
        env.put("OrclOIMHomeOrganizationPoliciesEffective", false);
        return env;
    }
}

Added the plugin.xml as:
<?xml version="1.0" encoding="UTF-8"?>
<oimplugins xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://www.oracle.com/schema/oim/plugin
plugin.xsd">

    <plugins
pluginpoint="oracle.iam.platform.authopss.plugin.AttributeResolver">
        <plugin
pluginclass="oracle.iam.platform.authopss.plugin.impl.RemoveHomeOrgUserPolicies
"
            version="1.0" name="RemoveHomeOrgUserPolicies">
         </plugin>
    </plugins>
</oimplugins>
```

The authorization policies related to Home organization will not work for the user entity. Therefore, the end-user cannot perform any operation on peers.

3.5 Enforcing Functional Security

You can enforce security by the following ways:

- **UI-level security:** This is used for UI-level validations to enforce security. For example, you can implement field-level security to ensure that only users with permissions to view and edit fields are able to access the fields. The fields are disabled or not displayed for users who do not have permissions on the fields. This type of security enforcement is at the UI level, and can be overridden if you use APIs to perform the validation.

  Note: To enforce functional security at the UI level, you must be aware of the following:

  - UI components and how to customize the components. See "Customizing the Interface" on page 30-1 for details.
  - Expression Language (EL) syntax and usage. See "Using Expression Language in UI Customization" on page 30-21 for details.

- **Backend security:** To enforce security at the backend, you can modify the OES policies by using the APM UI.

For implementing functional security, first a JAVA authorization file is created in PlatformUI. This file contains the UIPermission variables for all the permissions defined in PolicyConstants (OES policies) for each functionality or page or module. All the authorization files have an entry in the adfc-config.xml file in the MainUI project in JDeveloper.

Implementing functional security involves the following:

- Implementing Task Flow or Region

- Defining Actions
- Implementing Field-Level Security

3.5.1 Implementing Task Flow or Region

This level of implementation determines if the taskflow region is to be hidden or disabled to the user based on the permissions of the user. For securing a region, consider the following example:

On the my-access-accounts.jsff page, the taskflow details-information-tf is rendered selectively to the end users by using an expression that follows the Expression Language (EL) syntax, as shown:

```
rendered="#{oimappinstanceAuth.view[bindings.appInstanceKey].allowed}"
```

Here:

- oimappinstanceAuth is the mapped name of the ApplicationInstanceAuthz.java authorization bean in the adfc-config.xml file.
- view is the name of the UIPermission that is to be checked, where the permission defined in ApplicationInstanceAuthz.java, which is the actual bean file for reference of oimappinstanceAuth, is the following:

  ```
  private UIPermission view = new
  UIPermission(PolicyConstants.Resources.APPLICATION_INSTANCE.getId(),
  PolicyConstants.ApplicationInstanceActions.VIEW_SEARCH.getId());
  ```

- appInstanceKey is the ID of the application instance that the user is trying to view passed as a parameter.

3.5.2 Defining Actions

If actions, such as create, modify, disable, enable, revoke, delete, and withdraw request, are to be hidden or disabled for the user based on the user's permissions. For example, the **Create** button is displayed only to users with permission to create users.

Permissions defined in UserAuthz.java based on PolicyConstants is:

```
private UIPermission create = new UIPermission
(PolicyConstants.Resources.USER.getId(),
PolicyConstants.UserActions.CREATE.getId());
```

Mapping entry for UserAuthz.java in adfc-config.xml in the MainUI project is as follows:

```
<managed-bean id="__30">
<managed-bean-name id="__36">oimuserAuth</managed-bean-name>
<managed-bean-class id="__
29">oracle.iam.ui.platform.view.authz.UserAuthz</managed-bean-class>
<managed-bean-scope id="__31">session</managed-bean-scope>
</managed-bean>
```

Now, you can define EL expression for permission that is defined in the JSFF page. In search-users.jsff, use the following EL expression in the rendered attribute, which is the Create button in this example:

```
<af:commandToolbarButton
rendered="#{oimuserAuth.create.allowed}"
```

The EL expression defined in the rendered attribute hides or shows the button based on the Boolean value returned. Otherwise, the button can be made to read-only by defining the EL expression as disabled attribute instead of rendered. The Create button is now only shown to users whose role have permission defined in policies.

Similarly, you can define EL expressions for other actions, such as modify, enable, and disable. Another example of using EL expressions is to specify that reset password will be available to HelpDesk Admin only, and it will be hidden or read-only for other users.

3.5.3 Implementing Field-Level Security

Fields are displayed based on whether the user has permission to view those fields. For securing display fields, consider the following example:

On the userdetails.jsff page, under the Attributes tab, the user attributes, such as First Name, Last Name, and so on, have been secured by using the following EL expression:

```
rendered="#{oimuserAuth.viewSearch.attributes[bindings.firstName.hints.OIM_
ATTRIBUTE]}"
```

Here:

- oimuserAuth is the mapped name of UserAuthz.java in the adfc-config.xml.
- viewsearch is the UIPermission name, and the Oracle Identity Manager attribute name for the field to be secured is passed as a parameter.

Part II

Application Provisioning

This part describes how to configure application-specific connectors.

It contains the following chapters:

4

Developing Application Instances

Application instance is a provisionable entity. It is a combination of IT resource instance (target connectivity and connector configuration) and resource object (provisioning mechanism). Application instances have business-friendly names that are easier to remember. Creating and managing application instances are performed by using the Oracle Identity System Administration.

Application instances can be connected or disconnected. A connected application instance has a connector defined for the provisioning of entities. A disconnected application instance is used for the provisioning of a disconnected resource, for which a connector is not defined, and therefore, the provisioning is performed manually by the administrator.

For information about application instance concepts and how to create and manage application instances, see "Managing Application Instances" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager*.

This chapter describes how application developers can manage resource objects and IT resources manually. In addition, it describes the procedure to convert a disconnected application instance to a connected application instance.

This chapter includes the following topics related to managing resources, IT resources, and application instances:

4.1 Creating IT Resources

To create an IT resource:

> **Note:** The IT resource type is created before the IT resource can be created. The IT resource type can be created either by using the Design Console, or by importing the IT resource type using the Deployment Manager.

1. Login to Oracle Identity System Administration.

2. Under Configuration, click **IT Resource**. The Manage IT Resource page is displayed.
3. Click **Create IT Resource**. The Create IT Resource wizard is displayed.
4. 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.
5. Click **Continue**.
6. On the Step 2: Specify IT Resource Parameter Values page, specify values for the parameters of the IT resource, and then click **Continue**.
7. 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**.
8. 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**.
9. 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**.

10. Click **Continue**.
11. 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.
12. To proceed with the creation of the IT resource, click **Continue**.
13. 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**.
14. Click **Finish**.

4.2 Managing IT Resources

To locate an IT resource:

1. In Oracle Identity System Administration, under Configuration, click IT Resource. The Manage IT Resource page is displayed.
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

4.2.1 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.
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.

4.2.2 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**.

4.2.3 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**.

Note: Deleting IT resource instances soft-deletes the corresponding application instances.

4.3 Managing Resources By Using the Design Console

This chapter describes resource management in the Design Console. It contains the following sections:

Note: Only the users belonging to the SYSTEM ADMINISTRATORS group of Oracle Identity Manager can log in to Design Console.

- Overview of Resource Management
- IT Resources Type Definition Form
- Rule Designer Form
- Resource Objects Form
- Service Account Management

4.3.1 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 8, "Using the Adapter Factory" for more information about adapters and adapter tasks

4.3.2 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 4–1 shows the IT Resources Type Definition form.

Figure 4–1 The IT Resources Type Definition Form

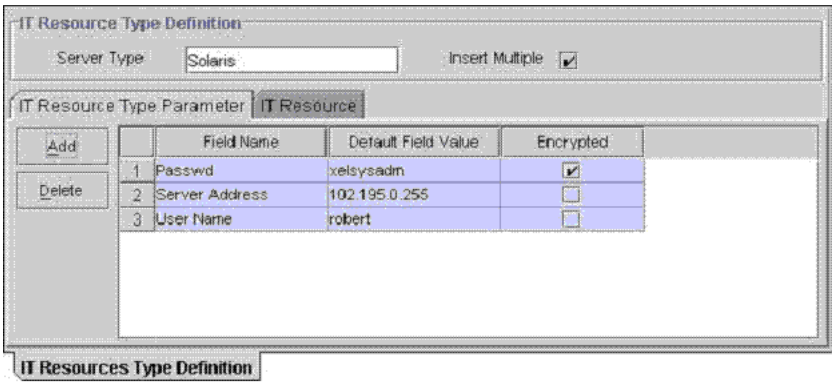


Table 4–1 describes the fields of the IT Resources Type Definition form.

Table 4–1 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*.

4.3.2.1 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.

4.3.2.2 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

4.3.2.2.1 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 4–1. 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.

4.3.2.2.2 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.

4.3.2.3 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

4.3.3 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.

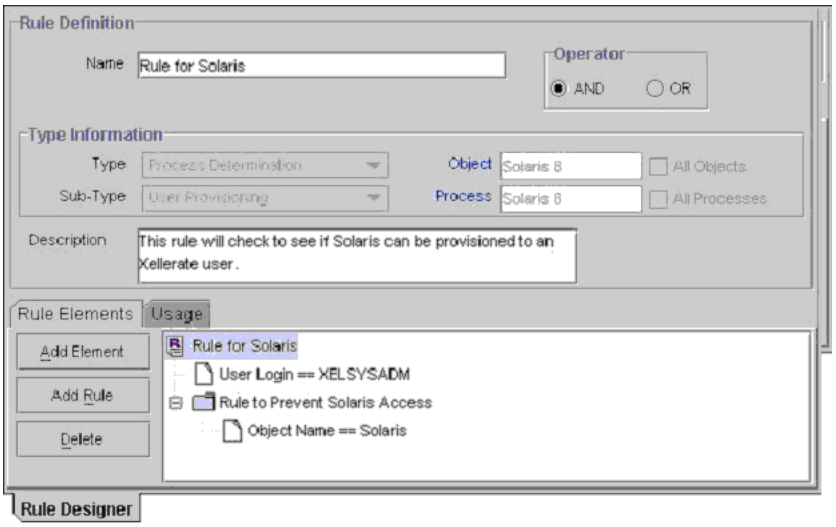
The following are examples of rule usage:

- 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 4–2 is in the Resource Management folder. You use this form to create and manage rules that are used with resources.

Figure 4–2 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 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 4–2, 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 4–2, 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 4–2, 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 4–2 describes the fields of the Rule Designer form.

Table 4–2 Fields of the Rule Designer Form

Field Name	Description
Name	The rule's name.

Table 4–2 (Cont.) Fields of the Rule Designer Form

Field Name	Description
AND/OR	These options specify the operation for the rule. 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**. **Important**: These options do not reflect the operations for rule elements that are contained within nested rules. In Figure 4–2, the **AND** operation applies to the `User Login == XELSYSADM` rule element and the `Rule to Prevent Solaris Access` nested rule. However, this operation has no effect on the `Object Name != Solaris` rule element within the `Rule to Prevent Solaris Access` rule.
Type	The rule's classification status. A rule can belong to one of four types: ■ **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	A rule of type Process Determination, Task Assignment, or Prepopulate can be categorized into one of four subtypes: ■ **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. 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.
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.

4.3.3.1 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 4–2 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** or **User Provisioning**) 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**.

4.3.3.2 Tabs on the Rule Designer Form

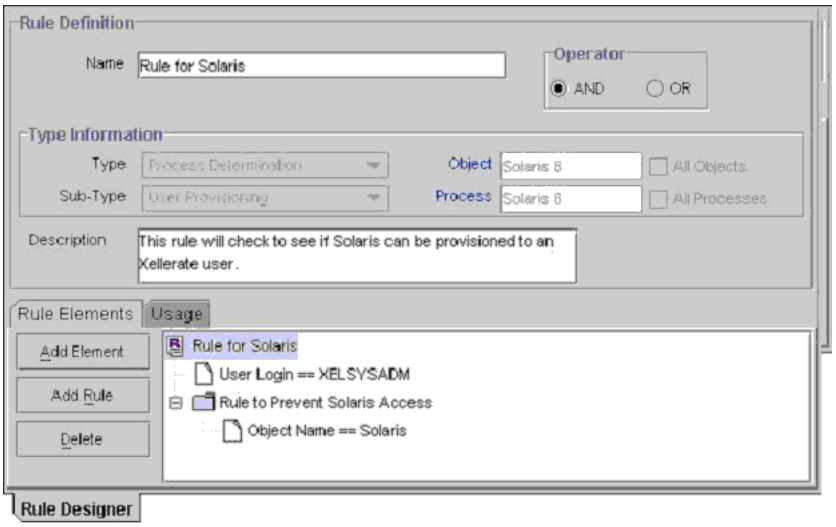
The Rule Designer form contains the following tabs:

- Rule Elements tab
- Usage tab

Each of these tabs is discussed in the following sections.

4.3.3.2.1 Rule Elements Tab From this tab, you can create and manage elements and nested rules for a rule. For example, in Figure 4–3, the `Rule for Solaris` contains the `User Login==XELSYSADM` rule element. It also has a nested `Rule to Prevent Solaris Access`. Figure 4–3 displays the Rule Elements tab of the Rule Designer form.

Figure 4–3 Rule Elements Tab of the Rule Designer Form



The rule in Figure 4–3 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 4–3 describes the data fields in the Edit Rule Element dialog box.

Table 4–3 Fields of the Edit Rule Element Dialog Box

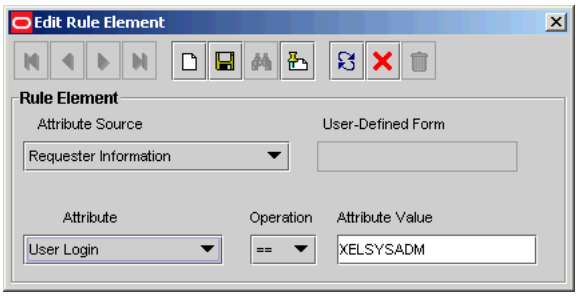
Name	Description
Attribute Source	From this box, select the source of the attribute. For example, if the attribute you wish to select is Object Name, the attribute source to select would be Object Information.
User-Defined Form	This field displays the user-created form that is associated with the attribute 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.

Table 4–3 (Cont.) Fields of the Edit Rule Element Dialog Box

Name	Description
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.

2. Set the parameters for the rule you are creating, as shown in Figure 4–4.

Figure 4–4 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 4-11

3. 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.

4. 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.

1. Click **Add Rule**.

   The Select Rule dialog box is displayed.

2. Select a nested rule and click **Save**.

3. 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:

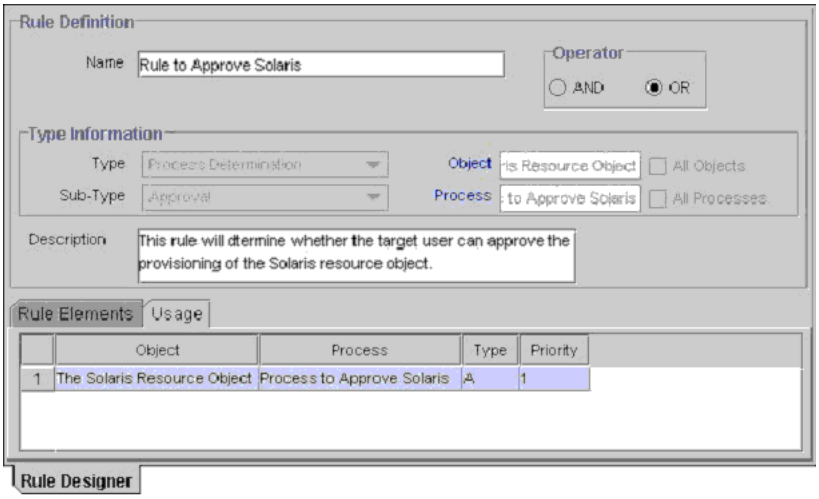
1. Select the rule element or nested rule that you want to remove.
2. Click **Delete**.

   The rule element or nested rule is removed from the rule.

4.3.3.2.2 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 4–5 shows the Usage tab.

Figure 4–5 Usage Tab of the Rule Designer Form



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.

4.3.3.3 Rule Designer Table

The Rule Designer Table, as shown in Figure 4–6, displays all available rules defined in the Rule Designer form.

Figure 4–6 Rule Designer Table

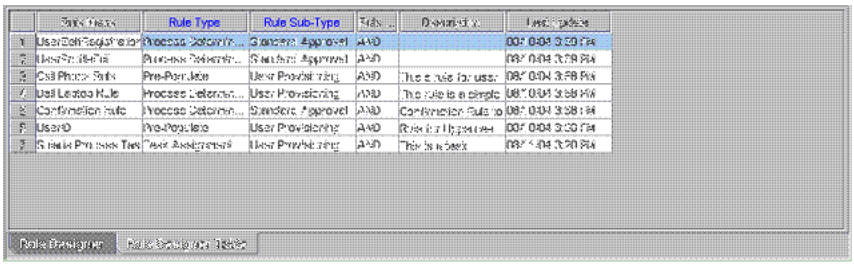


Table 4–4 shows the information displayed in the Rule Designer Table.

Table 4–4 Information in the Rule Designer Table

Field Name	Description
Rule Name	The name of the rule.
Rule Type	A rule can belong to one of four types: ■ **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	A rule of type Process Determination, Task Assignment, or Pre-Populate can be categorized into one of four sub-types: ■ **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.

4.3.4 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.

Figure 4–7 shows the Resource Objects form.

Figure 4–7 The Resource Objects Form

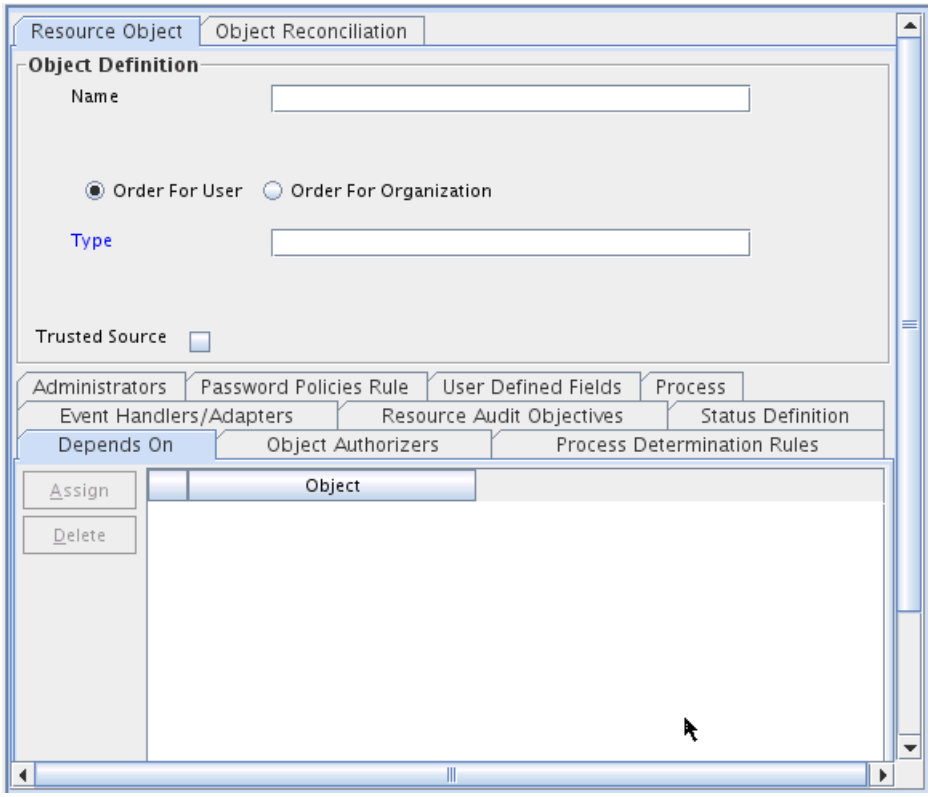


Table 4–5 describes the data fields of the Resource Objects form.

Table 4–5 Fields of the Resource Objects Form

Field Name	Description
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 the following types: ▪ **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. ▪ **Disconnected:** Classifies the resource object as a disconnected resource.

Table 4–5 (Cont.) Fields of the Resource Objects Form

Field Name	Description
Trusted Source	You can select this check box if you want to use the resource object for trusted user reconciliation. By default, this check box is not selected. It is selected by default only for the Xellerate User resource object.

4.3.4.1 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**, **System**, or **Disconnected**) to associate with the resource object.
5. If you want to use the resource object for trusted source user reconciliation, you must select the **Trusted Source** option. Otherwise, go to Step 6.
6. Click **Save**.

   The resource object is created.

4.3.4.2 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

4.3.4.2.1 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.

In addition, you must setup a parent-child relationship between the application instances as well. This is done by opening the application instance for the dependent resource and selecting the independent application instance as the parent from the drop-down for parent application instance. See "Managing Application Instances" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager* for information about the application instance UI.

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.

4.3.4.2.2 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 4-8 and "Assignment Tab of the Editing Task Window" on page 5-30 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.

4.3.4.2.3 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.

4.3.4.2.4 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.

4.3.4.2.5 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.

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)
- Quarterly Review

- 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.

4.3.4.2.6 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.

4.3.4.2.7 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.

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.

4.3.4.2.8 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.

4.3.4.2.9 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.

4.3.4.2.10 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.

4.3.4.2.11 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. 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

   – Number

   – 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 4–6.

Table 4–6 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 5-30 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.

4.3.4.3 Multiple Trusted Source Reconciliation

You can create 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.

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.

The attribute authoritative sources feature means that the 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 "Reconciliation Profile" on page 23-8 and "Updating Reconciliation Profiles Manually" on page 23-23.

4.3.4.3.1 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`

4.3.4.3.2 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 4-15.

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 5-5. 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.

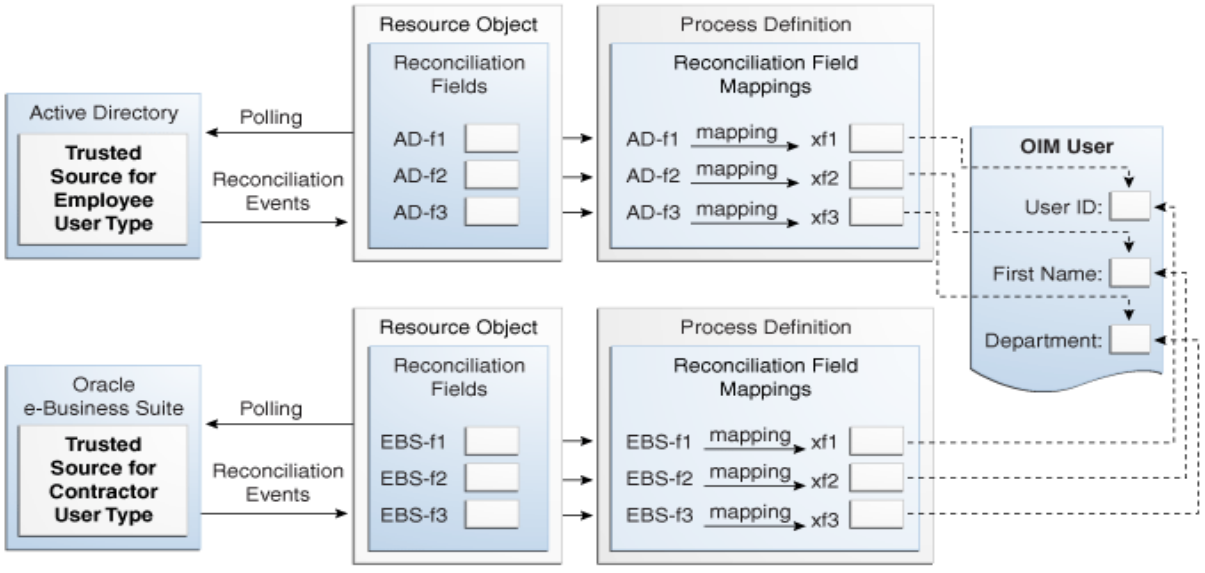
3. After you add the fields and the reconciliation rules, delete the Oracle e-Business Suite-specific fields and the corresponding rules created in the `Xellerate User` resource object.
4. Create the `EmpRecon` process definition in the Process Definition form. For detailed information about the procedure to create a process definition, see "Process Definition Form" on page 5-5. 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 4–8 shows the design time implementation of trusted source reconciliation based on the user type.

Figure 4–8 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 address` 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 `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 4-15.

   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 5-5. 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.

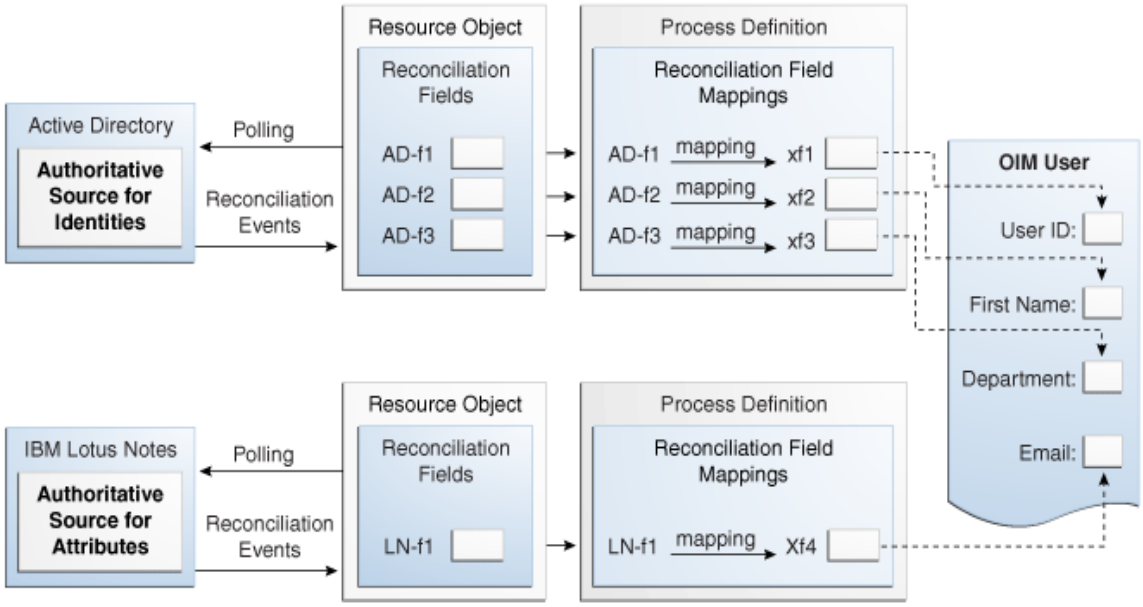
2. When you create the resource object, add only the `e-mail address` attribute.
3. On the Object Reconciliation>>Reconciliation Action Rules tab, create rule conditions and action mappings. Create any rule condition other than user creation if no matches are found. If a match is found, the link is established.
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 5-5. 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 4–9 shows the design time implementation of trusted source reconciliation of specific OIM User attributes.

Figure 4–9 Trusted Source Reconciliation for Specific OIM User Attributes



4.3.5 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.

4.4 Converting a Disconnected Application Instance to Connected Application Instance

To describe the procedure to convert a disconnected application instance to a connected application instance, the following assumptions have been made:

- A disconnected application instance exists in Oracle Identity Manager deployment, for example, the production environment. This disconnected application instance will be exported to another deployment of Oracle Identity Manager, for example, a test environment, and converted to a connected

application instance. After testing the connected application instance in the test environment, it will be imported in the production environment again.

Note: Optionally, the disconnected resource can be converted to a connected resource in the same environment. See "Modifying the Application Instance from Disconnected to Connected" on page 4-40 for further details.

- The application instance, process definition, forms, IT resource type definition, and IT resource retain the same name while converting a disconnected application instance to connected application instance.

The following are the broad-level steps to convert a disconnected application instance to a connected application instance:

- Import the existing disconnected resource from the existing environment to the test environment.
- Modify the implementation of the application instance, such as resource object definition and process definition.
- Test the application instance by provisioning it to users and validating the behavior for enable, disable, revoke, and update tasks.
- Export the new connected resource from the test environment and import it to the production environment.

Note:

- Only the resource is exported between environments and not the application instance.
- This section outlines the steps to import/export the resource of the application instance by using the Deployment Manager. Alternatively, the connector upgrade utility can also be used for import/export of the resource. See "Managing Connector Lifecycle" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager* for information about using the connector upgrade utility.

4.4.1 Creating a Disconnected Application Instance in the Production Environment

To create a disconnected application instance in the production environment:

1. Login to Oracle Identity System Administration.
2. Click **Sandboxes** to access sandbox management, create a sandbox, and activate it. See "Managing Sandboxes" on page 30-4 for information about sandboxes and how to create, activate, and publish sandboxes.
3. Under Configuration, click **Application Instances**. Click **Create** on the toolbar to open the Create Application Instance page.
4. Enter values in the Name and Display Name fields, such as LaptopAppInstance.
5. Select the **Disconnected** option to specify a disconnected application instance. Selecting the Disconnected option disables the Resource Object and IT Resource Instance fields in the page.

6. Click **Save**, and then click **OK** to confirm creation of the FinApp application instance. The artifacts for a disconnected application instance are created.
7. Go to the Manage Sandboxes page, and publish the sandbox.

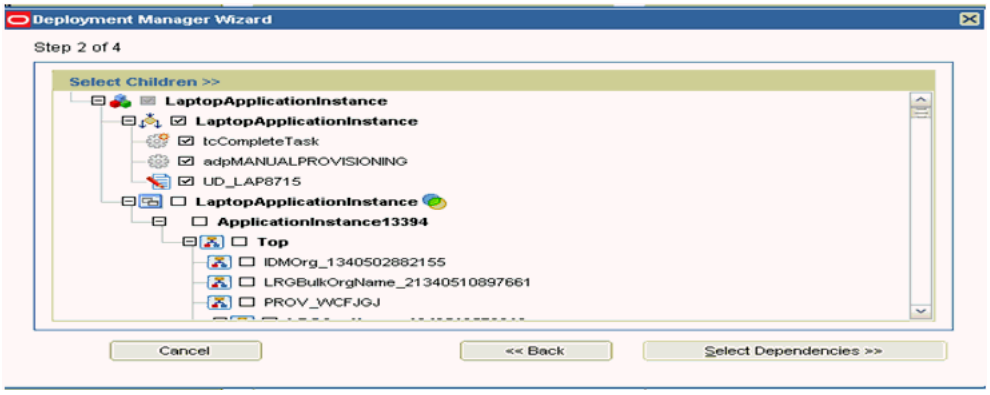
Upon successful creation of the application instance, organization and entitlements can be configured if necessary. For testing purpose, create four or five users and provision the newly created disconnected application instance to the users. Ensure that the users have the application instance in one of the following status: Provisioned, Enabled, Disabled, and Revoke. Try modifying one of the users to ensure that the account can be successfully updated.

4.4.2 Exporting Disconnected Application Instance From Production Environment

To export the disconnected application instance from the production environment:

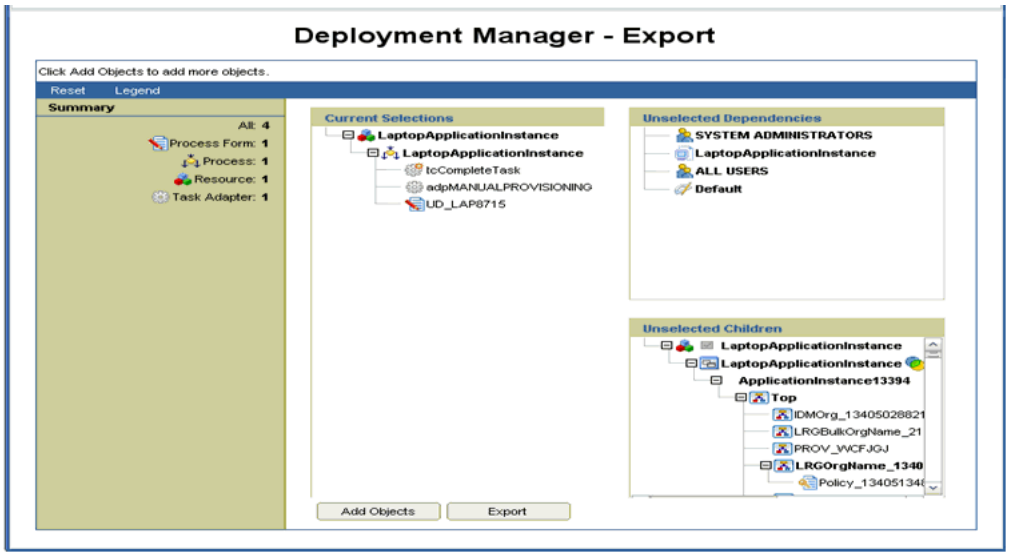
1. Login to Oracle Identity System Administration. In the left pane, under System Management, click **Export**. The Deployment Manager wizard is displayed in a new window.
2. Search for the disconnected application instance. To do so, in the search section, select Resource from the list, enter the name of the disconnected application instance, for example LaptopApplication*, and click **Search**. The disconnected application instance is displayed in the Search Results section.
3. Select **LaptopApplicationInstance** in the Search Results section, and then click **Select Children**. The Select Children page is displayed.
4. Select the required child attributes, as shown in Figure 4–10:

Figure 4–10 Child Attributes



5. Click **Select Dependencies**. The Select Dependencies page is displayed.
6. Click **Confirmation**. In the Confirmation page, click **Add For Export**.
7. After verifying that all the required dependencies are displayed in the export summary, as shown in Figure 4–11, click **Export**.

Figure 4–11 Export Summary



8. Provide a name to the XML file, such as DisconnectedLaptopExp.xml. Upon successful export, a message is displayed.

4.4.3 Importing the Disconnected Application Instance in Test Environment

To import the disconnected application instance in test environment:

1. In the left pane of the Oracle Identity System Administration, under System Management, click **Import**.
2. Provide the path to the exported XML file, and then click **OK**. A confirmation page is displayed. Click **Add File**.
3. In the Substitutions page, you can provide substitutions for users or groups. If there are no substitutions, then click **Cancel Substitution**.
4. In the import summary, as shown in , check for any unresolved dependency, and then click **Import**.

Figure 4–12 Import Summary

5. Verify that the process definition, resource object, and forms have been successfully imported.

4.4.4 Modifying the Application Instance from Disconnected to Connected

In the environment where the application instance has been imported, make the following changes to convert the disconnected application instance to a connected application instance:

1. Login to the Design Console.
2. Expand **Resource Management**. Click **Resource Objects** to open the Resource Objects form.
3. Change the type of the resource object from Disconnected to **Application**.
4. Define new IT resource parameters in conjunction with the connected resource as required in the IT Resource Type Definition form.
5. Modify the existing IT resource (assuming that the ITResource is the same) with the new parameters added in step 4.
6. Expand Process Management, and click **Process Definition** to open the Process Definition form.
7. Search the process definition of the disconnected application instance. The following tasks are displayed:
   - ManualProvisioningStart
   - ManualProvisioningEnd
   - ManualEnableStart
   - ManualEnableEnd
   - ManualDisableStart
   - ManualDisableEnd
   - ManualRevokeStart
   - ManualRevokeEnd
8. For each task, perform the following:
   a. Double-click the Task row to open the task details. See "Modifying Process Tasks" on page 5-15 for more information about modifying process tasks.
   b. Rename the task. For example, change the task name from ManualProvisioningStart to XXManualProvisioningStart.
   c. Make sure the **Conditional** option is selected. In addition, ensure that the **Required for Completion** option is not selected.
   d. If the task is an enable/disable/revoke task, then change the task effect to **No effect**.
   e. In the Integration tab, disassociate the adapters attached to the task by clicking on **Remove**.
   f. Remove task dependency, if any.
   g. Remove undo/recovery/generated tasks, if any.
   h. Change the object status mapping, if any, to none.

Note: Step 6a through 6g are to ensure that the existing tasks for disconnected application instance do not start when the application instance is exported as a connected application instance.

9. There is a task by the name *PARENT_FORM_NAME* Updated. This task triggers whenever the parent form is updated. Make sure to disassociate the existing adapters attached to the task and customize the task as required.
10. If there are any tasks related to the child form, then make sure to remove the triggers for create/update/delete by clicking **Clear**. If these tasks are not going to be reused, then disassociate the adapters attached to these tasks and rename the tasks to ensure that they do not run. Oracle recommends creating new tasks for each create, update, and delete trigger.

Note:

- Optionally, the same tasks for the child data can be retained but custom adapters must be defined for the create/update/delete trigger.
- For a disconnected application instance with child data, the task with the delete trigger will be associated with the tcCompleteTask adapter. Make sure to define and attach a custom adapter to this task to enable proper deletion of entitlement or child data.

11. Define custom adapters for the create, disable, enable, revoke, and update account tasks. If there are child tables, then make sure to define custom adapters for the same.
12. Create the following tasks in the process definition, and associate the corresponding adapters to each of those tasks. Map the required undo/recovery tasks and set the object status mapping.
    - **Create User:** Ensure that in the task properties, the **Required for Completion** option is selected and the **Conditional** option is not selected.
    - **Disable User:** Ensure that the task effect is Disable Processes or Access to Application.
    - **Enable User:** Ensure that the task effect is Enable Processes or Access to Application.
    - **Delete User:** Ensure that the task effect is Revoke Processes or Access to Application.
    - ***ATTRIBUTE_NAME* Updated:** For each attribute defined in the process form, corresponding update tasks have to be created. These tasks are triggered on updates to the process form, for example, Account Name Update, Account ID Updated, and so on.
13. If there is a child table, then define tasks for each trigger type, such as create, update, and delete.

Test the connected application instance by provisioning it to a few users in the test environment. You must define a new application instance with the modified resource object and IT resource to provision the application instance to users.

4.4.5 Testing the Connected Application Instance

After converting the disconnected application instance to a connected application instance:

- Export the modified resource from the test environment.
- Import the modified resource to the production environment.

5

Developing Provisioning Processes

This chapter describes process management with the Design Console. It contains the following topics:

5.1 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:

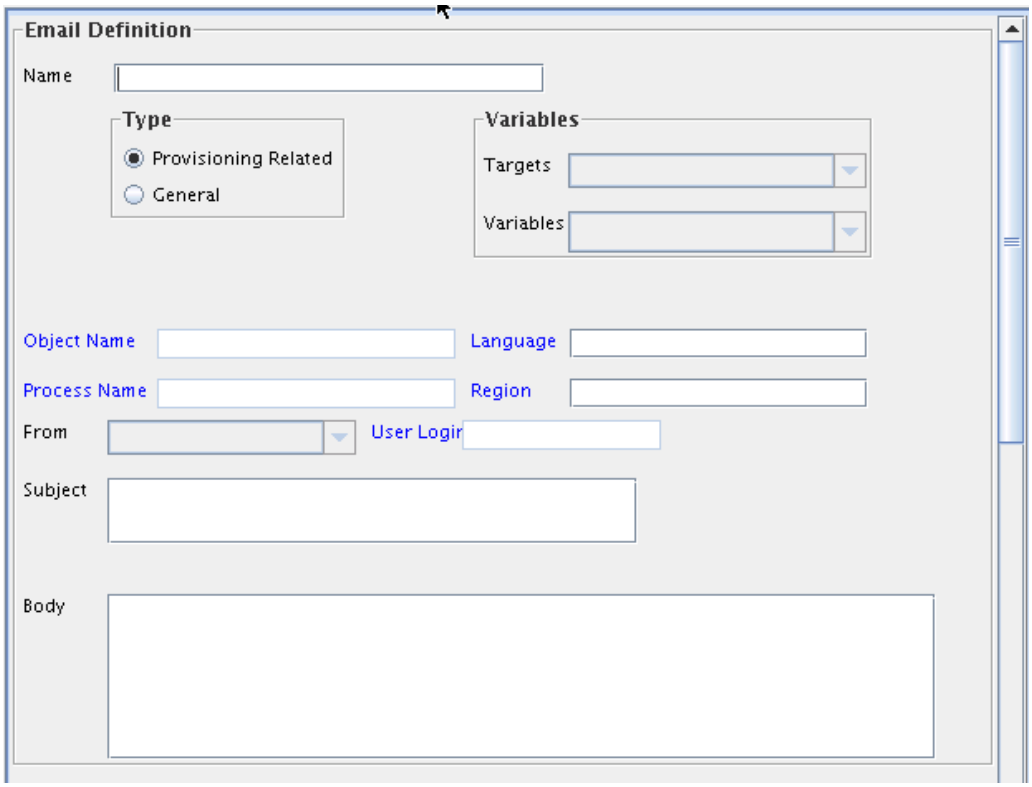
- **Email Definition**: This form enables you to create templates for e-mail notifications.
- **Process Definition**: This form lets you create and manage provisioning processes. It also lets you start the Workflow Definition Renderer that displays your workflow definition graphically.

5.2 Email Definition Form

The Email Definition form, as shown in Figure 5–1, 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 5–1 Email Definition Form



You apply e-mail definitions through the **Assignment** tab of the Process Definition form.

5.2.1 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.

The e-mail server is specified by using Oracle Identity System Administration. To specify the e-mail server:

1. Login to Oracle Identity System Administration.
2. Under **System Management**, 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.

When you click **Save**, a message box is displayed that states the following:

```
System Property has been modified. Please restart the server for changes to
take effect. Also make sure that IT Resource Definition for Mail Server is also
updated to reflect the changed value.
```

5. Click **OK**.

5.2.2 Email Definition Form

Table 5–1 describes the fields of the Email Definition form.

Table 5–1 Fields of the Email Definition Form

Field Name	Description
Name	The name of the e-mail definition.
Type	This region contains three options for the following: ■ 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 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. To categorize the e-mail definition as a general announcement, select the **General** option.
Object Name	From this lookup field, select the resource object that is associated with the provisioning process to which the e-mail definition is related. **Note**: Leave this lookup field empty to make the e-mail definition available for use with all resource objects.
Process Name	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. **Note**: If the **Provisioning Related** option is not selected, both the **Object Name** and **Process Name** lookup fields are grayed out.
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	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. **Note**: The items that are displayed in this box reflect the options you selected from the Type region.
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	Currently, two types of users can be selected from this box: ■ **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	The ID of the user in the From region of the e-mail notification. **Note**: If the User item is not displayed in the From box, the **User Login** field is grayed out.
Subject	The title of the e-mail definition.

Table 5–1 (Cont.) Fields of the Email Definition Form

Field Name	Description
Body	The content of the e-mail definition.

5.2.3 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. The Process Name field is grayed out until a value is selected in the Object Name field.

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	These options specify if a variable for the e-mail definition will be related to a provisioning process. To associate the e-mail variable with a provisioning process, select the **Provisioning Related** option.
Targets	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**.
Variables	From this box, select the variable for the e-mail definition, for example, **User Login**.

> **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** box.

14. Create an e-mail variable for the Subject field or Body text area.

15. Click **Save**.

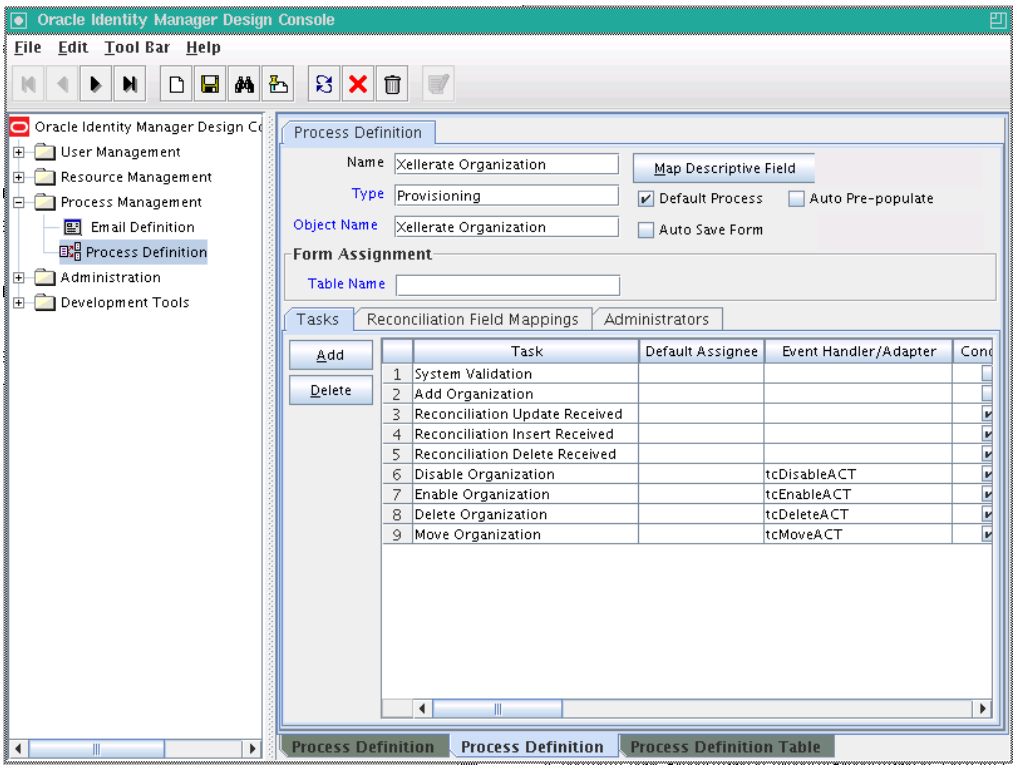
    The e-mail definition is created.

5.3 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 5–2 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 5–2 Process Definition Form



In Figure 5–2, 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 5–2; additional field columns extend on the right of the Tasks table.

Table 5–2 describes the fields of the Process Definition form.

Table 5–2 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.
Default Process	This check box determines if the current process is the default provisioning process for the resource object with which it is associated. 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.

Table 5–2 (Cont.) Fields of the Process Definition Form

Field Name	Description
Auto Save Form	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. 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.
Auto Pre-Populate	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: ■ Forms that are associated with the process ■ Forms that contain fields with prepopulated adapters attached to them 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. 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. **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. For more information about prepopulate adapters, see "Working with Prepopulate Adapters" on page 8-47. **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.
Table Name	The name of the table that represents the form that is associated with the process definition.

5.3.1 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 8, "Using the Adapter Factory".

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 an 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 8-47.

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.

   Note: 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.

5.3.2 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.

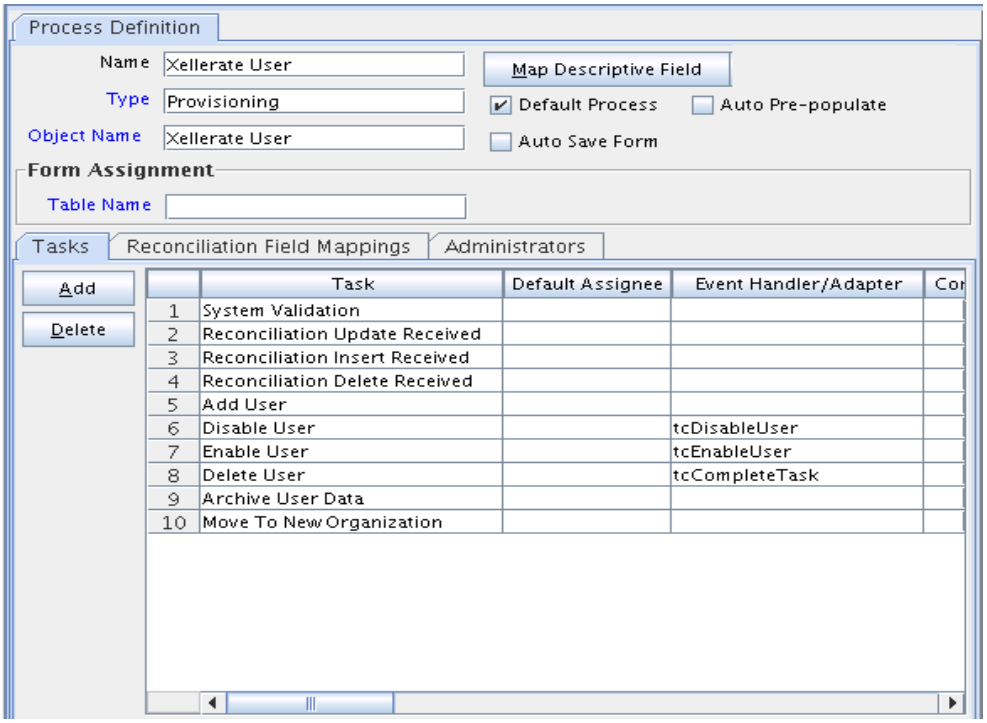
5.3.2.1 Tasks Tab

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 5–3 displays the Tasks tab of the Process Definition form.

Figure 5–3 Tasks Tab of the Process Definition Form



See Also: See "Modifying Process Tasks" on page 5-15 for information about editing process tasks

5.3.2.1.1 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.

5.3.2.1.2 Editing a Process Task

For instructions about how to edit and set process tasks, see "Modifying Process Tasks" on page 5-15.

5.3.2.1.3 Deleting a Process Task

To delete a process task:

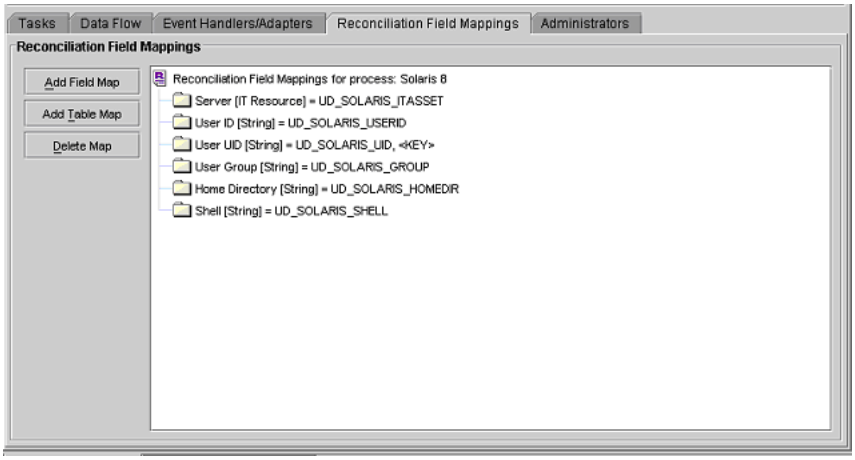
1. Select the process task that you want to delete.
2. Click **Delete**.

   The process task is removed from the process definition.

5.3.2.2 Reconciliation Field Mappings Tab

You use the Reconciliation Field Mappings tab shown in Figure 5–4 to define a relationship between data elements in a target system or trusted source and fields in Oracle Identity Manager.

Figure 5–4 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**.

  Note: The IT Resource must be marked as a key field.

- **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 4-28 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.

  Note: Oracle recommends configuring both the entitlement attribute and the key attribute for the child data in reconciliation field mappings to enable effective duplicate entitlement or child data validation. See "Duplicate Validation for Entitlements or Child Data" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager* for information about duplicate validation for entitlements or child data.

5.3.2.2.1 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

5.3.2.2.2 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.

5.3.2.2.3 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.

5.3.2.3 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.

5.3.2.3.1 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.

5.3.2.3.2 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.

5.3.3 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

Note: You must not modify the Xellerate Users process definition.

5.3.3.1 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 5–3 describes the fields of the General tab.

Table 5–3 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	This check box determines if a condition is met to add the current process task to the process. Select this check box to prevent the process task from being added to the process unless a condition has been met. Clear this check box to not require the condition to be met for the process task to be added to the process.
Required for Completion	This check box determines if the current process task must be completed for the process to be completed. Select this check box to require the process task to have a status of Completed before the process can be completed. Deselect this check box to ensure that the status of the process task does not affect the completion status of the process.
Constant Duration	Not applicable

Table 5–3 (Cont.) Fields of the General Tab of the Editing Task Dialog Box

Field Name	Description
Task Effect	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. **Note**: If you do not want the process task to be associated with a particular process action, select **No Effect** from the box.
Disable Manual Insert	This check box determines if a user can manually add the current process task to the process. Select this check box to prevent the process task from being added to the process manually. Deselect this check box to enable a user to add the process task to the process.
Allow Cancellation while Pending	This check box determines if the process task can be canceled if its status is Pending. Select this check box to allow the process task to be canceled if it has a Pending status. Deselecting this check box to prevent the process task from being canceled if its status is Pending.
Allow Multiple Instances	This check box determines if the process task can be inserted into the current process more than once. Select this check box to enable multiple instances of the process task to be added to the process. Deselect this check box to enable the process task to be added to the current process only once.
Retry Period in Minutes	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. 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. **Note:** If you specify a value for this field, then you must ensure the following: ■ The Task Timed Retry scheduled job is not disabled. See the "Predefined Scheduled Tasks" section in *Oracle Fusion Middleware Administrator's Guide* for more information. ■ Frequency of the Task Timed Retry scheduled job is less than or equal to value of this field. ■ The **Allow Multiple Instances** checkbox of the process task that is being retried must be selected.
Retry Count	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.

Table 5–3 (Cont.) Fields of the General Tab of the Editing Task Dialog Box

Field Name	Description
Child Table/ Trigger Type	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. 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 is to perform in the child table. These actions include: ■ **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.
Off-line	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.

5.3.3.1.1 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.
    - **Revoke Process or Access to Application**. When the resource is revoked, the revoke workflow is executed without canceling the existing tasks in the provisioning process.
    - **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.

5.3.3.1.2 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 5–3, " 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.

5.3.3.2 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`.
- From the Design Console, you cannot create or modify DOB event handlers. You can only view the existing event handlers.

See Also: Chapter 8, "Using the Adapter Factory" and Chapter 28, "Developing Event Handlers" for more information about adapters and event handlers

5.3.3.2.1 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 5-22 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 5–5.

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 5–5 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.

5.3.3.2.2 Mapping Adapter Variables

See Also: "Adapter Mapping Information" on page 8-56 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 *field* Updated, 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.

Note: IT Asset Type and IT Asset Property are displayed only when It Resources is selected from the Map To operations. The Literal Value field is displayed only when Literal is selected from Map To. Old Value check box is enabled only when Organization Definition or User Definition is selected from Map To.

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.

5.3.3.2.3 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.

5.3.3.3 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.

5.3.3.3.1 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.

5.3.3.3.2 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.

5.3.3.3.3 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.

5.3.3.3.4 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.

5.3.3.4 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.

5.3.3.4.1 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.

5.3.3.4.2 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.

5.3.3.4.3 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.

5.3.3.4.4 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.

5.3.3.5 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 `Cancelled` 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.

5.3.3.5.1 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.

5.3.3.5.2 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.

5.3.3.5.3 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.

5.3.3.5.4 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.

5.3.3.6 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 5-1 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.

5.3.3.6.1 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.

5.3.3.6.2 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.

5.3.3.7 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.

5.3.3.7.1 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.

5.3.3.7.2 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.

5.3.3.8 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 4-8. 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 5–4 describes the fields of the Assignment tab.

Table 5–4 Fields of the Assignment Tab of the Editing Task Window

Field Name	Description
Rule	The name of the Task Assignment rule to evaluate.
Target Type	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: ■ **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. ■ **Requestor's Manager:** The manager of the user who created the request. **Note**: Object Authorizer and Object Administrator roles are defined in the **Object Authorizers** and **Administrators** tabs, respectively, of the Resource Objects form.
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.

5.3.3.8.1 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), and Requestor's Manager 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.

5.3.3.8.2 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.

6

Developing Process Forms

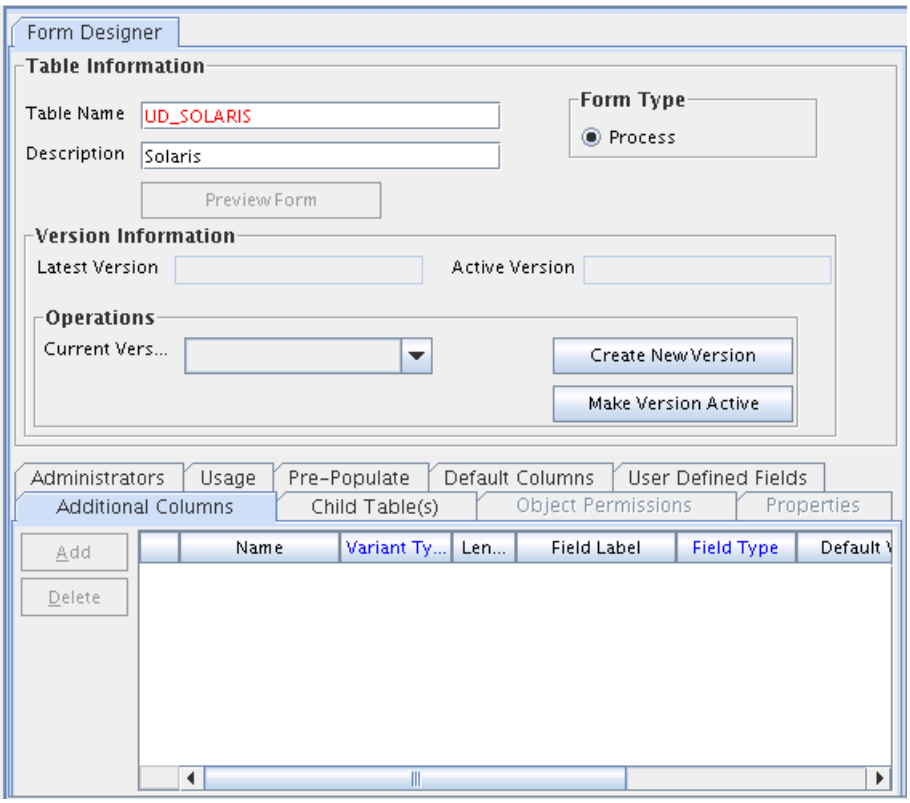
The information required to provision resources to a target user or organization cannot always be retrieved from an existing Oracle Identity Manager form. You can use the Form Designer form in the Development Tools folder of the Design Console to create a form with fields that contain the relevant information. 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 6–1 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 6–1 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 6–1 describes the data fields of the Form Designer form.

Table 6–1 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 6–1 (Cont.) Fields of the Form Designer Form

Field Name	Description
Form Type	These options are used to designate if the form is to be assigned to a process or a resource object. If you select the Process option, the form is associated with an approval or provisioning process.
Latest Version	The most recent version of the form.
Active Version	The version of the form that is used with the designated process or resource object. **Note**: After a version of the form is displayed in the Active Version field, it cannot be modified.
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	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. **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.
Make Version Active	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. **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).

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

6.1 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. Verify that the **Process** option is selected. This option is selected 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.

Note: When you create an application instance, associate a process form with it and publish the sandbox, and then make some changes to the process form in the Design Console, such as add new fields or prepopulate adapters, then the changes are not reflected in the application instance. To reflect the changes in the application instance, create a new process form and associate it with the application instance. The recommended method to do so is by using the Form Designer in Oracle Identity System Administration, and not the Form Designer form in the Design Console. See "Managing Forms" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager* for details.

6.2 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

6.2.1 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 6–2 describes the data fields of the Additional Columns tab.

Table 6–2 Fields of the Additional Columns Tab

Name	Description
Name	The name of the data field that 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 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**.

Table 6–2 (Cont.) Fields of the Additional Columns Tab

Name	Description
Variant Type	From this lookup field, select the variant type for the data field. The variant type denotes the type of data that the field accepts. This data field must be one of nine variant types: **Byte**, **Double**, **Date**, **Byte Array**, **Boolean**, **Long**, **String**, **Short**, and **Integer**.
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 dat field on the form that is generated by Oracle Identity Manager.

Table 6–2 (Cont.) Fields of the Additional Columns Tab

Name	Description
Field Type	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. You can select one of the following data types: ■ **TextField**: This data field is displayed on the generated form as a text field. The field type for TextField is displayed as `com.thortech.xl.gui.base.tcTextField` in lookup selection. 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 6-12. ■ **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. Oracle Identity Manager checks if a field with field type as Password Field has the `AccountPassword` property set to `true`. If so, then password policies apply. If such a field does not exist, then Oracle Identity Manager checks if a field of Password Field type having name in the *FORM_NAME*_PASSWORD format exists. If so, then password policies apply. Oracle Identity Manager automatically sets AccountPassword=true to a field with field type as Password Field and name in the *FORM_NAME*_PASSWORD format when such a field is being created. For information about setting the AccountPassword property to true, see "Setting the Value of the AccountPassword Property" on page 6-9. For information about adding a property value to a data field, see "Adding a Property and Property Value to a Data Field" on page 6-12. ■ **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 6–2 (Cont.) Fields of the Additional Columns Tab

Name	Description
Default Value	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: ■ A prepopulate 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 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	The Order field is not used in Oracle Identity Manager 11*g* Release 2 (11.1.2.2.0).
Application Profile	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. 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.
Encrypted	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. 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.

6.2.1.1 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, and the data is encrypted in the database.
>
> Do *not* use ParentAccountId as a form field name. ParentAccountId is used to store system information.

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 6–2 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 6–2 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**.

6.2.1.2 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 6.2.4.3, "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.

6.2.1.3 Setting the Value of the AccountPassword Property

To set the value of the AccountPassword property to true:

1. Create a password field in Form Designer form.
2. Click the **Properties** tab.
3. Select the password field, and click **Add Property**.
4. From the Property Name list, select **AccountPassword**.
5. In the Property Value field, enter `true`.
6. Click **Save**.

6.2.2 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 5.3, "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.

6.2.2.1 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.

6.2.2.2 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.

6.2.3 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).

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.

6.2.3.1 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.

6.2.3.2 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.

6.2.4 Properties Tab

You use the Properties Tab of the Form Designer Form to assign properties and property values to the data fields that are displayed on the form that is created through 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 A-1.

6.2.4.1 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.

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 6–3 lists the fields of the Add Property dialog box.

Table 6–3 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.

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 A-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.

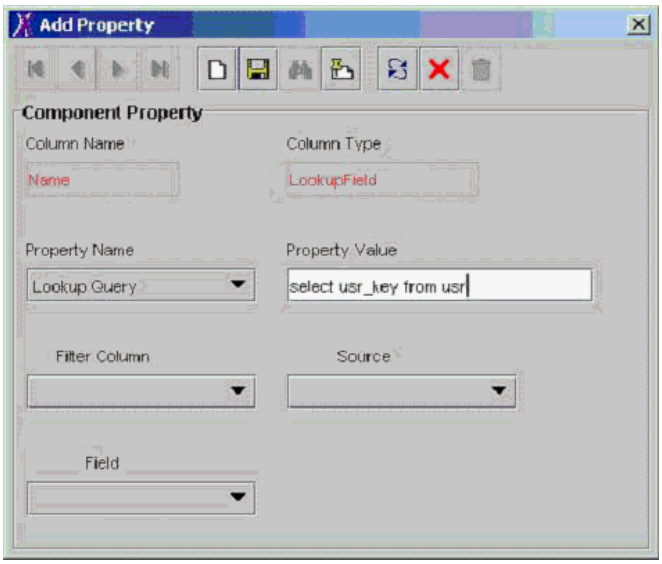
6.2.4.2 Adding a Property and Property Value for Customized Look up Query

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 6–2.

Figure 6–2 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 6–4 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 6–4 Fields of the Add Property Dialog Box

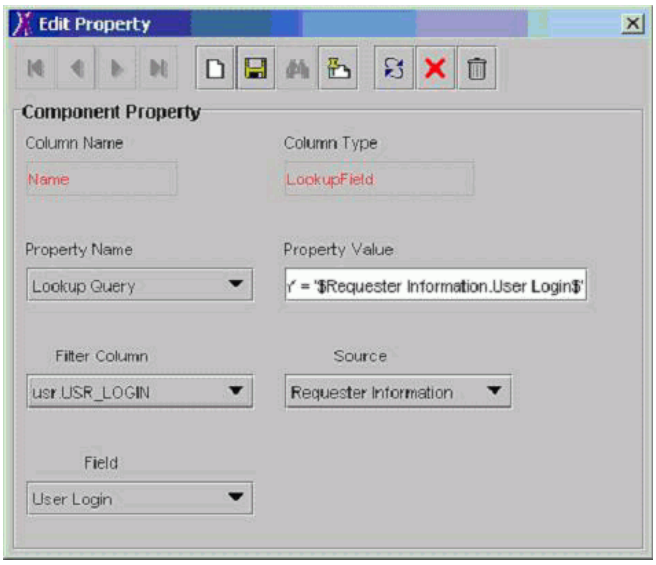
Name	Description
Column Name	The name of the data field.
Column Type	The data type of the data field.
Property Name	From this list, select the property for the data field.
Property Value	In this field, enter the property value, which is associated with the property that is displayed in the **Property Name** box. 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. 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.
Filter Column	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. For example, `usr.USR_LOGIN` signifies that Oracle Identity Manager will see the **User Login** field from the Users form for the lookup query.
Source	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. For example, **Requester Information** refers to the usr.USR portion of the **Filter Column** value.
Field	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. For example, the User Login corresponds to the **_LOGIN** part in the **Filter Column** value.

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 6–3 shows the Edit Property dialog box.

Figure 6–3 Edit Property Dialog Box



6.2.4.3 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.

6.2.5 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.

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.

6.2.5.1 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.

6.2.5.2 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.

6.2.6 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**.

6.2.7 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 8, "Using the Adapter Factory" 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

---

6.2.8 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.

6.2.9 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.

6.3 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.

If you use the new form version to manage all users, then run the Form Version Control (FVC) utility after the new form version is made active. See "Using the Form Version Control Utility" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager* for information about the FVC utility.

7

Managing Lookup Definitions

This section describes how to use the Design Console to administer Oracle Identity Manager. It contains the following topics:

7.1 Overview

The Design Console Administration folder provides system administrators with tools for managing Oracle Identity Manager administrative features. This folder contains the following form:

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.

Note: Oracle Identity Manager recommends that you create and manage lookups by using the Oracle Identity System Administration. See "Managing Lookups" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager* for details.

7.2 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 7–1 is in the Design Console Administration folder. You use this form to create and manage lookup definitions.

Figure 7–1 Lookup Definition Form

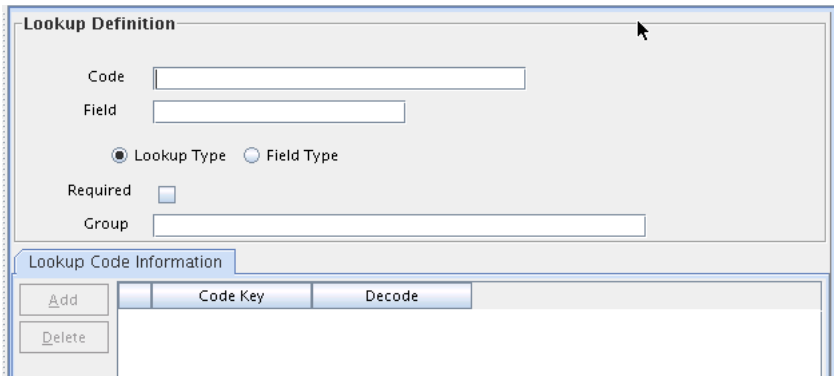


Table 7–1 describes the data fields of the Lookup Definition form.

Table 7–1 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	These options designate if the lookup definition is to represent a text field, a lookup field, or a box. If you select the **Field Type** option, the lookup definition will represent a text field. 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. **Note**: For forms or tabs that come packaged with Oracle Identity Manager, the lookup definition has already been set as either a lookup field *or* a box. This cannot be changed. However, you can add or modify the values that are accessible from the lookup field or box. 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. For more information about specifying the data type of a lookup definition, see "Additional Columns Tab".
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.

7.2.1 Creating a Lookup Definition

To create a lookup definition:

1. Login to Oracle Identity Manager Design Console.

2. Navigate to **Administration**, **Lookup Definition**.
3. In the **Code** field, enter the name of the lookup definition.
4. 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.
5. 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.
6. 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 7.
7. 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 7-1 for more information about naming conventions
8. 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.

7.2.2 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.

7.2.2.1 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.

7.2.2.2 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**, and then click **OK** in the message box. The selected lookup value is deleted.

7.2.3 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. Click **Add**.
5. In the Lookup Code Information tab, add questions by entering the appropriate values in the Code Key and Decode fields.
6. Add this key to the custom model bundle. For example the challenge question you want to add is "What is your dream destination?" Enter this in both Code Key and Decode fields, and save the Lookup Definition, as shown:
   a. Extracts the contents of the oracle.iam.ui.model.ear file, which is in the *OIM_ORACLE_HOME*/server/apps/ directory.
   b. Open the adflibCommonModel.jar file, which is in the /APP-INF/lib/ directory of the extracted EAR file.
   c. Navigate to the /oracle/iam/ui/common/model/ directory.
   d. Add new Key in the CommonModelBundle.properties and CommonModelBundle_en.properties files in the following format:

      ```
      KEY_WHAT_IS_YOUR_DREAM_DESTINATION?= What is your dream destination?
      ```

   e. Update the JAR file with the two new files, and update the EAR file with the updated JAR file.
   f. Redeploy the updated EAR file.
   g. Restart the Admin Server, SOA server, and Oracle Identity Manager server.

Part III

Connectors

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:

8

Using the Adapter Factory

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:

8.1 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 "Developing Process Forms" on page 6-1.

- 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 also allows you to create postprocessing handlers on entities, such as user, role, and organization.

8.2 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.

Note: In Oracle Identity Manager 11*g* Release 2 (11.1.2.2.0), creating new entity adapters and modifying existing entity adapters are not supported.

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 *Canceled* or *Unsuccessfully Completed*.
S	Suspended: This process/process task has been put on hold temporarily.
UC	Unsuccessfully Completed: This process task has been set to *Completed*. However, it had been rejected before.
W	Waiting: This process/process task cannot be completed until all preceding process tasks or processes are completed.
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*.

8.3 Adapter Environment and Tools

This section contains these topics:

8.3.1 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 36, "Understanding Customization Types" 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.

Note:

When the Java code is in two different JAR files in the Adapter Factory, and in the adapter tasks if an object from the first JAR file (which has the common or shared code) is passed into the constructor of the next adapter task that is located in the second JAR file, then a compilation error is thrown.

As a workaround for this issue, ensure that the entire Java code is in a single JAR file only.

8.3.2 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 and Access Management*.

The Connector Server also has the ability to run Action Scripts. See "Understanding the Identity Connector Framework" on page 9-1 for information about the Identity Connector Server.

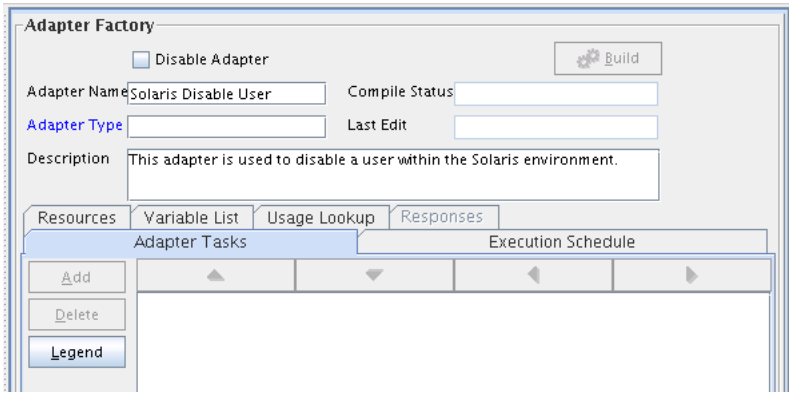
8.3.3 The Adapter Factory

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 8–1 shows the Adapter Factory Form in the Design Console.

Figure 8–1 Adapter Factory Form



8.3.4 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

8.3.4.1 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.

8.3.4.2 Compiling Adapters Manually

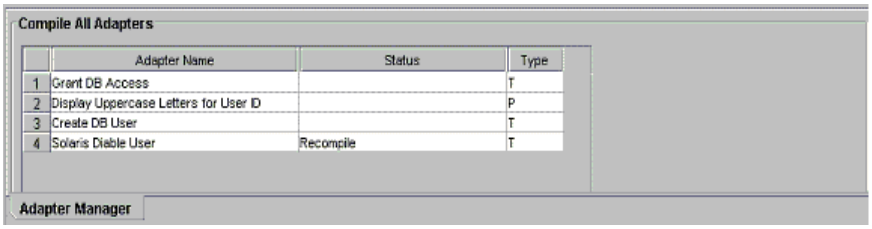
The Adapter Manager form is located in the Development Tools folder. You use it to compile multiple adapters simultaneously.

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 8–2.

Figure 8–2 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.

8.4 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: "Developing Process Forms" on page 6-1 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.

8.5 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

8.5.1 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.

8.5.2 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.

8.5.3 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.

8.5.4 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.

8.5.5 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.

8.5.6 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.

8.6 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.

8.7 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.

8.7.1 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 8–1.

Table 8–1 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.

8.7.2 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.

8.7.3 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.

8.8 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

8.8.1 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 8.8.9, "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

8.8.2 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 8–2 explains the options in the Object Instance Selection window.

Table 8–2 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 8–3 lists and describes the various regions of the Add an Adapter Factory Task window:

Table 8–3 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.
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 8.3.1, "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 8-56 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.

8.8.3 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 8.8.2, "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 8.8.2, "Creating a Java Task" for more information about the regions of the Add an Adapter Factory Task window

Section 8.3.1, "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 8-56 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.

8.8.4 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 8–4 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.

Table 8–4 (Cont.) Regions of the Add an Adapter Factory Task Window

Name	Description

Connection Status	Displays the status of the connection between Oracle Identity Manager and the database that contains the target stored procedure.
	
	When Oracle Identity Manager can connect to the database, Connection Established is displayed in the Connection Status region.
	
	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:
	
	`Exception Type:`
	`java.lang.ClassNotFoundExceptionMessage:`
	`java.lang.ClassNotFoundException:`
	`oracle.jdbc.driver.OracleDriver`
	
	In this example, Oracle Identity Manager could not connect to the designated database because it could not find a particular Java class.

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 8-56 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.

8.8.5 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 8-17 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 8-17 for more information about the regions of the Add an Adapter Factory Task window

"Configuring the Adapter Environment" on page 8-7

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.
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.

8.8.6 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 8-17 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 8-17 for more information about the regions of the Add an Adapter Factory Task window
   >
   > "Configuring the Adapter Environment" on page 8-7 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 `xlAPI.jar` file.

   > **Note:** `The xlAPI.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 8-56 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.

8.8.7 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 8-13 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 8–5 to understand the various types of operands.

Table 8–5 Types of Operands

Operand Name	Description
Variable	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. **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.
Adapter Task	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. **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.
Literal	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.

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.

8.8.8 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 "Defining Error Messages" on page 8-67.
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: "Defining Error Messages" on page 8-67 for information about creating error messages by using the Error Message Definition form

8.8.9 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 8–6.

Table 8–6 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
BREAK	X	
RETURN	X	
CONTINUE	X	

Table 8–7 explains the various regions of the Add Adapter Factory Logic Task Parameters window:

Table 8–7 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 8–8 to understand the various regions of this Add Adapter Factory Logic Task Parameters window.

Table 8–8 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 8.10, "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.

8.9 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.

8.10 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.

8.11 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.

8.12 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.

8.12.1 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.

8.12.2 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.

8.12.3 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.

8.13 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.

8.13.1 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.

Name	Description
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.

8.14 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

8.14.1 Mapping Rule Generator Adapter Variables

After creating a rule generator, 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: ■ 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: ■ 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. **Note**: The Map Adapters tab is grayed out until an adapter is assigned to the current data object.

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 (for example, *Solaris*). 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.

   For this example, the Solaris form has been selected. Its data object `Thor.CarrierBase.tcUD_SOLARIS` appears, along with the four rule generator adapters associated with it (`adpCONVERTTOLOWERCASE`, `adpSOLARISHMDSTRINGGEN`, `adpSETSOLARISASSET`, and `adpSETPASSWORDFROMMAIN`). Oracle Identity Manager will trigger these four rule generators on preinsert.

   Based on the sequence numbers of these adapters, Oracle Identity Manager will trigger the `adpCONVERTTOLOWERCASE` adapter first, followed by the `adpSOLARISHMDSTRINGGEN`, `adpSETSOLARISASSET`, and `adpSETPASSWORDFROMMAIN` adapters respectively.

   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 8-51 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 8–9 describes the various fields of the Data Mapping for Variable dialog box.

Table 8–9 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, Data).
Data Type	This field shows the data type of the adapter variable (for example, String is the data type for the Data adapter variable).
Map To	This field contains the source and target locations of the mappings you can set for the adapter variable (for example, User Definition). 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. If you are not mapping the adapter variable to a location or contact, 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, User Login).
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, *User Name*), 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.

Table 8–9 (Cont.) Fields of the Data Mapping for Variable Dialog Box

Field Name	Description
Literal Value	When you map the adapter variable to a literal, type the name of the specific literal in this field (for example, IBM). If you are not mapping the adapter variable to a literal, this field does not appear.

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

> **See Also:** "Adapter Mapping Information" on page 8-56 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.

8.14.2 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.

8.14.3 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.

8.15 Working with Entity Adapters

For information about working with entity adapters, see:

Note: In Oracle Identity Manager 11*g* Release 2 (11.1.2.2.0), creating new entity adapters and modifying existing entity adapters are not supported.

- Entity Adapters in "Types of Adapters" on page 8-3.
- The procedures in "Working with Rule Generator Adapters" on page 8-38 for details about mapping the variables of an entity adapter with Oracle Identity Manager forms and/or provisioning processes.

8.16 Working with Task Assignment Adapters

This section contains these topics:

8.16.1 Attaching Task Assignment Adapters to Process Tasks

After creating a task assignment adapter, 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 8-51 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 "Attaching Process Task Adapters to Process Tasks" on page 8-51.

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 8–10 lists the fields of the Edit Data Mapping for Variable dialog box is displayed.

Table 8–10 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, *String* is the data type for the UUID variable).

Table 8–10 (Cont.) Fields of the Edit Data Mapping for Variable Dialog Box

Field Name	Description
Map To	This field contains the types of mappings that you can set for the adapter variable (for example, IT Resources). 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, IT Asset).
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, Unique ID), 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 if the variable's data type is not String, this field does not appear. I**mportant**: 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.
Old Value	By selecting this check box, you map the adapter variable to the value that was originally in the selected Qualifier field before modification. 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. For example, if you want to disallow setting the password to the same value, 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, this check box is grayed out.

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 8-56 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.

8.16.2 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.

8.16.2.1 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.

8.17 Working with Prepopulate Adapters

This section contains these topics:

- Attaching Prepopulate Adapters to Form Fields
- Removing Prepopulate Adapters from Form Fields

8.17.1 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.
>
> If a process form has two IT resource fields, then the second IT resource must be populated using programmatic mechanism and prepopulate adapters. Two IT resources cannot be populated because the UI Form Designer does not support an IT resource type widget.

9. Click **Add**.

   The prepopulate Adapters dialog box is displayed.

   Table 8–11 lists and describes the fields of the prepopulate Adapters dialog box.

Table 8–11 Fields of the Edit Data Mapping for Variable 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.

Table 8–11 (Cont.) Fields of the Edit Data Mapping for Variable Dialog Box

Name	Description
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 8-51 for information about the various mapping statuses for an adapter.
Adapter Variables	This area displays the following: ▪ 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 8–12 describes the fields of the Map Adapter Variables window.

Table 8–12 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).

Table 8–12 (Cont.) Fields of the Map Adapter Variables WIndow

Field Name	Description
Data Type	This field shows the data type of the adapter variable (for example, *String* is the data type for the UserID adapter variable).
Map To	This field contains the types of mappings that you can set for the adapter variable (for example, Organization Definition and User Definition). **Note:** Process Data is not a valid source for mapping prepopulate adapter variables. 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. If you are not mapping the adapter variable to a location or contact, 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, 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, *User Name*), 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 8-56 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.

8.17.2 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.

8.18 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

8.18.1 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.

8.18.2 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 adp*SOLARISPASSWORDUPDATED* 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 8–13 describes the fields of the Data Mapping for Variable window.

Table 8–13 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, *SolarisUserID*).
Data Type	This field shows the data type of the adapter variable (for example, *String* is the data type for the *SolarisUserID* variable).
Map To	This field contains the types of mappings that you can set for the adapter variable (for example, *IT Resources*).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, *IT Asset*).
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.

Table 8–13 (Cont.) Fields of the Data Mapping for Variable WIndow

Field Name	Description
IT Asset Property	This field enables you to select a specific field that will receive the results of the mapping (for example, *User Name*), 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.
Old Value	By selecting this check box, you map the adapter variable to the value that was originally in the selected Qualifier field before modification. 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. 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.

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 8-56 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.

8.18.3 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.

8.18.3.1 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.

8.19 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

8.19.1 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

8.19.1.1 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	You can map the parameter to the adapter variables that you created for this adapter. **Note**: When the adapter variable's classification type is Object, it cannot be used with process task adapters. **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.

8.19.1.2 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.

8.19.1.3 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.

8.19.1.4 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.

8.19.1.5 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.

8.19.1.6 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.

8.19.1.7 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.

Map To Combo Box	Field Combo Box	Description
	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.
	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.

8.19.2 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

8.19.2.1 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.

Variable Type	Map To	Qualifier/Resource Type
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.

8.19.2.2 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.

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.
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.
	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.

Variable Type	Map To	Qualifier/Description
	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.
String	IT Resources	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. Note: If the data type of the adapter variable is not String, IT Resources cannot be selected from the Map To combo box.

8.19.2.3 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.
	IT Resource	You can map the parameter to an IT resource.
Object (Adapter Return Value)	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

Variable Type	Map To	Qualifier/Description
	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	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. **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. **Note**: If the data type of the adapter variable is Object, Literal cannot be selected from the Map To combo box.
String	IT Resources	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. IT Asset Type. You can map the parameter to an IT resource type.
String	IT Resources	IT Asset Property. You can map this parameter to one of the properties that comprise the selected IT resource type.

8.19.2.4 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	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. **Note**: If the data type of the adapter variable is Object, Literal cannot be selected from the Map To combo box.
	Entity Field	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.
	Organization Definition	The fields of the Organizations form to which you can map the adapter variable. **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.
	User Definition	The fields of the Users form to which you can map the adapter variable.

8.19.2.5 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	Process Data	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.
	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	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	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. **Note**: If the data type of the adapter variable is not String, then IT Resources cannot be selected from the Map To combo box.

8.20 Defining Error Messages

The Error Message Definition form, as shown in Figure 8–3, is in the Development Tools folder of the Design Console. 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 Identity Self Service or Identity System Administration 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 11*g* Release 2 (11.1.2.2.0) does not support creating new entity adapters.

Figure 8–3 Error Message Definition Form

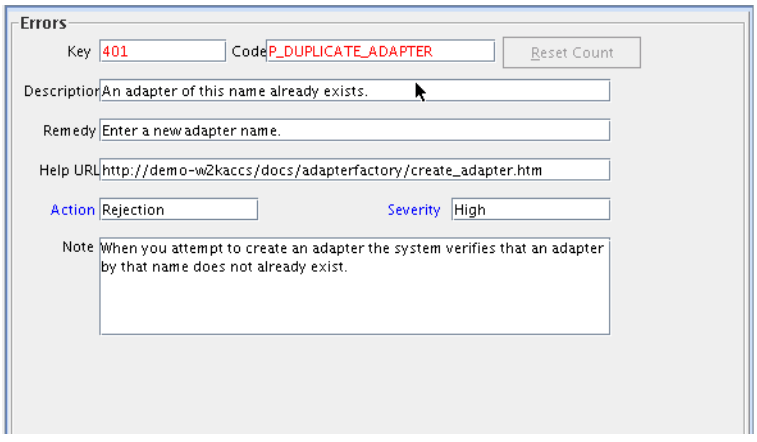


Table 8–14 describes the data fields of the Error Message Definition form.

Table 8–14 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.

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 displayed in Identity Self Service or Identity System Administration.

9

Understanding the Identity Connector Framework

Identity connectors are components developed to link Oracle Identity Manager with external stores of applications, directories, and databases. Oracle Identity Manager provides support for developing and building identity connectors by using the Identity Connector Framework (ICF). ICF decouples Oracle Identity Manager from 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:

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.

9.1 Advantages of ICF

ICF provides the following benefits:

- Single platform: Identity Connectors are shared between Oracle Identity Manager and Oracle Waveset (OW), which means they are built on top of the same platform so that a single connector can be used for both OIM and OW to communicate with external identity-aware applications.
- Simple installation: ICF offers simple installation as most of the manual configuration during installation, such as copying the connector files and the external code files are automatically taken care by ICF.
- Stateless by design: 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. This is 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.

- ICF Common: ICF provides common connector integration layer for all ICF based connectors in OIM and no development effort is required to develop ICF Common.
- Remote Execution: ICF supports remote execution of connector server using Java or .NET implementation.
- JVM Isolation: Remote ICF provides JVM isolation, which means running a Java connector on a different host avoids JVM conflicts.
- Reuse: In future, other products can reuse Identity Connectors.

9.2 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 9–1 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 9-21.

Figure 9–1 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 9–1 Identity Connector Framework Deployment

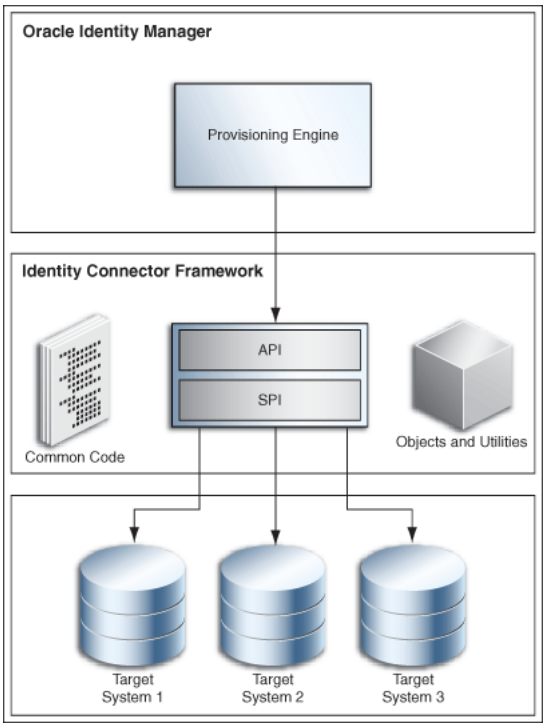


Figure 9–2 illustrates the backwards compatibility of the ICF. Newer bundles may be deployed without affecting existing ones. In addition, newer versions of the ICF are generally backward-compatible with existing bundles. Therefore, any connector should work with a new version of framework.

Figure 9–2 Compatibility Between the ICF and Connector Bundles

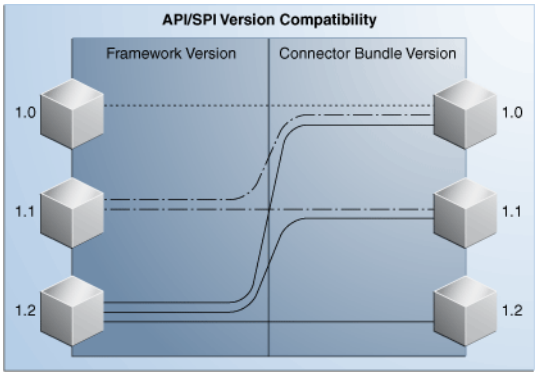


Figure 9–3 illustrates deployment methodology of the ICF. Framework supports LCM to clone connector to support multiple versions of the same target. In addition, Framework supports connection pooling.

Figure 9–3 Deployment Methodology to Support Multiple Versions of Same Target

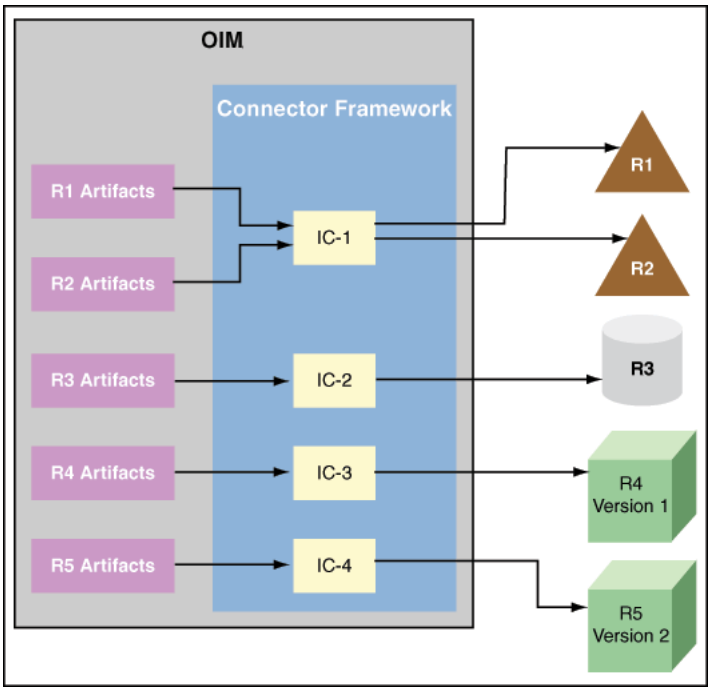


Figure 9–4 illustrates Framework installed on remote system. This enables remote execution of connector server using Java or .NET implementation with targets being local or remote to connector bundles. This is required when a connector bundle is not directly executed with in an application and ICF allows the application to communicate with externally deployed bundles. In addition, the connector artifacts can be same for local or remote system.

Figure 9–4 Connector Server Remote System Framework

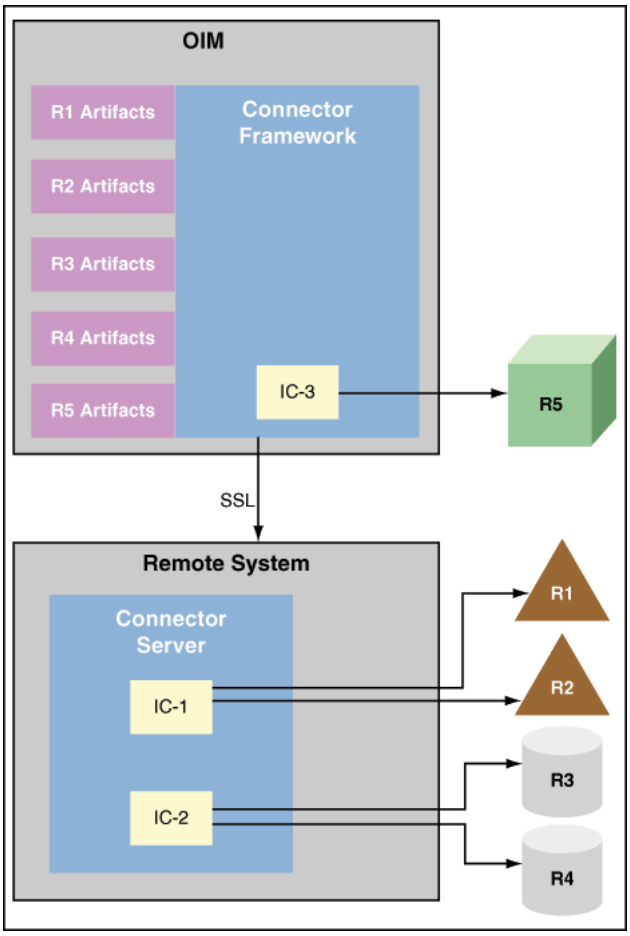
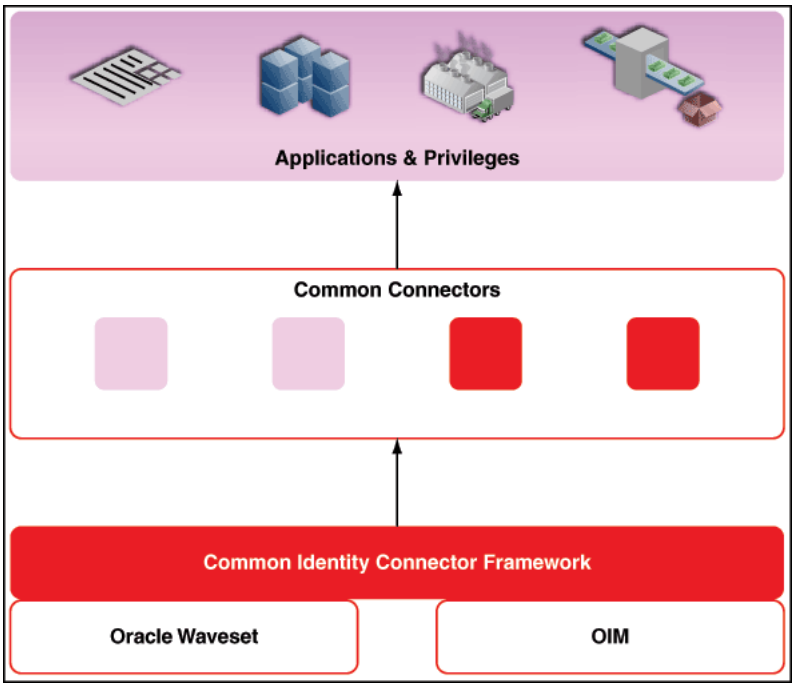


Figure 9–5 illustrates ICF Framework, which enables the convergence of Oracle Identity Manager and Oracle Waveset (OW) connectors to a single connector, best of both.

Figure 9–5 ICF Framework



9.3 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

9.3.1 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 9.3.2, "The ConnectorInfoManager Interface" for more information.) Example 9–1 illustrates the ConnectorInfoManagerFactory implementation.

Example 9–1 ConnectorInfoManagerFactory Implementation

```
//create ConnectorInfoManagerFactory
```

```
ConnectorInfoManagerFactory cInfoManagerFactory =
    ConnectorInfoManagerFactory.getInstance();
```

9.3.2 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 by obtained by calling getRemoteManager method on the ConnectorInfoManagerFactory. The getRemoteManager method accepts an instance of RemoteFrameworkConnectionInfoand, which is used for getting information about connectors deployed on Connector Server.

In Example 9–2, 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 9–2 ConnectorInfoManager Implementation

```
//get the ConnectorInfoManager
ConnectorInfoManager cInfoManager =
    cInfoManagerFactory.getLocalManager(bundleURL);
```

9.3.3 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 9–3.

Example 9–3 ConnectorKey Implementation

```
//get the ConnectorKey reference
ConnectorKey flatFileConnectorKey =
    new ConnectorKey(bundleName, bundleVersion, connectorName);
```

9.3.4 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 9–4 illustrates how to implement the ConnectorInfo.

Example 9–4 ConnectorInfo Implementation

```
//get the ConnectorInfo
ConnectorInfo info =
    cInfoManager.findConnectorInfo(flatFileConnectorKey);
```

In the example, cInfoManager is the ConnectorInfoManager and flatFileConnectorKey is the identity connector key.

9.3.5 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 9–5 illustrates this.

Example 9–5 APIConfiguration Definition

```
APIConfiguration apiConfig =
    info.createDefaultAPIConfiguration();
```

9.3.6 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 9–6.

Example 9–6 setPropertyValue Method Signature

```
public void setPropertyValue
  (java.lang.String name, java.lang.Object value)
```

Example 9–7 illustrates an implementation of the ConfigurationProperties interface.

Example 9–7 ConfigurationProperties Implementation

```
//get the default APIConfiguration
ConfigurationProperties flatFileConfigProps =
    apiConfig.getConfigurationProperties();
```

9.3.7 The ConnectorFacadeFactory Class

The ConnectorFacadeFactory class allows an application to get a Connector instance and to manage a pool of Connector instances. Example 9–8 illustrates the ConnectorFacadeFactory definition.

Example 9–8 ConnectorFacadeFactory Definition

```
//get a reference to ConnectorFacadeFactory
ConnectorFacadeFactory facadeFactory =
    ConnectorFacadeFactory.getinstance();
```

9.3.8 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 9–9 illustrates the ConnectorFacade implementation.

Example 9–9 ConnectorFacade Implementation

```
//create a ConnectorFacade (nothing but a reference to Connector on SPI side)
ConnectorFacade connectorFacade = facadeFactory.newInstance(apiConfig)
```

9.4 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

9.4.1 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

9.4.1.1 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 9-12 for more information on how to implement the configuration class.
- **displayNameKey:** Display name key that must be present in the message catalog.

Example 9–10 is a sample connector implementation.

Example 9–10 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 9–10:

- **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 9-15 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 10-1.

- Implementing the init Method
- Implementing the dispose Method
- Implementing the getConfiguration Method

9.4.1.1.1 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 9–11 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 9–11 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 10-1.

The init method implementation shown in Example 9–11 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.

9.4.1.1.2 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 9–12 illustrates how to implement the `dispose` method.

Example 9–12 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
 }
```

9.4.1.1.3 Implementing the getConfiguration Method

The getConfiguration method returns the Configuration instance passed to the Connector when the init method was used. Example 9–13 illustrates how to implement the getConfiguration method.

Example 9–13 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().

9.4.1.2 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 9–14 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

9.4.1.2.1 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 9–15 illustrates how to implement the validate method.

Example 9–15 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 10-1.

9.4.1.2.2 The setConnectorMessages() Method

The setConnectorMessages method sets the org.identityconnectors.framework.common.objects.ConnectorMessages message catalog instance, allowing the Connector to localize messages. Example 9–16 illustrates the setConnectorMessages method definition.

Example 9–16 setConnectorMessages Method Definition

```
public final void setConnectorMessages(ConnectorMessages messages) {
_connectorMessages = messages;
}
```

9.4.1.2.3 The getConnectorMessages() Method

The getConnectorMessages method returns the ConnectorMessages set by the setConnectorMessages method. Example 9–17 illustrates the getConnectorMessages method definition.

Example 9–17 getConnectorMessages Method Definition

```
public final ConnectorMessages getConnectorMessages() {
return _connectorMessages;
}
```

9.4.2 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

9.4.2.1 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 9–18 is a sample flat file PoolableConnector implementation.

Example 9–18 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 9.4.1.1, "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 9–19 illustrates how to implement the checkAlive method.

Example 9–19 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
}
```

9.4.2.2 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 9–20 illustrates the normalizeAttribute method definition.

Example 9–20 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()));
}
}
```

9.4.3 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

9.4.3.1 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 9–21.

Example 9–21 schema Method Signature

```
public Schema schema
```

The implementation should return the schema containing the types of objects that this identity connector supports.

Example 9–22 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.

9.4.3.2 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 9–23.

Example 9–23 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 9–24.

Example 9–24 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.

9.4.3.3 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 9–25.

Example 9–25 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 9–26 illustrates how to implement the delete method.

Example 9–26 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.

9.4.3.4 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

9.4.3.4.1 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 9–27 illustrates the createFilterTranslator method definition.

Example 9–27 createFilterTranslator Method Signature

```
public FilterTranslator createFilterTranslator
   (ObjectClass oClass, OperationsOptions options)
```

> **Note:** The return value should not be null.

Example 9–28 illustrates an implementation of the createFilterTranslator method.

Example 9–28 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 10-8 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 9-21.

9.4.3.4.2 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 9-19). 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 9–29.

Example 9–29 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 9–30 illustrates how to implement the exectueQuery method.

Example 9–30 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;
            }
        }
    }
```

9.4.3.5 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 9–31.

Example 9–31 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 9–32 illustrates how to implement the update method.

Example 9–32 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);
    }
```

9.4.4 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 FilterTranslater 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 9.4.3.4, "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 9.4.3.4, "Implementing the SearchOp Interface."

9.5 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 AbtractDatabaseConnector 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)

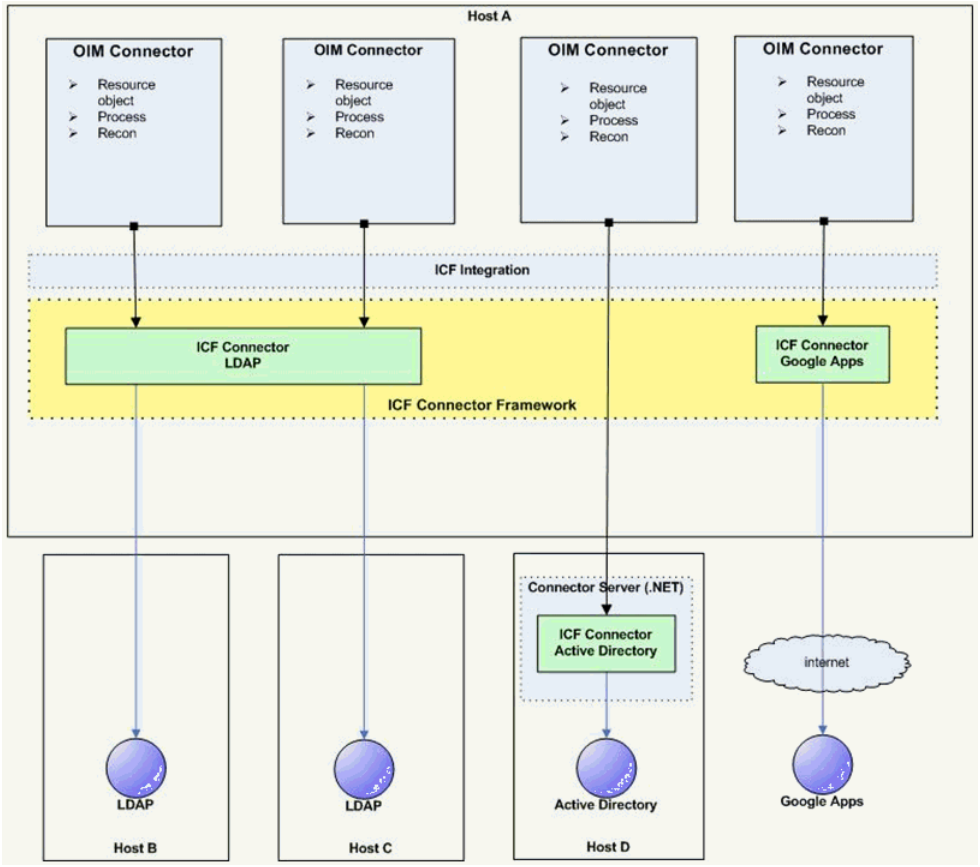
9.6 Using an Identity Connector Server

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 9–6 shows how Oracle Identity Manager connectors integrate with resources via ICF connectors:

Figure 9–6 ICF Connectors and Connector Server



In Figure 9–6:

- 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 9–6, 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

9.6.1 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

9.6.1.1 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 9–1 lists the properties in the ConnectorServer.properties file:

Table 9–1 Properties in the ConnectorServer.properties File

Property	Description
connectorserver.port	This is a common property for denoting both SSL and non-SSL connector server port. If the **connectorserver.usessl** property is set to `true`, then the connector server listens on a secure channel using the same port. The default SSL and non-SSL port number is `8759`. To change the default connector server port (SSL or non-SSL), update the connectorserver.port property with the new port number in the ConnectorServer.properties file.
connectorserver.bundleDir	Directory where the connector bundles are deployed. The default value is `bundles`.

Table 9–1 (Cont.) Properties in the ConnectorServer.properties File

Property	Description
connectorserver.libDir	Directory in which to place dependent libraries. The default value is `lib`.
connectorserver.usessl	If set to true, the Java Connector Server uses SSL for secure communication. The default value is `false`. If you specify true, then use the following options on the command line when you start the Java Connector Server: ■ `-Djavax.net.ssl.keyStore` ■ `-Djavax.net.ssl.keyStoreType` (optional) ■ `-Djavax.net.ssl.keyStorePassword`
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 9-27 and "Running the Java Connector Server on Solaris and Linux" on page 9-28 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.

9.6.1.2 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 9-26.
2. Change to the *CONNECTOR_SERVER_HOME*\bin directory and find the ConnectorServer.bat script.

   Table 9–2 lists the options supported by the ConnectorServer.bat script:

Table 9–2 Options Supported by the ConnectorServer.bat Script

Option	Description
/install [serviceName] ["-J java-option"]	Installs the Java Connector Server as a Microsoft Windows service. Optionally, you can specify a service name and Java options. If you do not specify a service name, then the default name is ConnectorServerJava.

Table 9–2 (Cont.) Options Supported by the ConnectorServer.bat Script

Option	Description
/run ["-J java-option"]	Runs the Java Connector Server from the console. Optionally, you can specify Java options. For example, to run the Java Connector Server with SSL: `ConnectorServer.bat /run "-J-Djavax.net.ssl.keyStore=mykeystore.jks" "-J-Djavax.net.ssl.keyStorePassword=password"`
/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]	Uninstall the Java Connector Server. If you do not specify a service name, then the script uninstall the ConnectorServerJava service.

3. If you need to stop the Java Connector Server, then stop the respective Microsoft Windows service.

9.6.1.3 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 9-26.
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 9–3 lists the options supported by the connectorserver.sh script:

Table 9–3 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: `./connectorserver.sh /run -J-Djavax.net.ssl.keyStore=mykeystore.jks -J-Djavax.net.ssl.keyStorePassword=password`
/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.

Table 9–3 (Cont.) Options Supported by the connectorserver.sh Script

Option	Description
/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.

9.6.1.4 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.

9.6.1.5 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.

9.6.2 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

9.6.2.1 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.

9.6.2.2 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.certificatestorename"
value="ConnectorServerSSLCertificate" />
<add key="connectorserver.ifaddress" value="0.0.0.0" />
```

The connectorserver.port property is a common key for denoting both SSL and non-SSL connector server port. If the **connectorserver.usessl** property is set to `true`, then the connector server listens on a secure channel using the same port. The default SSL and non-SSL port number is `8759`. To change the default connector server port (SSL or non-SSL), update the connectorserver.port property with the new port number in the connectorserver.exe.config file.

To use SSL, set the value of connectorserver.usessl to true, and set the value of connectorserver.certifacatestorename 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 9.6.1.5, "Using SSL to Communicate with a Connector Server." For information on trace setting configurations, see Section 9.6.2.3, "Configuring Trace Settings."

9.6.2.3 Configuring Trace Settings

The Connector Server uses the standard .NET trace mechanism. Trace settings are defined in the connectorserver.exe.config configuration file. Example 9–33 illustrates how they are defined.

Example 9–33 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.

9.6.2.4 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.

9.6.2.5 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.

10

Developing Identity Connectors Using Java

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:

10.1 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 10-9 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 10–1 is a sample of this. See Section 9.4.1.2, "org.identityconnectors.framework.spi.Configuration" for more information.

Example 10–1 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;
/*
 * Delimeter used
 */
private String textFieldDelimeter;
/*
 * Unique attribute field name
 */
private String uniqueAttributeName = "";
/*
 * Change attribute field name. Should be numeric
 */
private String changeLogAttributeName = "";

public File getStoreFile() {
return storeFile;
}

public String getTextFieldDelimeter() {
return textFieldDelimeter;
}

     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 delimeter
 * @param textFieldDelimeter
 */
@ConfigurationProperty(order = 2,
        helpMessageKey = "USER_STORE_TEXT_DELIM_HELP",
        displayMessageKey = "USER_STORE_TEXT_DELIM_DISPLAY")
public void setTextFieldDelimeter(String textFieldDelimeter) {
    this.textFieldDelimeter = textFieldDelimeter;
}

/**
 * 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 10–2 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 10–2 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 10–3 illustrates the sample implementation of org.identityconnectors.framework.common.objects.filter.AbstractFilterTranslator<T> that defines the filter operation.

Example 10–3 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 10–4 shows the contents of the MANIFEST.MF file:

Example 10–4 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 entries listed in step 4.

10.1.1 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 10–5, "FlatFileIOFactory"
- Example 10–6, "FlatFileMetadata"
- Example 10–7, "FlatFileParser"
- Example 10–8, "FlatFileWriter"
- Example 10–9, "FlatfileLineIterator"
- Example 10–10, "FlatfileUserAccount"
- Example 10–11, "FlatfileAccountConversionHandler"
- Example 10–12, "Messages.Properties"

Example 10–5 shows the implementation of the FlatFileIOFactory supporting class:

Example 10–5 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 10–6 shows the implementation of the FlatFileMetaData supporting class:

Example 10–6 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 uniqueAttributeFiledName;

    /**
     * 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.getTextFieldDelimeter());

            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 10–7 shows the implementation of the FlatFileParser supporting class:

Example 10–7 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 10–8 shows the implementation of the FlatFileWriter supporting class:

Example 10–8 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 accountUid = accountRecord.getAttributeValue(fileConfig
                .getUniqueAttributeName());
        FlatFileUserAccount accountByAccountId = recordParser
                .getAccount(accountUid);

        if (accountByAccountId != null)
            throw new RuntimeException("Account " + accountUid
                    + " 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");
        }
    }
}
```

Example 10–9 FlatfileLineIterator

```
package org.identityconnectors.flatfile.io;
 .
 import java.io.BufferedReader;
 import java.io.File;
 import java.io.FileReader;
 import java.io.IOException;
 import java.util.Iterator;
 import java.util.Map;
 .
 import org.identityconnectors.flatfile.FlatFileConfiguration;
 import org.identityconnectors.flatfile.FlatFileUserAccount;
 import org.identityconnectors.flatfile.utils.AccountConversionHandler;
 .
 /**
  * Iterator class to fetch the records dynamically during search operations
 This
  * is needed to prevent VM overloading when all records are stored in memory
  *
  * @author admin
  *
  */
 public class FlatFileLineIterator implements Iterator<FlatFileUserAccount> {
 .
     private File recordsStore;
     private AccountConversionHandler accountConverter;
     private FlatFileUserAccount nextRecord;
     private BufferedReader userRecordReader;
     private Map<String, String> attrConstraints;
 .
     /**
      * Making it package private to prevent global initialization
      *
      * @param metadata
      * @param fileConfig
      * @param attributeValConstraints
      *            Iterator will apply this constraint and filter the result
      */
```

```
    FlatFileLineIterator(FlatFileMetadata metadata,
            FlatFileConfiguration fileConfig,
            Map<String, String> attributeValConstraints) {
        this.recordsStore = fileConfig.getStoreFile();
        this.accountConverter = new AccountConversionHandler(metadata,
                fileConfig);
        this.attrConstraints = attributeValConstraints;
.
        initializeReader();
        this.nextRecord = readNextValidRecord();
    }
.
    private void initializeReader() {
        try {
            userRecordReader = new BufferedReader(new FileReader(recordsStore
                    .getAbsolutePath()));
.
            // Skip headers
            userRecordReader.readLine();
.
        } catch (IOException io) {
            throw new IllegalStateException("Unable to read "
                    + recordsStore.getName());
        }
    }
.
    @Override
    public boolean hasNext() {
        return (nextRecord != null);
    }
.
    @Override
    public FlatFileUserAccount next() {
        FlatFileUserAccount currentRecord = this.nextRecord;
        this.nextRecord = readNextValidRecord();
        return currentRecord;
    }
.
    @Override
    public void remove() {
        // Nothing to do here
    }
.
    /**
     * Returns next valid record. This happens after applying
     *
     * @return
     */
    private FlatFileUserAccount readNextValidRecord() {
        try {
            FlatFileUserAccount userAccObj = null;
            String recordStr;
            // Match the constraints or read next line
            do {
                System.out.println("Before record string");
                recordStr = getNextLine();
.
                // No more records ??
                if (recordStr == null)
                    return null;
```

```
.
                userAccObj = accountConverter
                        .convertStringRecordToAccountObj(recordStr);
            } while (!userAccObj.hasMatchingAttributes(attrConstraints));

            return userAccObj;
        } catch (Exception e) {
            System.out.println("Error reading record" + e.getMessage());
            e.printStackTrace();
            return null;
        }
    }
.
    private String getNextLine() throws IOException {
        String nextLine = userRecordReader.readLine();
.
        // No records ??
        if (nextLine == null) {
            this.userRecordReader.close();
            return null;
        }
.
        if (nextLine.trim().isEmpty()) {
            return getNextLine();
        }
.
        return nextLine;
    }
}
```

Example 10–10 FlatfileUserAccount

```
package org.identityconnectors.flatfile;
.
import java.util.HashMap;
import java.util.HashSet;
import java.util.Map;
import java.util.Set;
.
import org.identityconnectors.framework.common.objects.Attribute;
.
/**
 * Object representing a user entity
 *
 * @author admin
 *
 */
public class FlatFileUserAccount {
.
    /*
     * Mandatory attribute names
     */
    private Set<String> mandatoryAttrNames = new HashSet<String>();
.
    /*
     * Attributes making the account
     */
    private Map<String, String> attributes = new HashMap<String, String>();
.
```

```
    /**
     * Instantiates the attribute value map
     *
     * @param mandatoryAttributeNames
     *            Names of the attributes that are necessary
     * @param attributeValMap
     *            Name value map for the attributes.
     * @throws IllegalStateException
     *             If mandatory attributes are not found in attribute val map
     */
    public FlatFileUserAccount(Set<String> mandatoryAttributeNames,
            Map<String, String> attributeValMap) {
        // Check if mandatory attribute values are passed
        Set<String> attrValuesKeySet = attributeValMap.keySet();
        if (!attrValuesKeySet.containsAll(mandatoryAttributeNames))
            throw new IllegalStateException("Mandatory attributes missing");
.
        // Initialize
        this.mandatoryAttrNames = mandatoryAttributeNames;
        this.attributes = attributeValMap;
.
    }

    /**
     * Instantiates the attribute value map.
     * Considers all attributes to be mandatory
     * @param attributeValMap
     */
    public FlatFileUserAccount(Map<String, String> attributeValMap) {
        this.mandatoryAttrNames = attributeValMap.keySet();
        this.attributes = attributeValMap;
    }

    /**
     * Instantiates the attribute value map
     * @param attrs
     */
    public FlatFileUserAccount(Set<Attribute> attrs) {
    for(Attribute attr: attrs) {
    String attrName = attr.getName();

    //Consider first value. Multivalued not supported
    String attrVal = (String) attr.getValue().get(0);
    this.attributes.put(attrName, attrVal);
    }
    }
.
    /**
     * Updates the set of attributes. If new attributes present, they are
added,
     * If old attributes are present in the parameter set, values are updated
     *
     * @param updatedAttributeValMap
     */
    public void updateAttributes(Map<String, String> updatedAttributeValMap)
{
        this.attributes.putAll(updatedAttributeValMap);
    }

    /**
```

```
     * Updates the set of attributes.
     * @param upatedAttributes
     */
    public void updateAttributes(Set<Attribute> upatedAttributes) {
    Map<String, String> updatedAttributeValMap = new HashMap<String,
String>();
    for(Attribute attr: upatedAttributes) {
    String attrName = attr.getName();

    //Consider first value. Multivalued not supported
    String attrVal = (String) attr.getValue().get(0);
    updatedAttributeValMap.put(attrName, attrVal);
    }
    this.attributes.putAll(updatedAttributeValMap);
    }
.
    /**
     * Deletes the attributes with given name.
     *
     * @param attributeKeys
     *            Set of the attribute names that are needed
     * @throws UnsupportedOperationException
     *             if delete for mandatory attributes is attempted
     */
    public void deleteAttributes(Set<String> attributeKeys) {
        // Check if mandatory attributes are not there.
        for (String attrKey : attributeKeys) {
            if (this.mandatoryAttrNames.contains(attrKey))
                throw new UnsupportedOperationException(
                        "Delete for mandatory attributes not supported. Try
update");
            // Not deleting here as it might result inconsistent
        }
        // Remove the attributes
        for (String attrKey : attributeKeys) {
            this.attributes.remove(attrKey);
        }
    }
.
    /**
     * Gets the attribute of a given name
     *
     * @param attributeName
     * @return
     * @throws IllegalArgumentException
     *             if attribute is not there for a given name
     */
    public String getAttributeValue(String attributeName) {
        return this.attributes.get(attributeName);
    }
.
    /**
     * Returns the current set of attributes
     *
     * @return
     */
    public Map<String, String> getAllAttributes() {
        return this.attributes;
    }
.
```

```
    /**
     * Returns true if all passed attributes are matching for this object
     *
     * @param attrValMap
     * @return
     */
    public boolean hasMatchingAttributes(Map<String, String> attrValMap) {
        boolean noFilterSupplied = (attrValMap == null )||
(attrValMap.isEmpty());
        if (noFilterSupplied)
            // No filter. Everything matches
            return true;

        // Iterate to match attributes one by one
        Set<String> keySet = attrValMap.keySet();
        for (String attrName : keySet) {
            String objAttrVal = this.attributes.get(attrName);
            String passedValue = attrValMap.get(attrName);
.
            if (!objAttrVal.equals(passedValue))
                // This attribute is not same
                return false;
        }
.
        // All attributes are same
        return true;
    }
.
    /**
     * Returns the change log number
     *
     * @param changeLogAttrName
     *            attribute representing the number
     * @return
     */
    public int getChangeNumber(String changeLogAttrName) {
        String changeNumStr = this.attributes.get(changeLogAttrName);
        int changeNumber = 0;
.
        try {
            changeNumber = Integer.parseInt(changeNumStr);
        } catch (Exception e) {
            System.out.println("Not a valid change log number "
                    + changeLogAttrName + " :" + changeNumStr);
        }
.
        return changeNumber;
    }

    /**
     * Sets the given attribute with a new value
     * @param attrName
     * @param attrVal
     */
    public void setAttribute(String attrName, String attrVal) {
        this.attributes.put(attrName, attrVal);
    }

    /**
     * Updates the changelog number
```

```
     * @param changeLogAttrName
     * @param newChangeNumber
     */
    public void setChangeNumber(String changeLogAttrName, int
newChangeNumber) {
        String changeNumberValStr = "" + newChangeNumber;
        this.attributes.put(changeLogAttrName, changeNumberValStr);
    }
.
    @Override
    public String toString() {
        // Just print the attributes
        return this.attributes.toString();
    }
.
}
```

Example 10–11 FlatfileAccountConversionHandler

```
package org.identityconnectors.flatfile.utils;
.
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Set;
import java.util.StringTokenizer;
.
import org.identityconnectors.flatfile.FlatFileConfiguration;
import org.identityconnectors.flatfile.FlatFileUserAccount;
import org.identityconnectors.flatfile.io.FlatFileMetadata;
.
/**
 * Class for the utility functions
 *
 * @author Admin
 *
 */
public class AccountConversionHandler {
.
    private FlatFileConfiguration fileConfig;
    private FlatFileMetadata metadata;
.
    /**
     * Instantiates the handler class. But needs the configuration
     *
     * @param metadata
     * @param fileConfig
     */
    public AccountConversionHandler(FlatFileMetadata metadata,
            FlatFileConfiguration fileConfig) {
        this.fileConfig = fileConfig;
        this.metadata = metadata;
    }
.
    /**
     * Converts strings records to the user account objects.
     *
     * @param accountRecord
     * @return
```

```
     * @throws RuntimeException
     *             If string is not formatted as per accepted standards
     */
   public FlatFileUserAccount convertStringRecordToAccountObj(
           String accountRecord) {
.
        StringTokenizer tokenizer = new StringTokenizer(accountRecord,
                fileConfig.getTextFieldDelimeter());
.
        // Assert number of columns matching with number of tokens
        if (metadata.isDifferentFromNumberOfFields(tokenizer.countTokens()))
            throw new RuntimeException(
                    "Number of tokens doesn't match number of columns");
.
        // Get the attributes
        List<String> attrNames = metadata.getOrderedTextFieldNames();
        Map<String, String> attrValMap = new HashMap<String, String>();
.
        // Number of tokens are same. Same loop will work
        for (String attrName : attrNames) {
            String attrVal = "";
            if (tokenizer.hasMoreTokens())
                attrVal = tokenizer.nextToken();
.
            attrValMap.put(attrName, attrVal);
        }
.
        // Assumption : All attributes are mandatory for user. Change with
the
        // change in assumption
        Set<String> mandatoryAttributeNames = attrValMap.keySet();
        FlatFileUserAccount userAccountRecordObj = new FlatFileUserAccount(
                mandatoryAttributeNames, attrValMap);
        return userAccountRecordObj;
.
   }
.
    /**
     * Converts account objects to storable string records
     *
     * @param accountObj
     * @return
     */
    public String convertAccountObjToStringRecord(
            FlatFileUserAccount accountObj) {
        StringBuilder strRecord = new StringBuilder();
.
        // Build the string record from the object
        List<String> attrNames = metadata.getOrderedTextFieldNames();

        int index=0;
        for (String attrName: attrNames) {
            String attrVal = accountObj.getAttributeValue(attrName);
            strRecord.append(attrVal);

           // Add delimeter
            if (index < attrNames.size()-1) {
                strRecord.append(fileConfig.getTextFieldDelimeter());
                index++;
            } else {
```

```
                // Record ended
                String newLineCharacter = "\n";
                strRecord.append(newLineCharacter);
                break;
            }
        }
        return strRecord.toString();
    }
.
    /**
     * Asserts if given object is not null
     *
     * @param message
     * @param obj
     */
    public void assertNotNull(String message, Object obj) {
        if (obj == null)
            throw new RuntimeException(message);
    }

}
```

Example 10–12 Messages.Properties

```
USER_ACCOUNT_STORE_HELP=File in which user account will be stored
USER_ACCOUNT_STORE_DISPLAY=User Account File
USER_STORE_TEXT_DELIM_HELP=Text delimeter used for separating the columns
USER_STORE_TEXT_DELIM_DISPLAY=Text Field Delimeter
UNIQUE_ATTR_HELP=The name of the attribute which will act as unique identifier
UNIQUE_ATTR_DISPLAY=Unique Field
CHANGELOG_ATTR_HELP=The name of the attribute which will act as changelog
CHANGELOG_ATTR_DISPLAY=Changelog Field
```

10.2 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

10.2.1 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 in 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**.

10.2.2 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

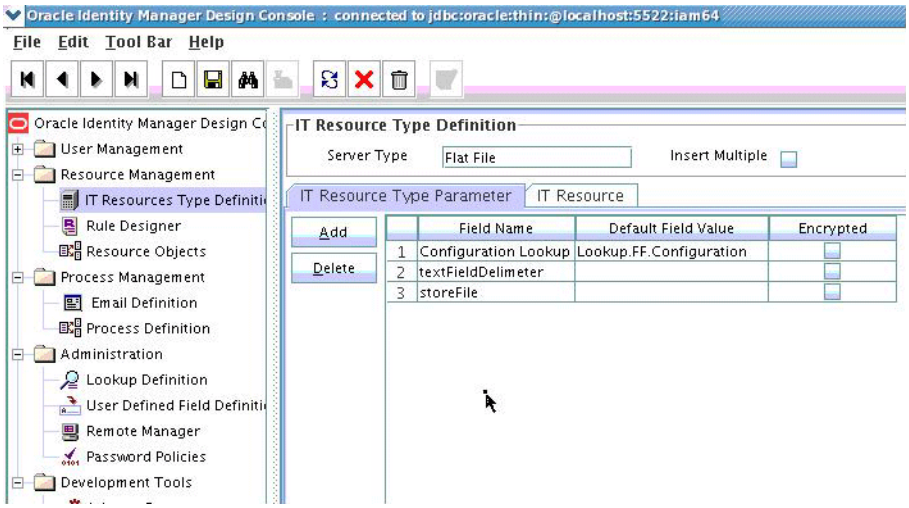
10.2.2.1 Creating the IT Resource Type Definition

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 10–1.
   - **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.
   - **textFieldDelimeter** maps to the textFieldDelimeter 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 10–1 IT Resource Type Definition in Design Console



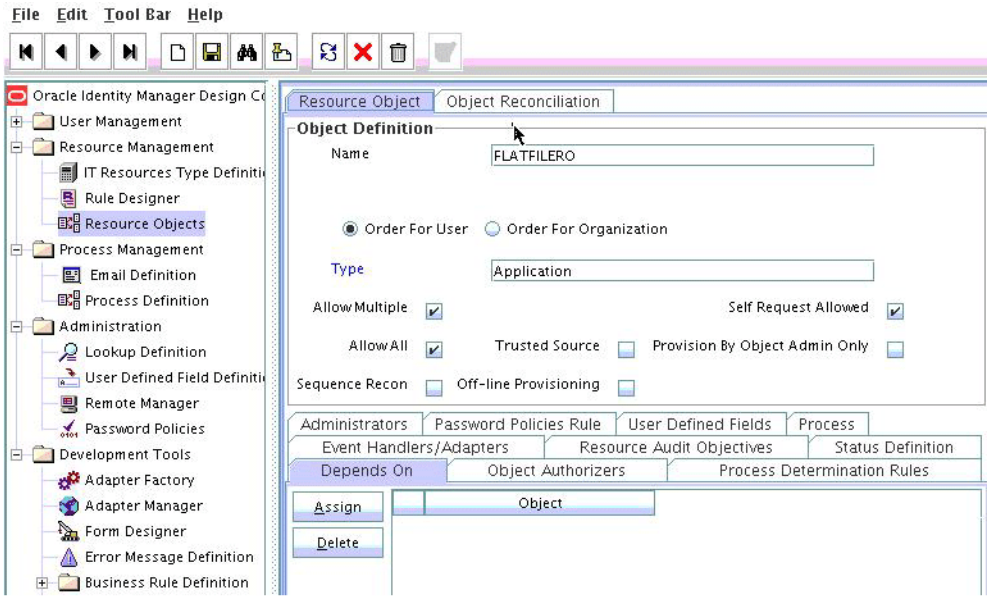
10.2.2.2 Creating the Resource Object

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 10–2.

Figure 10–2 Resource Objects in Design Console



10.2.2.3 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.

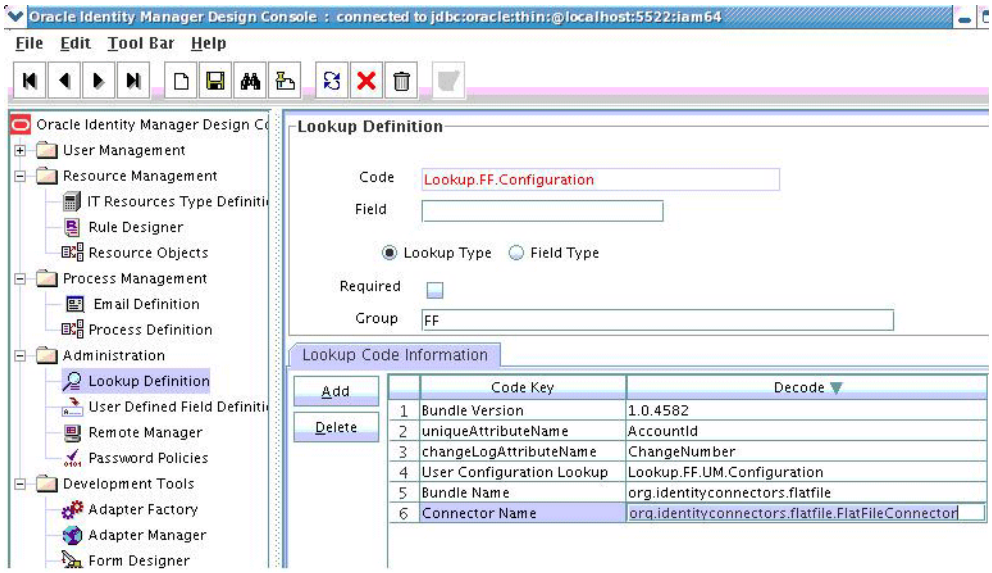
- Creating the Main Configuration Lookup
- Creating Object Type Configuration Lookup

10.2.2.3.1 Creating the Main Configuration Lookup The Configuration Lookup (as defined in Section 10.2.2.1, "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 10.2.2.1, "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 10–3.
   - 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 10–3 Lookup Definition in Design Console



10.2.2.3.2 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 10.2.2.3.1, "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.

---

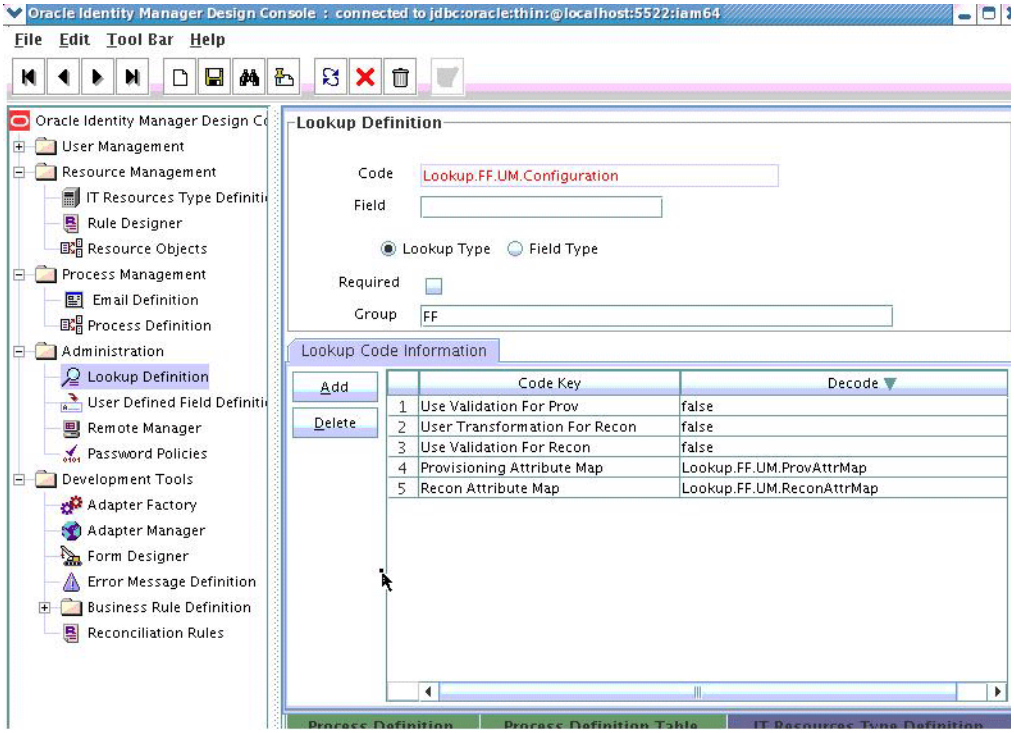
1. Log in to the Oracle Identity Manager Design Console.
2. Click Lookup Definition under Administration.
3. Create a new Lookup and add Lookup.FF.UM.Configuration as the Code.
4. Set the following attributes as illustrated in Figure 10–4.

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 10–4 Second Lookup Definition in Design Console



10.2.3 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

10.2.3.1 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.

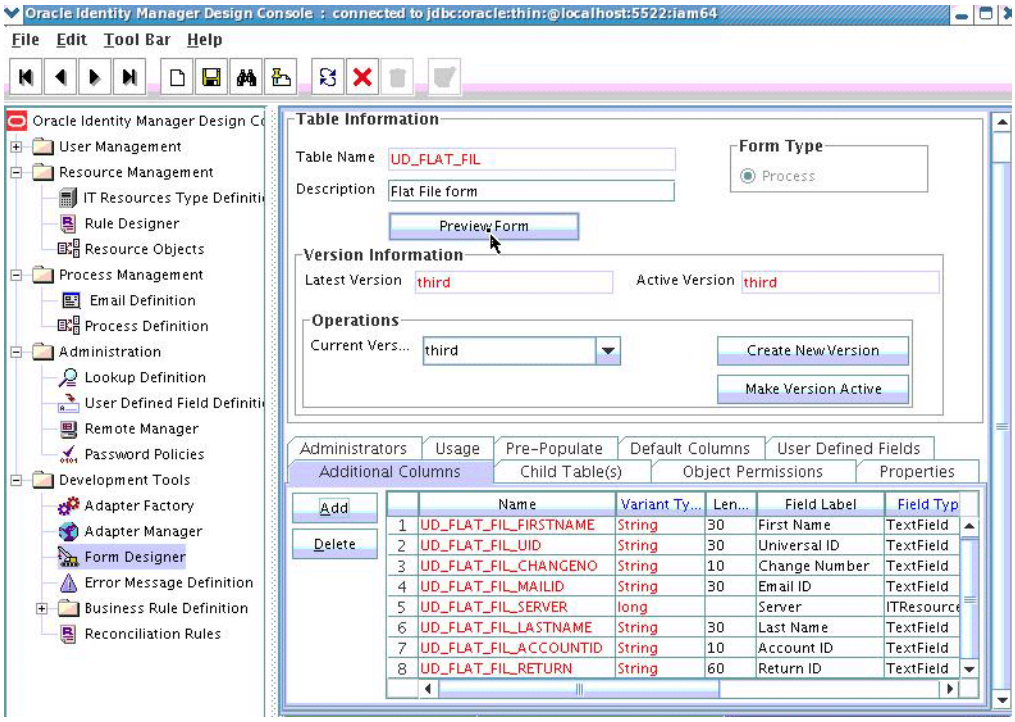
Attributes defined in the process form are not conventions. The form is a way to challenge the attributes that need to be passed to the identity connector. In general, define an attribute for each supported attribute in the identity connector.

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 10–5.

Figure 10–5 Form Designer in Design Console



4. Add the attributes defined in the connector schema, as listed in Table 10–1.

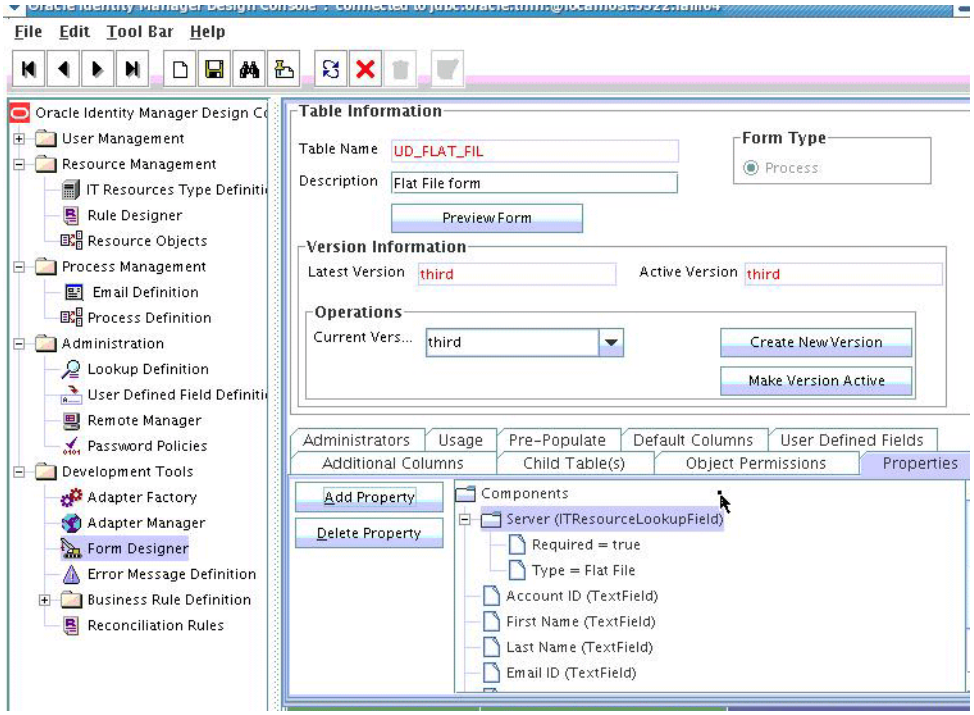
Table 10–1 Form Designer Fields

Name	Variant	Field Label	Field Type
UD_FLAT_FIL_FIRSTNAME	String	First Name	TextField
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 10–6.
   - Required = true
   - Type = Flat File

Figure 10–6 Properties of Form Designer in Design Console



7. Save the form.
8. Click Make Version Active.

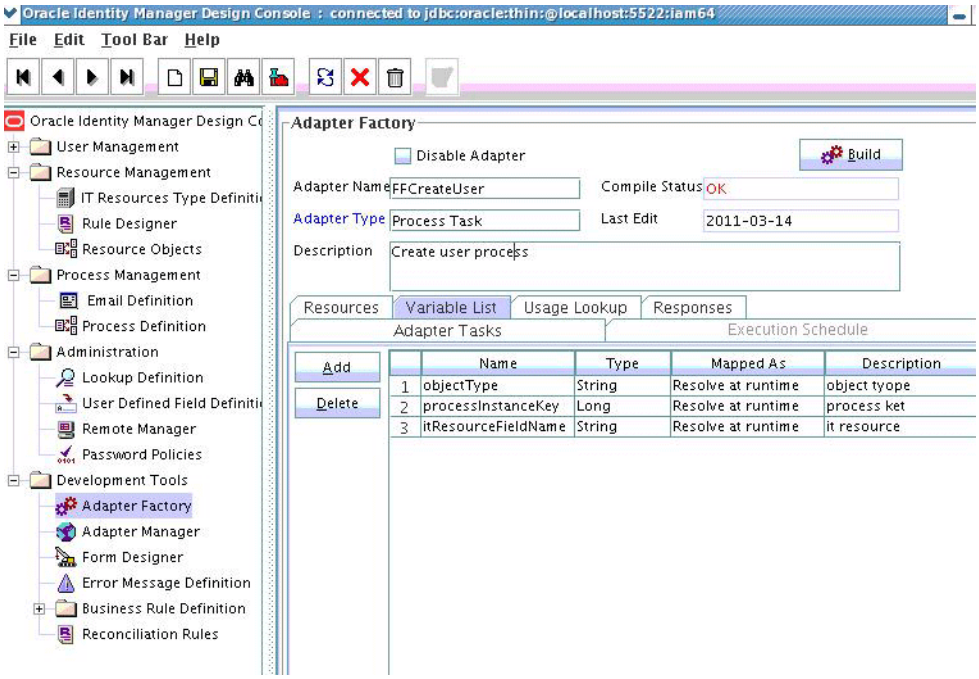
10.2.3.2 Creating Adapters

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 10–7.

- 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.

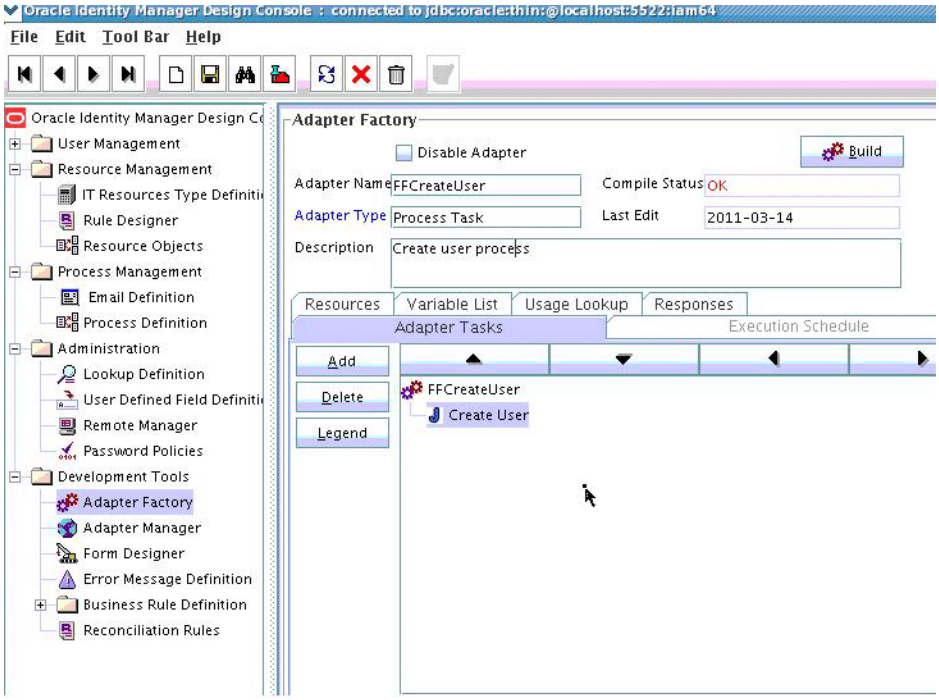
Figure 10–7 Adapter Factory Variable List in Design Console



7. Add a Java functional task to the adapter by following this sub procedure, as shown in Figure 10–8.
   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 10–8 Adapter Factory in Design Console



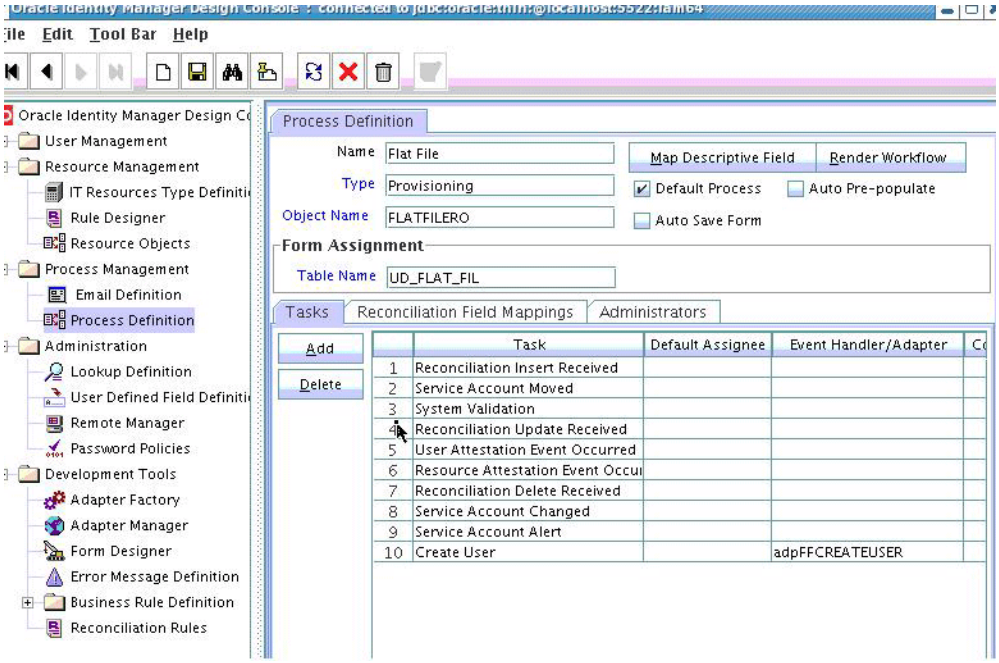
8. Save and build the adapter.

10.2.3.3 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.

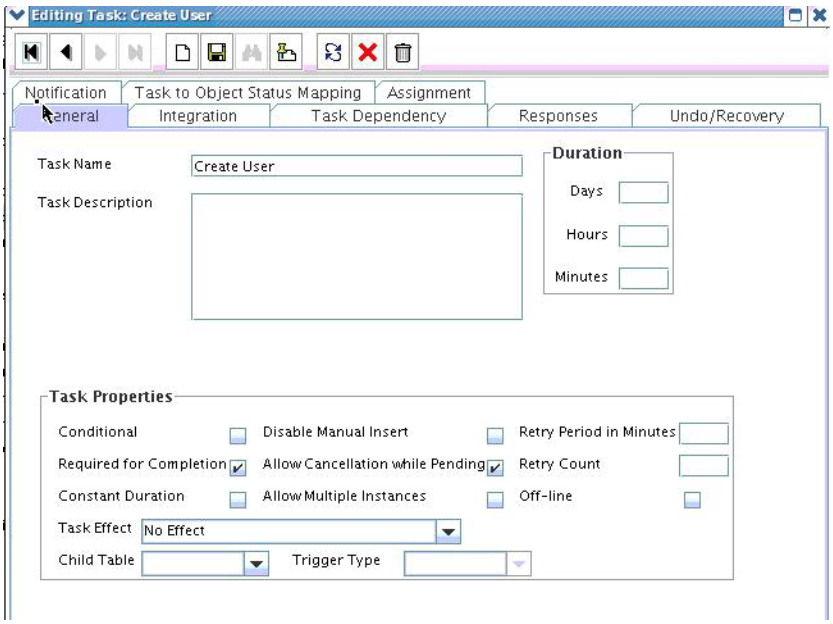
1. Log in to the Oracle Identity Manager Design Console.
2. Click Process Definition under the Process Management tab.
3. Create a new process definition and name it Flat File as illustrated in Figure 10–9.

Figure 10–9 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 10–10.

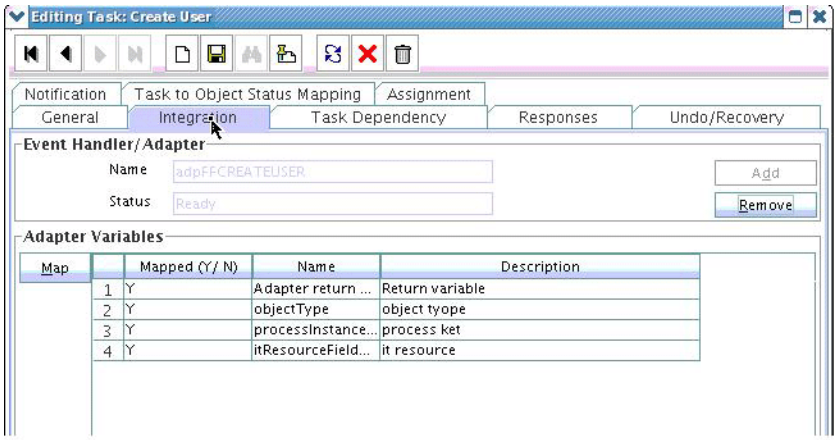
Figure 10–10 Editing Task Screen in Design Console



9. Click the **Integration** tab.
10. Click Add and select the FFCreateUser adapter from the list as illustrated in Figure 10–11.

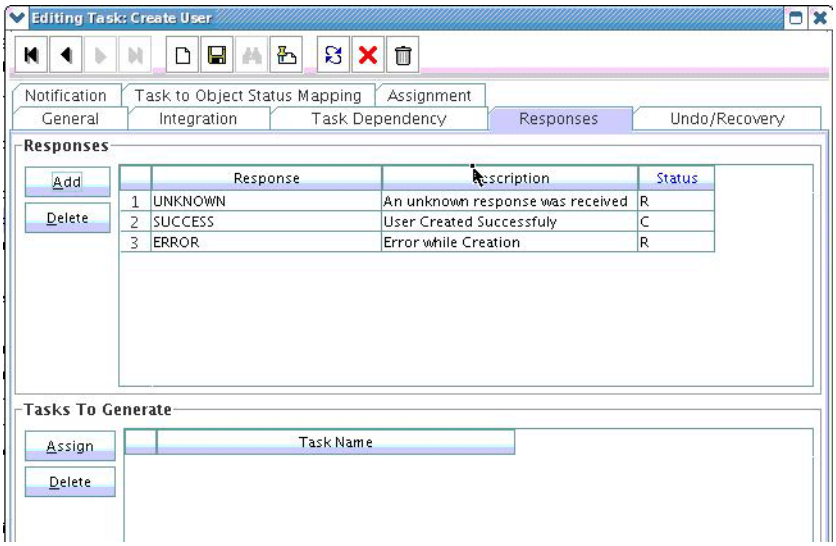
    The adapter will be available only after it is compiled.

Figure 10–11 Integration Tab in Design Console



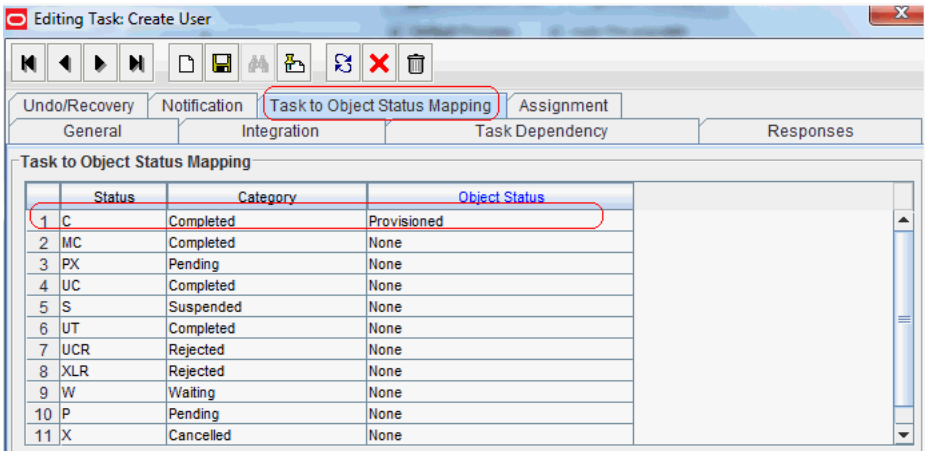
11. Map the variables as follows to set the response code returned by the identity connector.
    - Adapter Return Variable – Response Code
    - Object Type – [Literal:String] User (Name of the object type)
    - Process Instance Key – [Process Data] Process Instance
    - IT Resource Field Name – [Literal:String] 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 10–12.
    - 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 10–12 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 10–13:

Figure 10–13 Task to Object Status Mapping



15. Save the process task.

10.2.3.4 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 10.2.2.3.2, "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

10.2.3.4.1 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].

10.2.4 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

10.2.4.1 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

10.2.4.1.1 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 10–13. Make sure to update database value to your database.

Example 10–13 Deployment Manager XML with Scheduled Task Details

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<xl-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>
```

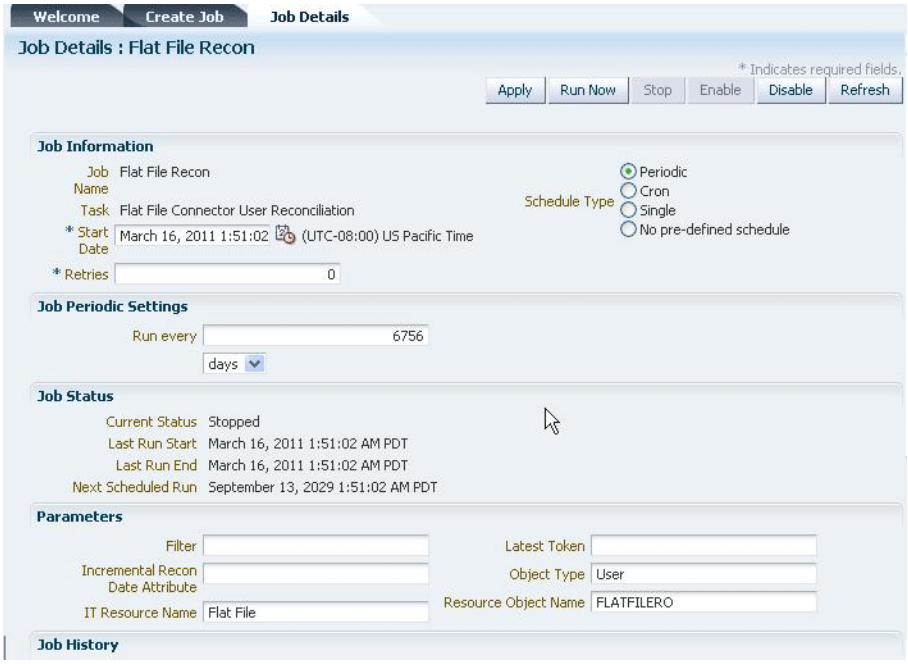
```
            <name>Flat File Connector User Reconciliation</name>
            <class>oracle.iam.connectors.icfcommon.recon.SearchReconTask</class>
            <description>Flat File Connector User Reconciliation</description>
            <retry>0</retry>
            <parameters>
              <string-param 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 to Oracle Identity System Administration. Under System Management, click **Import**.
4. Select the Flat File Connector User Reconciliation.xml file, and click **Import**.
5. Complete the steps in the wizard.

10.2.4.1.2 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 10–14.

Figure 10–14 The Scheduled Task Screen



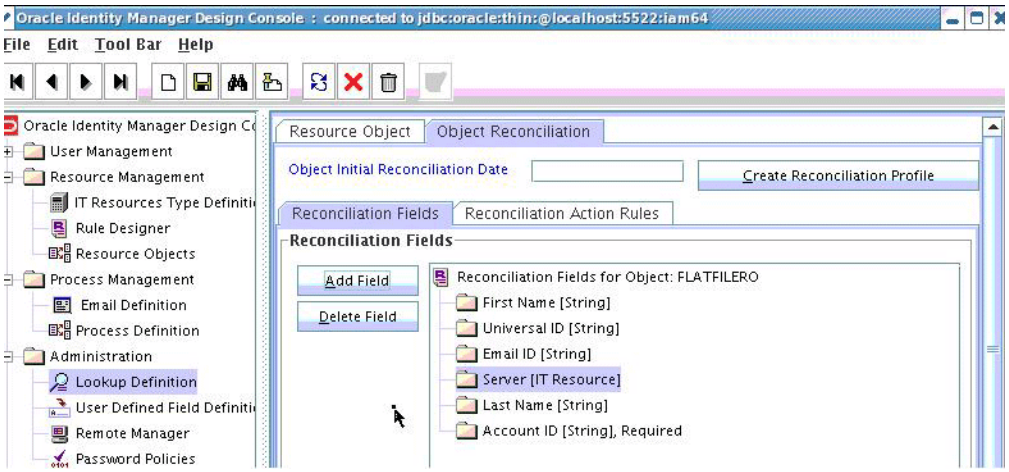
4. Set the parameters as follows:
   - IT Resource Name takes a value of Flat File.
   - Resource Object Name takes a value of FLATFILERO.
   - Object Type takes a value of User.
5. Click **Apply**.

10.2.4.2 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 10–15.

Figure 10–15 Object Reconciliation in Design Console



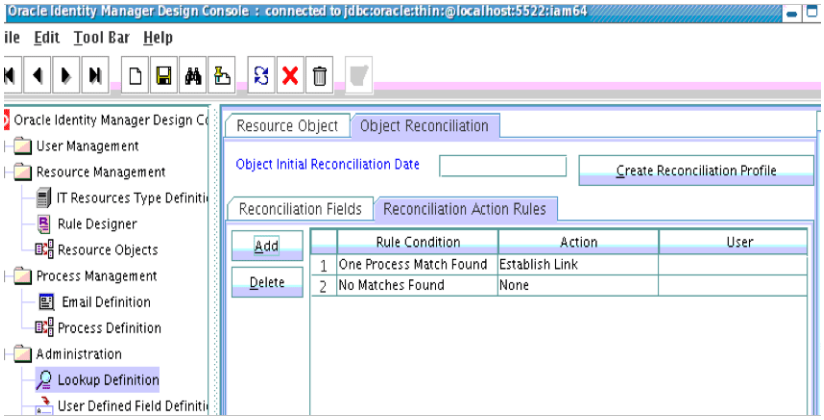
5. Add following reconciliation fields:
   - First Name [String]
   - Universal ID [String]
   - Email ID [String]
   - IT Resource Name [String]
   - Last Name [String]
   - Account ID [String], Required
6. Save the configuration.

10.2.4.3 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 10–16 Reconciliation Action Rules in Design Console



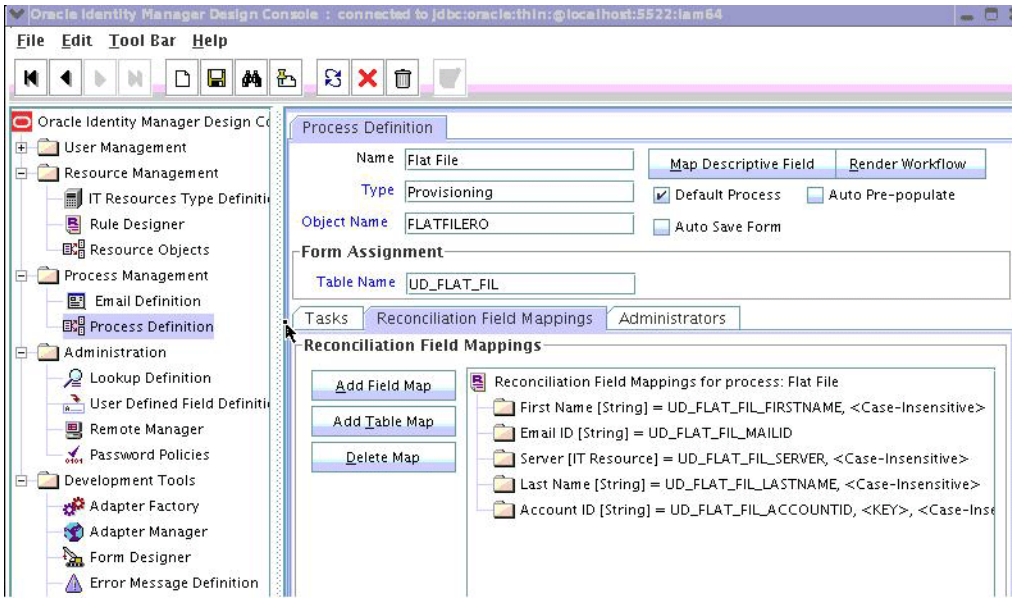
4. Add an action rule defined as One Process Match Found (Rule Condition) and Establish Link (Action).
5. Add an action rule defined as One Entity Match Found (Rule Condition) and Establish Link (Action).
6. Click **Create Reconciliation Profile**.
7. Click **Save**.

10.2.4.4 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 10–17.

Figure 10–17 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.

10.2.4.4.1 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].

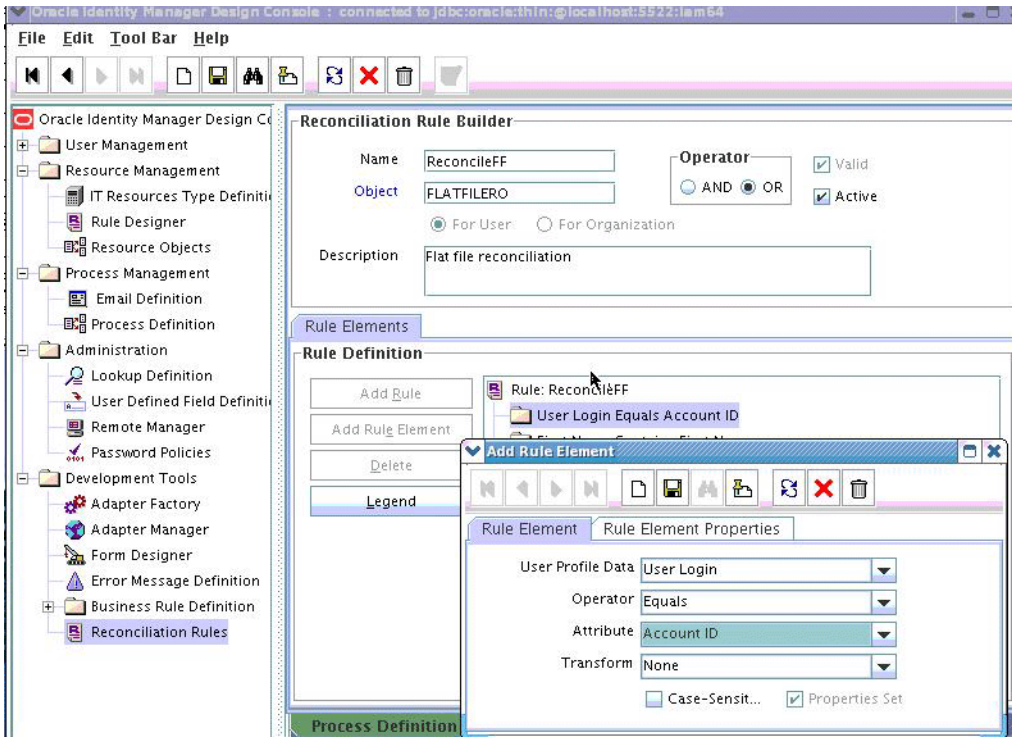
10.2.4.5 Defining a Reconciliation Matching Rule

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 10–18 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.

10.3 Provisioning a Flat File Account

The flat file connector is ready to work. 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.

10.4 Configuring SSL for Java Connector Server

To enable SSL for Java connector server:

1. Edit $*CONNECTOR_SERVER_HOME*/conf/ConnectorServer.properties as the following:

```
connectorserver.usessl=true
connectorserver.keyStore={full path to your keystore file}
connectorserver.keyStoreType=JKS (optionally you can set key store type, if
```

```
not set JSK is used by default)
```

2. Provide key store password:
   a. Set it in ConnectorServer.properties.
   b. Set connectorserver.promptKeyStorePassword=false in ConnectorServer.properties.
   c. Set the password:

      ```
      cd $CONNECTOR_SERVER/bin
      ```

      For UNIX

      ```
      connectorserver.sh /setKeyStorePassword thepassword
      ```

      For Windows

      ```
      ConnectorServer.bat /setKeyStorePassword thepassword
      ```

      This command will set the encrypted value to "connectorserver.keyStorePassword" in ConnectorServer.properties.
3. Ask for key store password every time by starting the connector server set connectorserver.promptKeyStorePassword=true in ConnectorServer.properties.

   The following example snippet shows the ConnectorServer.properties shipped with Java Connector Server:

```
.
##
## The port we are to run on
##
connectorserver.port=8759
##
## The bundle directory in which to find the bundles
##
connectorserver.bundleDir=bundles
.
##
## The bundle directory in which to find any libraries needed by bundles at
runtime
##
connectorserver.libDir=lib
.
##
## Set to true to use SSL.
## NOTE: Check also the following settings which are related to SSL:
## connectorserver.promptKeyStorePassword
## connectorserver.keyStore
## connectorserver.keyStoreType
## connectorserver.keyStorePassword
connectorserver.usessl=false
.
##
## If set to true the user is prompted for key store password at startup.
## If set to false the key store password needs to be set with
-setKeyStorePassword command first.
##
connectorserver.promptKeyStorePassword=true
.
##
```

```
## Full path to key store.
##
connectorserver.keyStore=/tmp/KeyStore.jks
.
##
## KeyStore type
##
#connectorserver.keyStoreType=JKS
.
##
## Encrypted password. Set this by using the -setKeyStorePassword flag.
## It is used only if connectorserver.promptKeyStorePassword is set to false.
##
connectorserver.keyStorePassword=
.
##
## Optionally specify a specific address to bind to
##
#connectorserver.ifaddress=localhost
.
##
## Secure hash of the gateway key. Set this by using the
## -setKey flag
##
connectorserver.key=lmA6bMfENJGlIDbfrVtklXFK32s\=
.
##
## Use standard JDK logging
##
connectorserver.loggerClass=org.identityconnectors.common.logging.impl.JDKLogge
r
```

11

Developing Identity Connectors Using .NET

This chapter is a tutorial that walks through the procedures necessary to develop an identity connector in .NET 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 .NET identity connector and creating Oracle Identity Manager metadata for user provisioning and reconciliation processes. It contains the following sections:

- Section 11.1, "Developing a Flat File .NET Connector"
- Section 11.2, "Deploying the Identity Connector Bundle on .NET Connector Server"
- Section 11.3, "Provisioning a Flat File Account"

11.1 Developing a Flat File .NET 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. In the documentation we would discuss sample implementation of a flat file connector showing Create, Delete, Update and Search operations. To keep implementations and documentation simple, Configuration properties and Schema supported by connector have been kept to a minimum. This connector implementation should only be used as a sample which would help to create actual connectors.

To keep the connector implementation simple, lets assume that flat file has only Name, Gender, Qualification, Age attributes. We would have only two configurations File Location and Delimiter. Rest configurations would be hardcoded in the sample.

1. Setting up the project in Microsoft Visual Studio and using the connector:
   a. Create a new visual studio project of type library.
   b. Make sure to add the following dlls as references:

      - Common.dll

      - Framework.dll

      - FrameworkInternal.dll

      - System.dll

      - System.Core.dll

      These dlls should be available with the .NET connector server.

2. Implement the configuration class for the Flat File Connector by extending the Org.IdentityConnectors.Framework.Spi.AbstractConfiguration base class.

Example 11–1 Implementation of AbstractConfiguration

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Org.IdentityConnectors.Framework.Spi;?
using Org.IdentityConnectors.Framework.Common.Exceptions;?
using System.IO;

namespace Org.IdentityConnector.FlatFileConnector?
{
    /// <summary>
    /// Configuration class for flat file connector representing target system
information?
    /// </summary>
    public class FlatFileConfiguration : AbstractConfiguration?
    {
        #region FileName
        /// <summary>
        /// Target file name
        /// </summary>
        /// <value>
        /// File name with complete path. As for executing the .NET Connector
bundle we need .NET Connector Server, hence the file should reside
        /// on the machine where the connector server is present.
        /// </value>
        [ConfigurationProperty(Required = true, Order = 1)]
        public String FileName { get; set; }
        #endregion

        #region Delimiter
        /// <summary>
        /// Delimiter used within the target flat file
        /// </summary>
        /// <value>
        /// Delimter
        /// </value>
        [ConfigurationProperty(Required = true, Order = 2)]
        public String Delimiter { get; set; }
        #endregion

        #region
        /// <summary>
        /// Validates if the configuration properties provided are as requiered,
if not throw ConfigurationException
        /// </summary>
        public override void Validate()
        {
            if (this.FileName == null || this.FileName.Length == 0)
            {
                throw new ConfigurationException("Configuration property FileName
cannot be null or empty");
            }
            if (!File.Exists(this.FileName))
            {
                throw new ConfigurationException("Target file " + this.FileName +
```

```
" does not exist");
            }
            if (this.Delimiter == null || this.Delimiter.Length == 0)
            {
                throw new ConfigurationException("Configuration property Delimiter
cannot be null or empty");
            }
        }
        #endregion
    }
}
```

3. Create connector class for the Flat File Connector by implementing different SPI interfaces `Org.IdentityConnectors.Framework.Spi`.

   Example 11–2 implements the PoolableConnector,CreateOp,SchemaOp,TestOp,DeleteOp,UpdateOp,SearchOp<String> interfaces and thus supports all CRUD operations

Example 11–2 Implementation of PoolableConnector

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Org.IdentityConnectors.Framework.Spi;
using System.IO;
using Org.IdentityConnectors.Framework.Common.Exceptions;
using System.Security.AccessControl;
using Org.IdentityConnectors.Framework.Spi.Operations;
using Org.IdentityConnectors.Framework.Common.Objects;
using Org.IdentityConnectors.Common;

namespace Org.IdentityConnector.FlatFileConnector
{
    /// <summary>
    /// FlatFileConnector showing implementation of SchemaOp, test, create,
delete, update and search operations.
    /// </summary>
    [ConnectorClass("FlatFileConnector_
DisplayNameKey",typeof(FlatFileConfiguration))]
    public class FlatFileConnector :
PoolableConnector,CreateOp,SchemaOp,TestOp,DeleteOp,UpdateOp,SearchOp<String>
    {

        /// <summary>
        /// Flat file configuration instance. This instance has the target system
information.
        /// </summary>
        private FlatFileConfiguration config;


        #region Init
        /// <summary>
        /// Create a connection to target and store it for later use. But here we
just set attributes of target file
        /// name to Normal
        /// </summary>
        /// <param name="config">Configuration Object</param>
        public void Init(Configuration config)
```

```
        {
            this.config = (FlatFileConfiguration)config;
            File.SetAttributes(this.config.FileName, FileAttributes.Normal);
        }
        #endregion



        #region CreateOp Members
        /// <summary>
        /// This creates a new row in the target file with the data as sent in the
'attrs'
        /// </summary>
        /// <param name="objClass">The ObjectClass. Here we support only
Account</param>
        /// <param name="attrs">Attributes of this Account that need to be created
on target</param>
        /// <param name="options">Will always be empty</param>
        /// <returns>Unique id Uid, representing the Account which was just
created</returns>
        public Uid Create(ObjectClass objClass, ICollection<ConnectorAttribute>
attrs, OperationOptions options)
        {
            ConnectorAttribute NameAttribute =
ConnectorAttributeUtil.Find(Name.NAME, attrs);
            ConnectorAttribute AgeAttribute = ConnectorAttributeUtil.Find("Age",
attrs);
            ConnectorAttribute QualificationAttribute =
ConnectorAttributeUtil.Find("Qualification", attrs);
            ConnectorAttribute GenderAttributute =
ConnectorAttributeUtil.Find("Gender", attrs);
            StreamWriter writer = File.AppendText(this.config.FileName);
            writer.WriteLine("\nName:" +
ConnectorAttributeUtil.GetAsStringValue(NameAttribute) + this.config.Delimiter +
"Age:" + ConnectorAttributeUtil.GetAsStringValue(AgeAttribute) +
this.config.Delimiter + "Qualification:" +
ConnectorAttributeUtil.GetAsStringValue(QualificationAttribute) +
this.config.Delimiter + "Gender:" +
ConnectorAttributeUtil.GetAsStringValue(GenderAttributute));
            writer.Flush();
            writer.Dispose();
            writer.Close();
            return new
Uid(ConnectorAttributeUtil.GetAsStringValue(NameAttribute));
        }
        #endregion



        #region DeleteOp Members
        /// <summary>
        /// Deletes an entity from target flat file. We support only ACCOUNT
object class.
        /// If the Uid (user name) is not found then UnknownUidException is thrown
        /// </summary>
        /// <param name="objClass"></param>
        /// <param name="uid"></param>
        /// <param name="options"></param>
        public void Delete(ObjectClass objClass, Uid uid, OperationOptions
options)
```

```
        {
            String[] allLines = File.ReadAllLines(this.config.FileName);
            String[] newLines = new String[allLines.Length];
            Boolean userExisted = false;
            for (int i = 0; i < allLines.Length; i++)
            {
                char[] separator = new char[] { '$' };
               String[] thisLineSplit = allLines[i].Split(separator);

                String name = "";
                foreach (String str in thisLineSplit)
                {
                    if (str.StartsWith("Name"))
                    {
                        name = str;
                        break;
                    }
                }
                if (!name.Equals("Name" + ":" + uid.GetUidValue()))
                {
                    newLines[i] = allLines[i];
                }
                else
                {
                    userExisted = true;
                }

            }
            if (userExisted)
            {
                File.WriteAllText(this.config.FileName, String.Empty);
                File.WriteAllLines(this.config.FileName, newLines);
            }
            else
            {
                throw new UnknownUidException("Uid "+uid.GetUidValue()+" not
found");
            }
        }
        #endregion

        #region UpdateOp Members
        /// <summary>
        /// Updates information of an existing user on the target flat file
        /// </summary>
        /// <param name="objclass">The ObjectClass. Here we support only
user</param>
        /// <param name="uid">Unique id of the user using which we can find out
the user on target. This is the returned vaue by CreateOp implementation</param>
        /// <param name="replaceAttributes">Updated attributes of user which
should replace all existing user information on target</param>
        /// <param name="options">This will always be empty</param>
        /// <returns>Updated uid. It can be the same value which was provided to
this method.</returns>
        public Uid Update(ObjectClass objclass, Uid uid,
ICollection<ConnectorAttribute> replaceAttributes, OperationOptions options)
        {
            String uidValue = uid.GetUidValue();
            String[] allLines = File.ReadAllLines(this.config.FileName);
            String[] updatedLines = new String[allLines.Length];
```

```
            Boolean userExists = false;
            Uid updatedUid = uid;
            for(int i = 0; i < allLines.Length; i++)
            {
                String[] thisLineSplit = allLines[i].Split(new char[] { '$' });
                String name = "";
                foreach (String str in thisLineSplit)
                {
                    if (str.StartsWith("Name"))
                    {
                        name = str;
                        break;
                    }
                }
                String nameToBeUpdated = "Name:" + uidValue;
                if (!name.Equals(nameToBeUpdated))
                {
                    updatedLines[i] = allLines[i];
                }
                else
                {
                    ConnectorAttribute NameAttribute =
ConnectorAttributeUtil.Find(Name.NAME, replaceAttributes);
                    ConnectorAttribute AgeAttribute =
ConnectorAttributeUtil.Find("Age", replaceAttributes);
                    ConnectorAttribute QualificationAttribute =
ConnectorAttributeUtil.Find("Qualification", replaceAttributes);
                    ConnectorAttribute GenderAttribute =
ConnectorAttributeUtil.Find("Gender", replaceAttributes);
                    updatedLines[i] =
"Name:"+NameAttribute.Value.First().ToString()+this.config.Delimiter+

AgeAttribute.Name+":"+AgeAttribute.Value.First().ToString()+this.config.Delimiter+

QualificationAttribute.Name+":"+QualificationAttribute.Value.First().ToString()+th
is.config.Delimiter+

GenderAttribute.Name+":"+GenderAttribute.Value.First().ToString();
                    userExists = true;
                    updatedUid = new Uid(NameAttribute.Value.First().ToString());
                }

            }
            File.WriteAllText(this.config.FileName, String.Empty);
            File.WriteAllLines(this.config.FileName, updatedLines);
            if (!userExists)
            {
                throw new UnknownUidException("User "+uid.GetUidValue()+" not
found");
            }
            return updatedUid;
        }

        #endregion

        #region SearchOp<string> Members

        /// <summary>
        /// Returns a filter translator used by ExecuteQuery. The functionality of
filter translator is to translate any filters provided by calling application
```

```
(OIM/OW/OPAM) to native queries.
        /// </summary>
        /// <param name="oclass">The ObjectClass. We support only ACCOUNT</param>
        /// <param name="options">Options</param>
        /// <returns>FilterTranslator instance</returns>

        public
Org.IdentityConnectors.Framework.Common.Objects.Filters.FilterTranslator<string>
CreateFilterTranslator(ObjectClass oclass, OperationOptions options)
        {
            return new FlatFileFilterTranslator();
        }

        /// <summary>
        /// Performs search on target based on query. Uses the handler instance to
return back the searched result.
        /// </summary>
        /// <param name="oclass">The ObjectClass. This tells if we have to search
for user (ACCOUNT) or group (GROUP). We support only user</param>
        /// <param name="query">Query as returned by FilterTranslator</param>
        /// <param name="handler">handler to return back result to caller</param>
        /// <param name="options">Options containing what attributes of entity to
return back</param>
        public void ExecuteQuery(ObjectClass oclass, string query, ResultsHandler
handler, OperationOptions options)
        {

            String[] results = GetResults(query);
            foreach (String result in results)
            {
                Console.WriteLine("Result = "+result);
                String result1 = result.Trim();
                if (result1.Length > 0)
                {
                    Console.WriteLine("Submitting result = " + result1);
                    SubmitConnectorObject(result1, handler);
                }
            }
        }

        #region SchemaOp Members
        /// <summary>
        /// Defines the schema supported by this connector
        /// </summary>
        /// <returns>Schema</returns>
        public Schema Schema()
        {
            SchemaBuilder schemaBuilder = new
SchemaBuilder(SafeType<Connector>.Get(this));
            ICollection<ConnectorAttributeInfo> connectorAttributeInfos = new
List<ConnectorAttributeInfo>();

connectorAttributeInfos.Add(ConnectorAttributeInfoBuilder.Build("Name"));

connectorAttributeInfos.Add(ConnectorAttributeInfoBuilder.Build("Age"));

connectorAttributeInfos.Add(ConnectorAttributeInfoBuilder.Build("Qualification"));

connectorAttributeInfos.Add(ConnectorAttributeInfoBuilder.Build("Gender"));
            schemaBuilder.DefineObjectClass(ObjectClass.ACCOUNT_NAME,
```

```
connectorAttributeInfos);
            return schemaBuilder.Build();
        }

        #endregion

        #region TestOp Members
        /// <summary>
        /// Should ideally test the connecttion with target. But here we just
print something as we have assumed that target file is on same machine
        /// </summary>
        public void Test()
        {
            Console.Write("Tested connection!");
        }

        #endregion


        #region CheckAlive
        /// <summary>
        /// Check connection to target system is alive or not. But here we just
check if target file name
        /// provided in the FlatFileConfiguration is available or not.
        /// </summary>
        public void CheckAlive()
        {
            if (!File.Exists(this.config.FileName))
            {
                throw new ConnectorException("Target file " + this.config.FileName
+ " does not exist");
            }
        }
        #endregion



        #region Dispose
        /// <summary>
        /// Remove connection from target, dispose any of the resources used. But
here we just chill.
        /// </summary>
        public void Dispose()
        {
            //chill :)
        }
        #endregion

        private void SubmitConnectorObject(String result, ResultsHandler handler)
        {
            ConnectorObjectBuilder cob = new ConnectorObjectBuilder();
            String[] resultSplit = result.Split(new char[]{'$'});
            ICollection<ConnectorAttribute> attrs = new
List<ConnectorAttribute>();
            foreach (String str in resultSplit)
            {
                ConnectorAttributeBuilder cab = new ConnectorAttributeBuilder();
                cab.AddValue(str.Split(new char[] { ':' })[1]);
                if (str.StartsWith("Name"))
                {
```

```
                    cob.SetName(Name.NAME);
                    cob.SetUid(str.Split(new char[] { ':' })[1]);
                    cab.Name = Name.NAME;
                }
                else
                {
                    cab.Name = str.Split(new char[] { ':' })[0];
                }
                attrs.Add(cab.Build());
            }
            cob.AddAttributes(attrs);
            handler(cob.Build());
        }

        private String[] GetResults(String query)
        {
            String[] allLines = File.ReadAllLines(this.config.FileName);
            String[] results = allLines;
if (query != null)
            {
                for (int i = 0; i < allLines.Length; i++)
                {
                    String[] thisLineSplit = allLines[i].Split(new char[]{'$'});
                    Boolean foundResult = false;
                    foreach (String str in thisLineSplit)
                    {
                        if (str.StartsWith("Name") && str.Equals(query))
                        {
                            foundResult = true;
                            break;
                        }
                    }
                    if (foundResult)
                    {
                        return new String[] {allLines[i]};
                    }
                }
            }

            return results;
        }

        #endregion
    }
}
```

4. This connector supports only the CreateEqualsExpression operation. Implement the CreateEqualsExpression. Example 11–3 illustrates the sample implementation of Org.IdentityConnectors.Framework.Common.Objects.Filters.AbstractFilterTranslator<T> that defines the filter operation.

Example 11–3 Implementation of AbstractFilterTranslator<T>

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Org.IdentityConnectors.Framework.Common.Objects.Filters;
using Org.IdentityConnectors.Framework.Common.Objects;
```

```
namespace Org.IdentityConnector.FlatFileConnector
{
    /// <summary>
    /// FlatFileFilterTranslator. This translator converts the equalsFilter
provided by the calling application to native query which can be used by the
connector while searching.
    /// The implementation shown supports only equals filter. i.e it has provided
implementation for only CreateEqualsExpression, this means that if any other
filter is provided
    /// by the calling application, it would not be translated as a native query
and search implementation gets all users and filtering will be done by ICF with
all results.
    ///
    /// </summary>
    public class FlatFileFilterTranslator : AbstractFilterTranslator<String>
    {
        /// <summary>
        /// Creates a native query for equals filter and returns it only if equals
filter is constructed for Name attribute and not for any other attributes.
        /// </summary>
        /// <param name="filter">Filter provided by calling application</param>
        /// <param name="not"></param>
        /// <returns></returns>
        protected override string CreateEqualsExpression(EqualsFilter filter, bool
not)
        {
            ConnectorAttribute attr = filter.GetAttribute();
            if (attr.Name.Equals(Name.NAME))
            {
                return "Name:" + attr.Value.First().ToString();
            }
            return null;
        }
    }
}
```

5. Implement the different classes (as mentioned in steps 2, 3, and 4).
6. Make a note of AssemblyVersion present in the AssemblyInfo.cs of the project.

   Sample AssemblyInfo.cs file:

```
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;

// General Information about an assembly is controlled through the following
// set of attributes. Change these attribute values to modify the information
// associated with an assembly.
[assembly: AssemblyTitle("FlatFileConnector")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("Oracle Corporation")]
[assembly: AssemblyProduct("FlatFileConnector")]
[assembly: AssemblyCopyright("Copyright © Oracle Corporation 2012")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]

// Setting ComVisible to false makes the types in this assembly not visible
// to COM components.  If you need to access a type in this assembly from
```

```
// COM, set the ComVisible attribute to true on that type.
[assembly: ComVisible(true)]

// The following GUID is for the ID of the typelib if this project is exposed
to COM
[assembly: Guid("79eec317-62bd-49a5-9512-88d61135684c")]

// Version information for an assembly consists of the following four values:
//
//      Major Version
//      Minor Version
//      Build Number
//      Revision
//
// You can specify all the values or you can default the Build and Revision
Numbers
// by using the '*' as shown below:
// [assembly: AssemblyVersion("1.0.*")]
[assembly: AssemblyVersion("1.0.0.0")]
[assembly: AssemblyFileVersion("1.0.0.0")]
```

7. Build the project. The project must create the connector DLL.

11.2 Deploying the Identity Connector Bundle on .NET Connector Server

For all the connectors that are implemented in .NET, you need to have .NET Connector Server for the execution of the connector. The connector bundle cannot be deployed within Oracle Identity Manager. Therefore, you must perform the following procedures in order to integrate the ICF .NET Identity Connector with Oracle Identity Manager:

- Section 11.2.1, "Registering the Connector Bundle with .NET Connector Server"
- Section 11.2.2, "Creating Basic Identity Connector Metadata"
- Section 11.2.3, "Creating Provisioning Metadata"
- Section 11.2.4, "Creating Reconciliation Metadata"

11.2.1 Registering the Connector Bundle with .NET Connector Server

For registering or deploying the connector bundle on .NET Connector Server, perform the following steps:

1. Install the .NET Connector Server. See Section 9.6.2.1, "Installing the .NET Connector Server" for more information about installing the .NET Connector Server.
2. Stop the Connector Server. Make sure that Connector Server Service is not running.
3. Copy the connector DLL in the CONNECTOR_SERVER_HOME location. CONNECTOR_SERVER_HOME is the location where ConnectorServer.exe and other connector server related files are present after .NET Connector Server installation.
4. Start the .NET Connector Server.

11.2.2 Creating Basic Identity Connector Metadata

This metadata configuration is needed for both provisioning and reconciliation. Perform the following procedures by using the Oracle Identity Manager Design Console.

- Section 11.2.2.1, "Creating the IT Resource Type Definition"
- Section 11.2.2.2, "Creating the Resource Object"

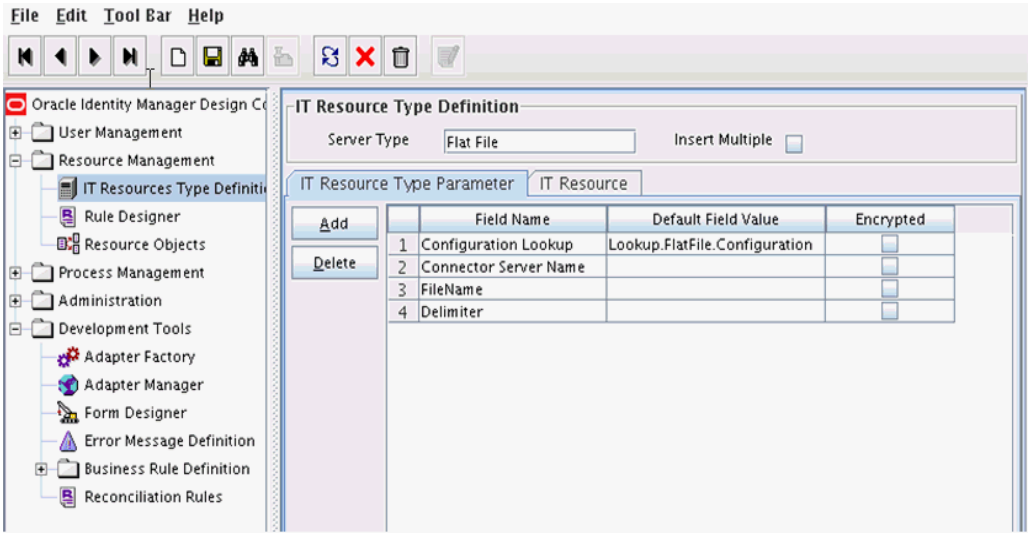
11.2.2.1 Creating the IT Resource Type Definition

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 11–1.
   - 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.
   - Delimiter maps to the Delimiter parameter in the bundle configuration. The value of this parameter will be passed.
   - FileName maps to the FileName parameter in the bundle configuration. The value of this parameter will be passed.
   - Connector Server Name, provide the connector server IT Resource name where .NET Connector Server is running.

Figure 11–1 IT Resource Type Definition in Design Console



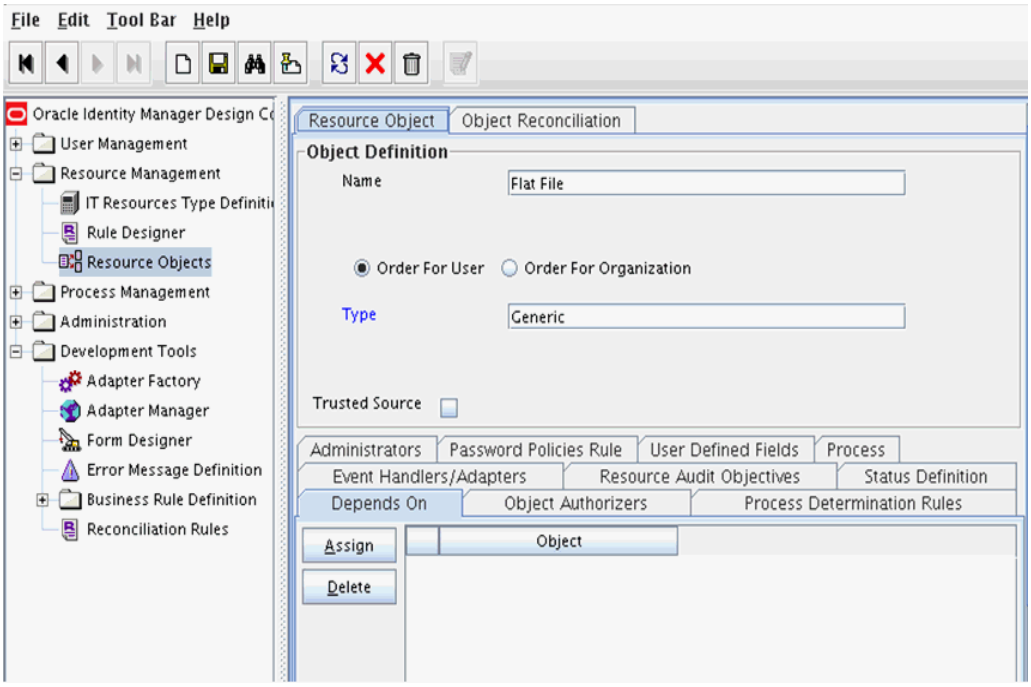
11.2.2.2 Creating the Resource Object

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 **Flat File**.

   As the resource object is a target resource, do not check the Trusted Source box as illustrated in Figure 11–2.

Figure 11–2 Resource Objects in Design Console



11.2.2.3 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.

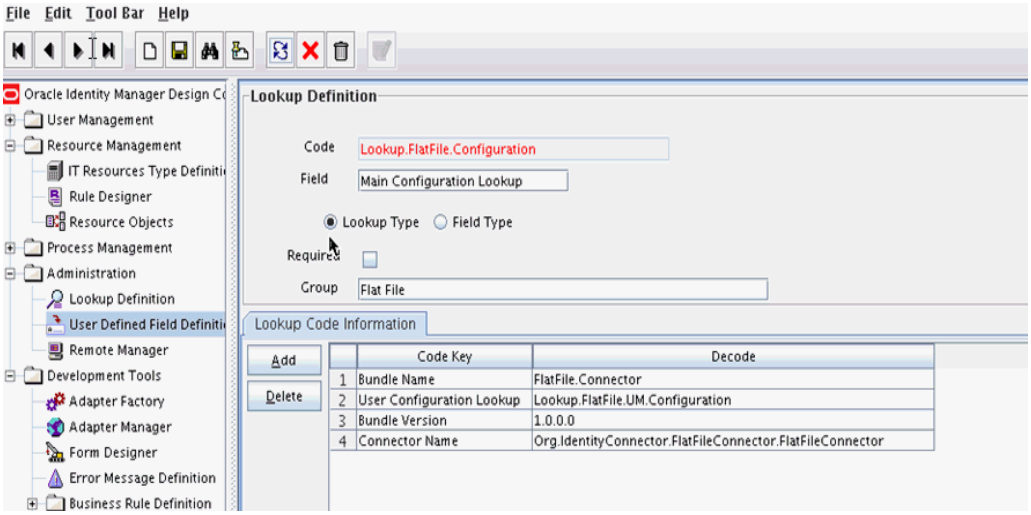
- Creating the Main Configuration Lookup
- Creating Object Type Configuration Lookup

11.2.2.3.1 Creating the Main Configuration Lookup The Configuration Lookup (as defined in Section 11.2.2.1, "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 11.2.2.1, "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.FlatFile.Configuration as the value for Code.
4. Add the following Lookup Code Information as illustrated in Figure 11–3.
   - Add *AssemblyVersion* as the required Bundle Version.
   - Add **FlatFile.Connector** as the required Bundle Name. The bundle name can be identified from the connector dll name. Connector DLL is in BUNDLE_NAME.dll format.
   - Add **Org.IdentityConnector.FlatFileConnector.FlatFileConnector** as the required Connector Name.
   - 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 11–3 Lookup Definition in Design Console



11.2.2.3.2 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 11.2.2.3.1, "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.

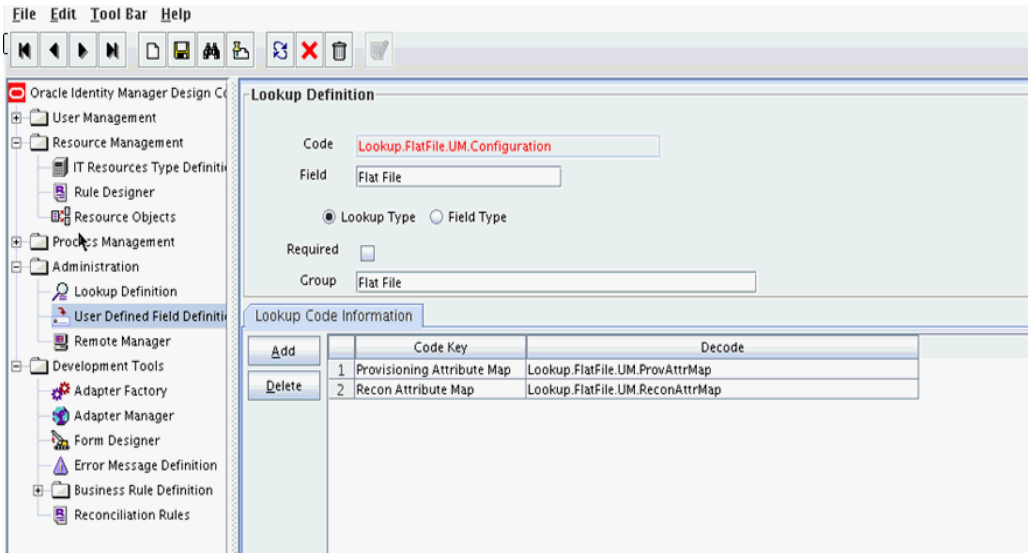
---

1. Log in to the Oracle Identity Manager Design Console.
2. Click Lookup Definition under Administration.
3. Create a new Lookup and add Lookup.FlatFile.UM.Configuration as the Code.
4. Set the following attributes as illustrated in Figure 11–4.

Note: This tutorial focuses on the minimum configurations needed to run an identity connector.

- **Provisioning Attribute Map** takes a value of Lookup.FlatFile.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.FlatFile.UM.ReconAttributeMap. This lookup contains the mapping between Oracle Identity Manager reconciliation fields and identity connector attributes. The mapping is used during reconciliation.

Figure 11–4 Second Lookup Definition in Design Console



11.2.3 Creating Provisioning Metadata

The following sections should be followed in order to configure Oracle Identity Manager for flat file provisioning.

- Section 11.2.3.1, "Creating a Process Form"
- Section 11.2.3.2, "Creating Adapters"
- Section 11.2.3.3, "Creating A Process Definition"
- Section 11.2.3.4, "Creating a Provisioning Attribute Mapping Lookup"

11.2.3.1 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.

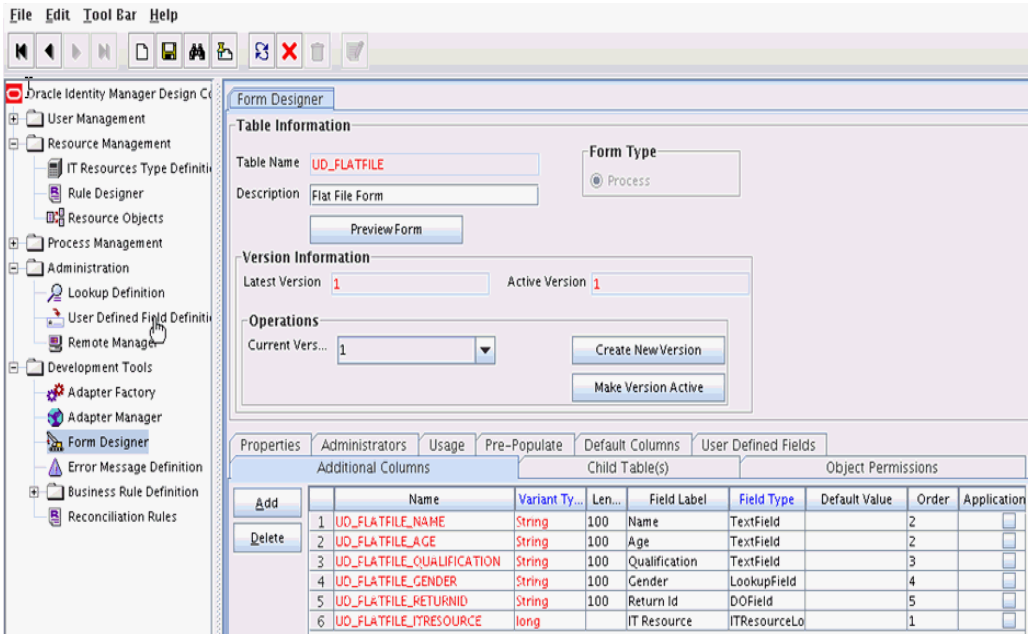
Attributes defined in the process form are not conventions. The form is a way to challenge the attributes that need to be passed to the identity connector. In general, define an attribute for each supported attribute in the identity connector.

> **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_FLATFILE as illustrated in Figure 11–5.

Figure 11–5 Form Designer in Design Console



4. Add the attributes defined in the connector schema, as listed in Table 11–1.

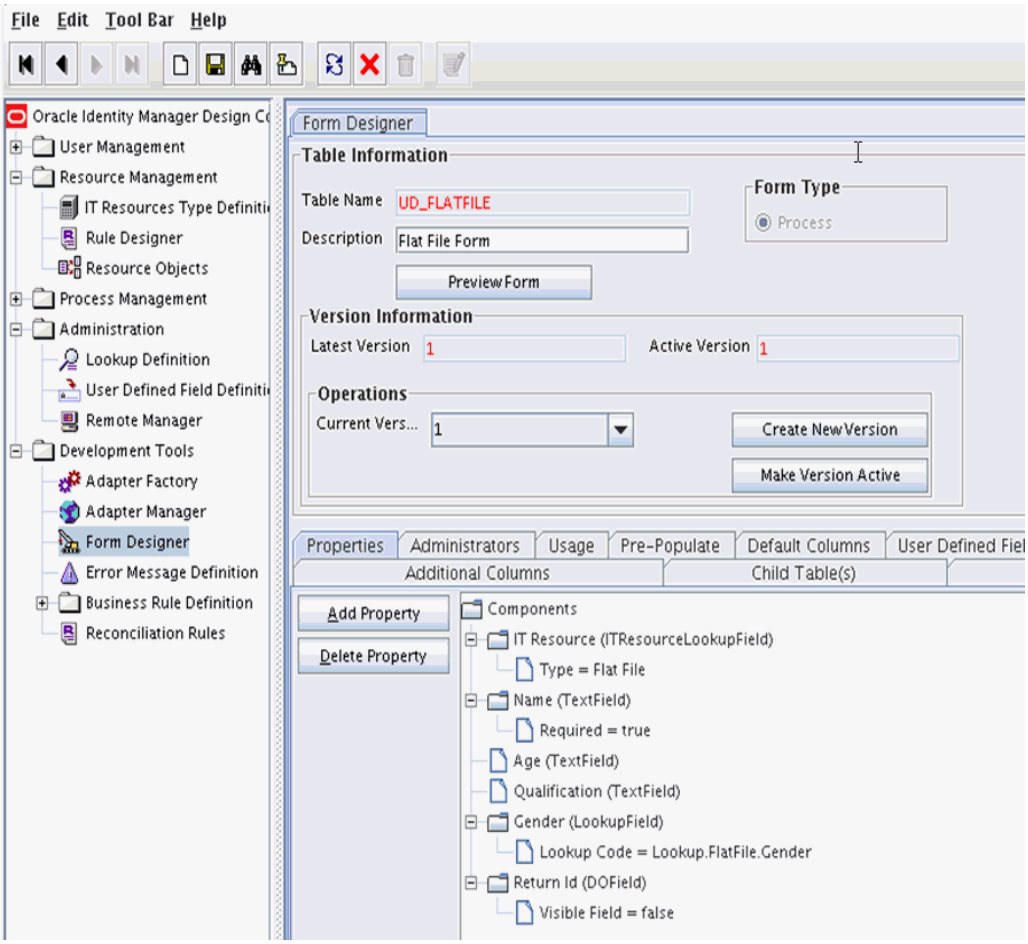
Table 11–1 Form Designer Fields

Name	Variant	Field Label	Field Type
UD_FLATFILE_NAME	String	Name	TextField
UD_FLATFILE_AGE	String	Age	TextField
UD_FLATFILE_ QUALIFICATION	String	Qualification	TextField
UD_FLATFILE_GENDER	String	Gender	LookupField
UD_FLATFILE_RETURNIDQ	String	Return Id	DOField
UD_FLATFILE_ITRESOURCE	Long	IT Resource	ITResourceLooku p

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 11–6.
   - Required = true
   - Type = Flat File

Figure 11–6 Properties of Form Designer in Design Console



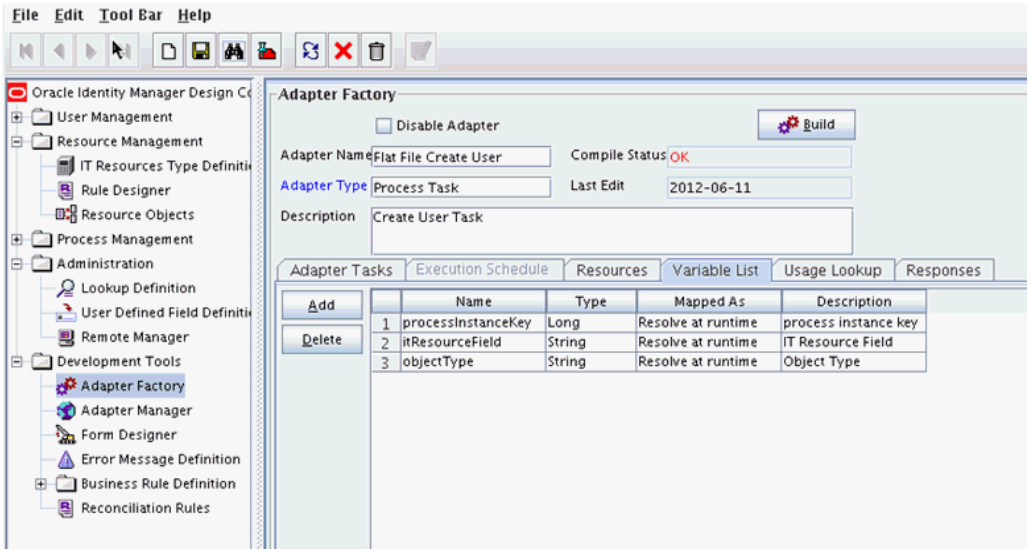
7. Save the form.
8. Click Make Version Active.

11.2.3.2 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 Flat File Create User 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 11–7.
   - 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 11–7 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 11–8.

   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.

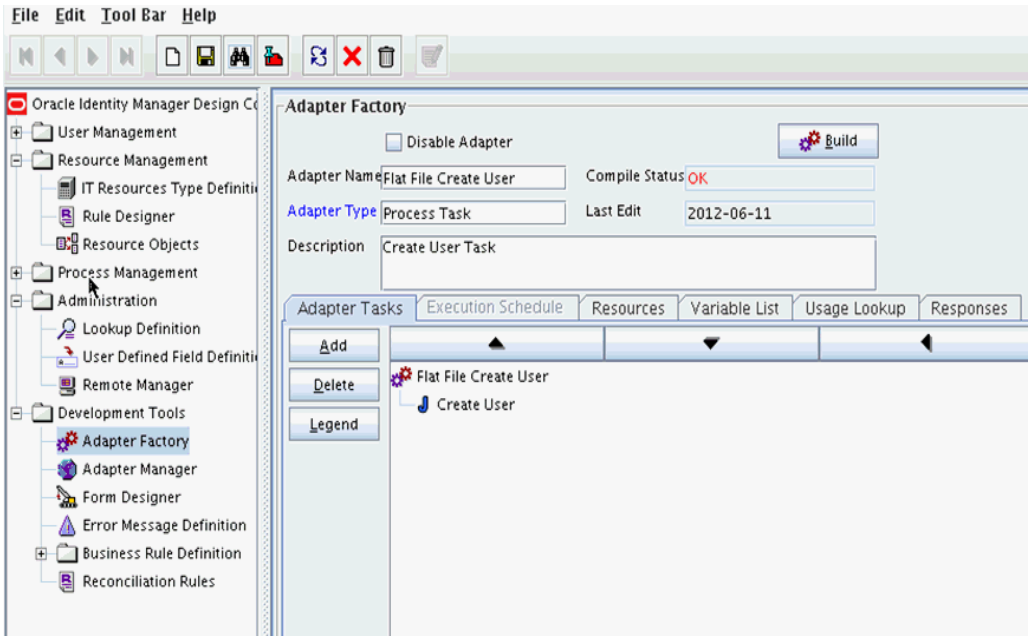
   g. Save the configurations.

   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 11–8 Adapter Factory in Design Console



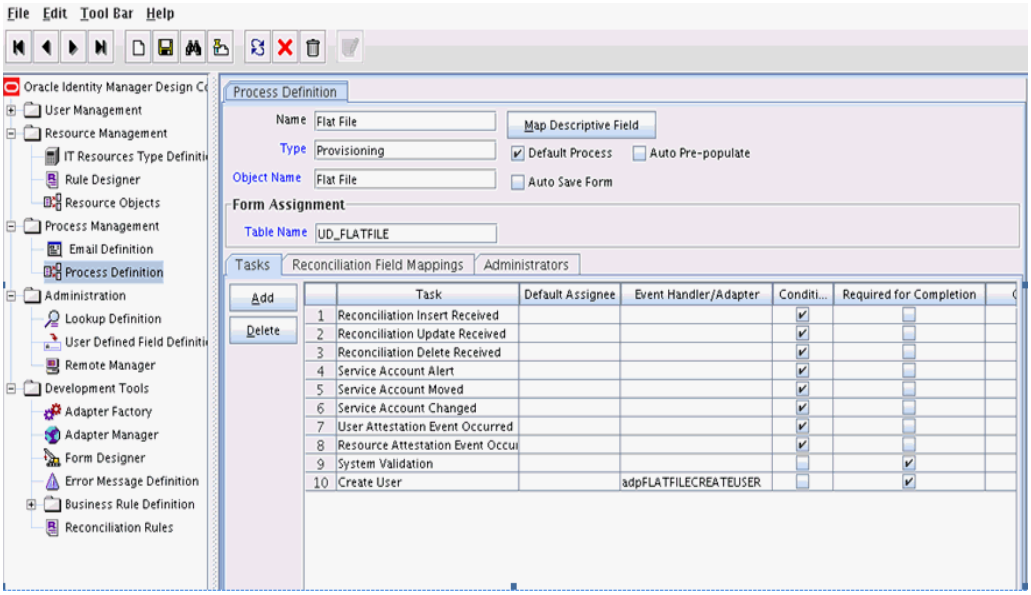
8. Save and build the adapter.

11.2.3.3 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.

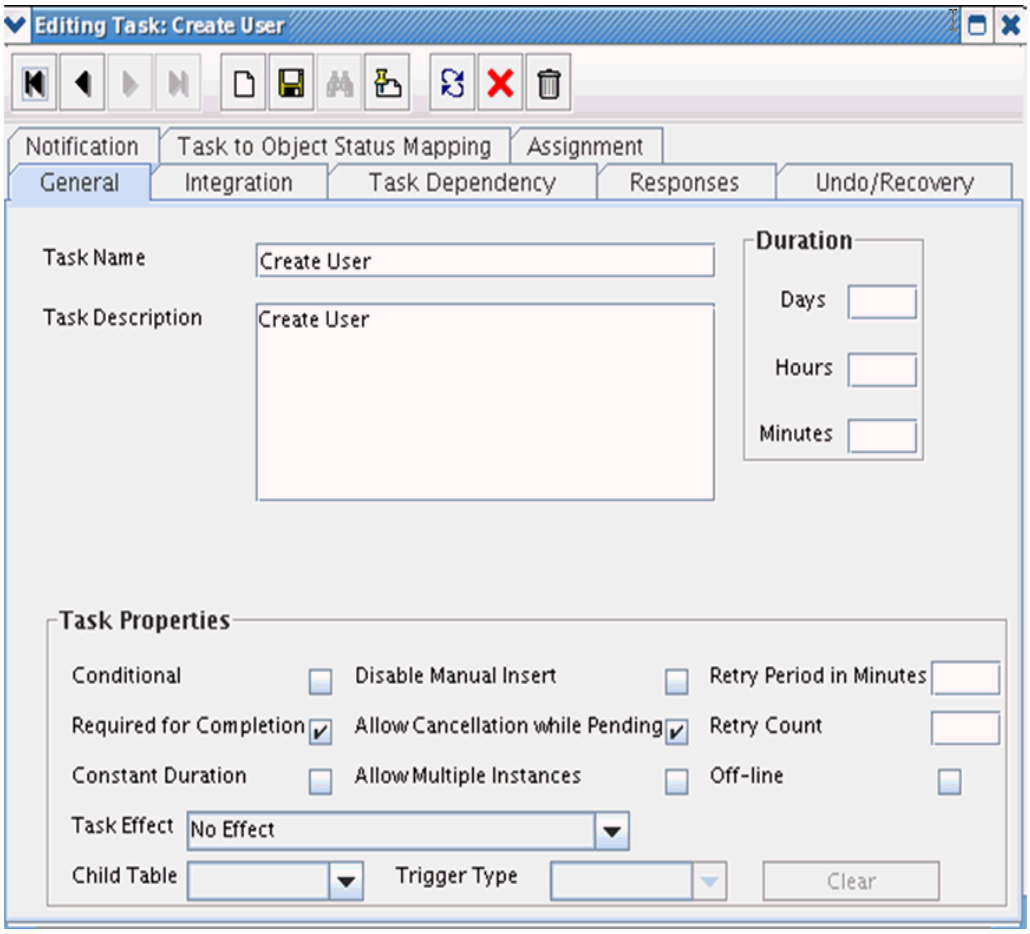
1. Log in to the Oracle Identity Manager Design Console.
2. Click Process Definition under the Process Management tab.
3. Create a new process definition and name it Flat File as illustrated in Figure 11–9.

Figure 11–9 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, Flat File.
6. Provide the process form Table Name; in this example, UD_FLATFILE.
7. Add a process task and name it Create User.
8. Double click **Create User** to edit as illustrated in Figure 11–10.

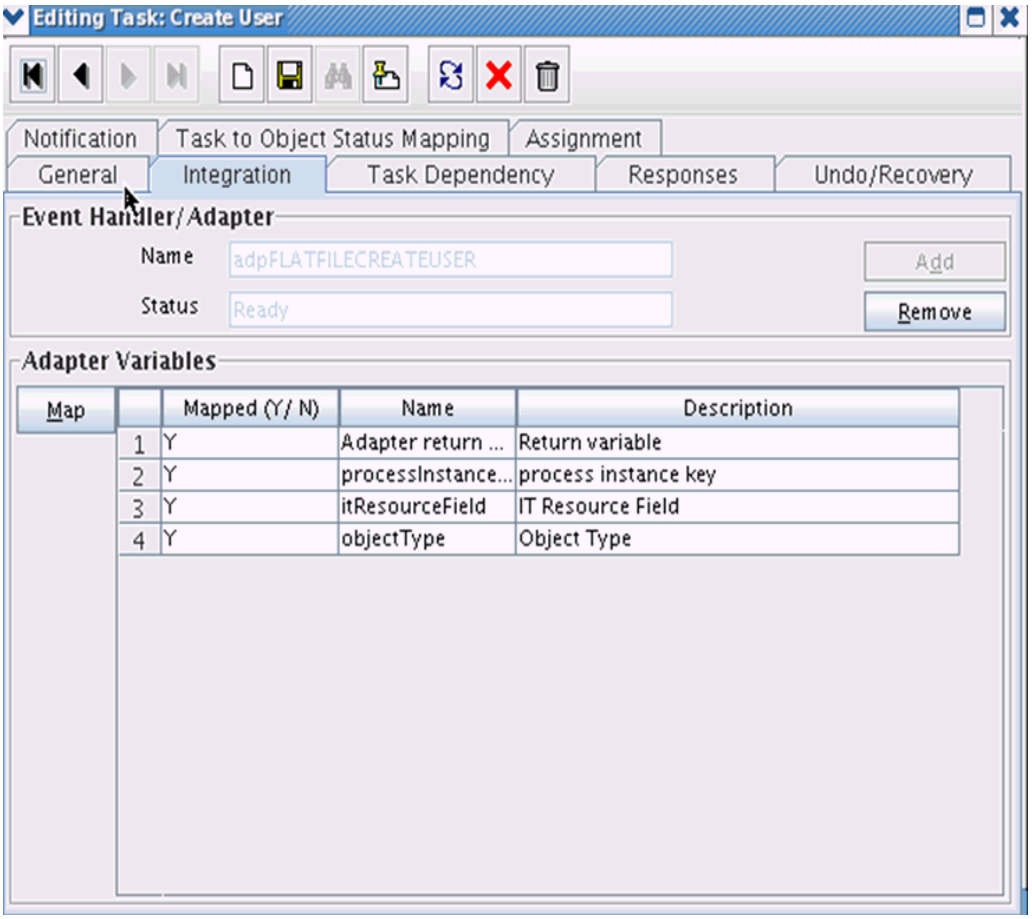
Figure 11–10 Editing Task Screen in Design Console



9. Click the **Integration** tab.
10. Click Add and select the adpFLATFILECREATEUSER from the list as illustrated in Figure 11–11.

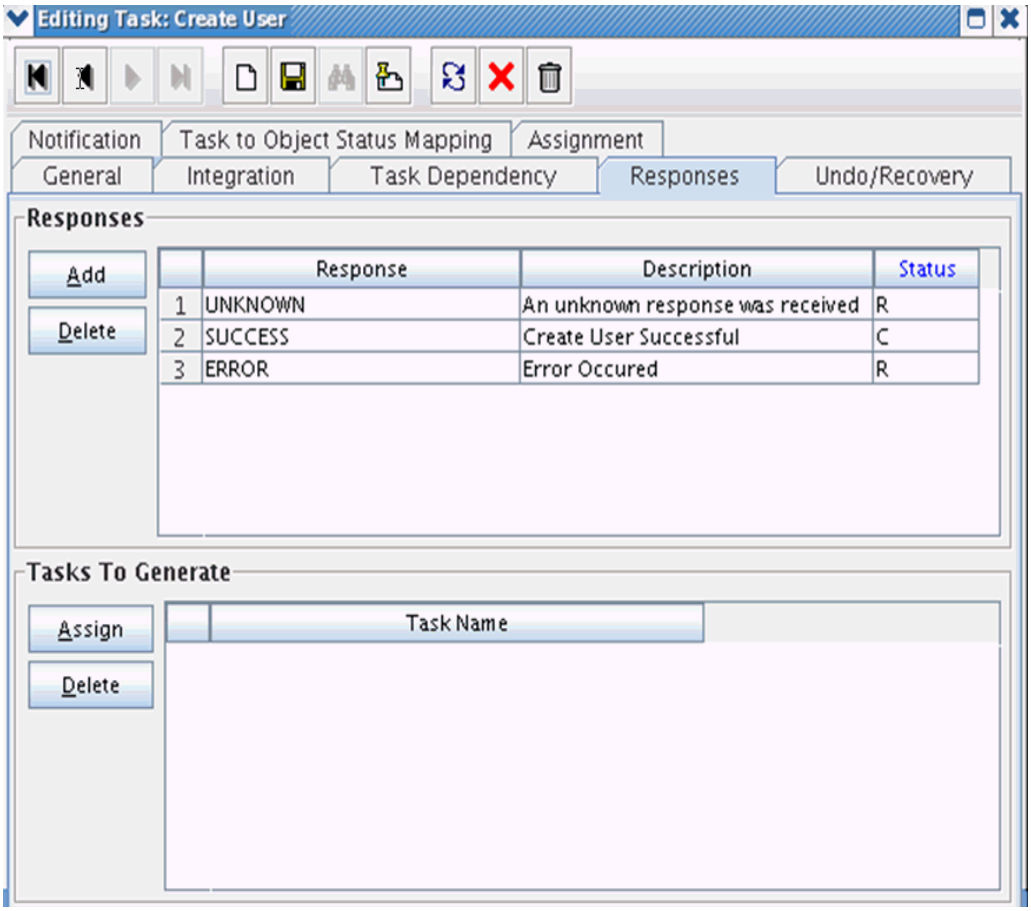
    The adapter will be available only after it is compiled.

Figure 11–11 Integration Tab in Design Console



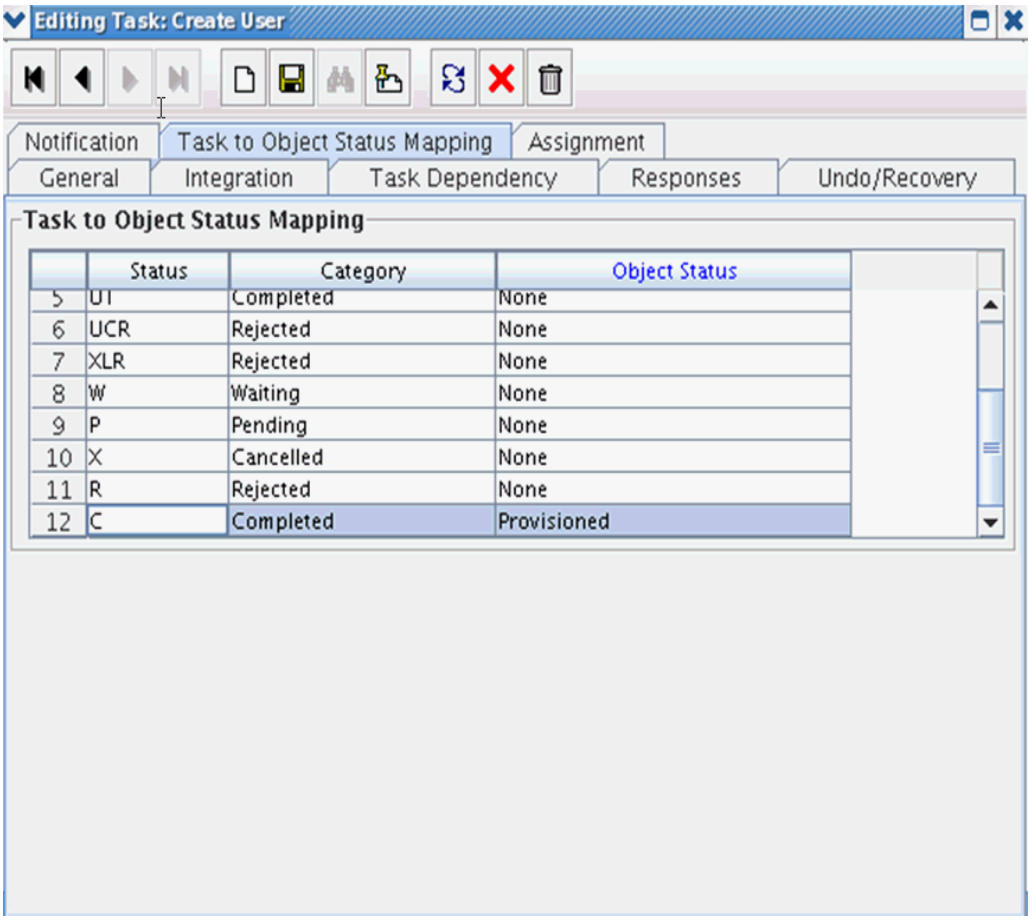
11. Map the variables as follows to set the response code returned by the identity connector.
    - Adapter Return Variable – Response Code
    - Object Type – [Literal:String] User (Name of the object type)
    - Process Instance Key – [Process Data] Process Instance
    - IT Resource Field Name – [Literal:String] UD_FLATFILE_ITRESOURCE (Form field name that contains the IT resource information)
12. Click the Responses tab and configure the responses as illustrated in Figure 11–12.
    - 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 11–12 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 11–13:

Figure 11–13 Task to Object Status Mapping



15. Save the process task.

11.2.3.4 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.

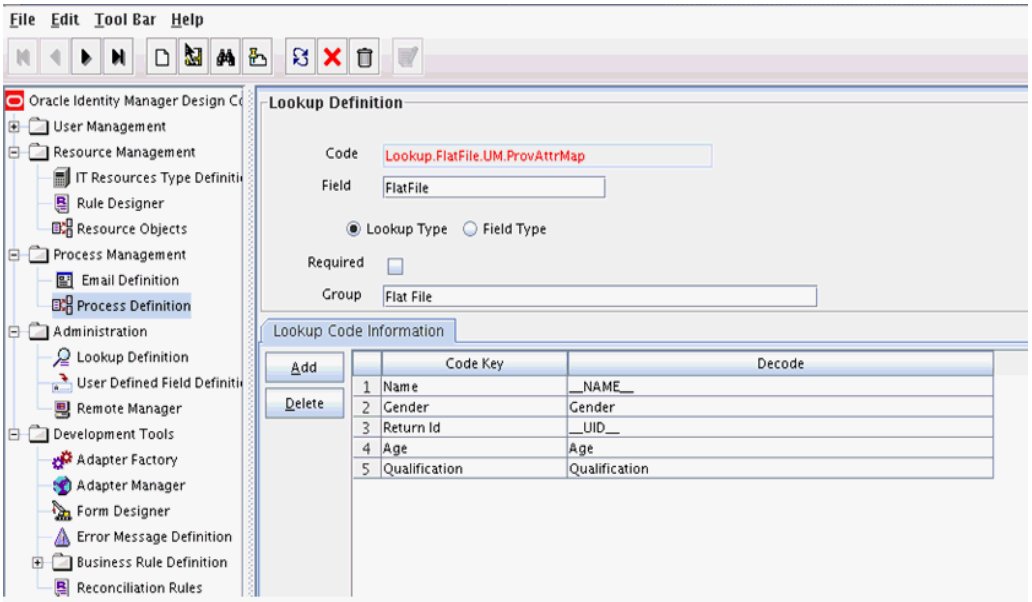
Following is the procedure to create a Provisioning Attribute Mapping Lookup.

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.FlatFile.UM.ProvAttrMap.

   The name of this lookup is referred from the object type configuration lookup. See Section 11.2.2.3.2, "Creating Object Type Configuration Lookup."

4. Add the form Field Labels as the code keys and identity connector bundle attributes as the decode as shown in Figure 11–14.
   - Name : __NAME__
   - Gender: Gender
   - Return Id: __UID__
   - Age: Age
   - Qualification: Qualification

Figure 11–14 Lookup Code Mapping



11.2.3.4.1 Field Flags Used in the Provisioning Attributes Map

> **Note:** These properties are advanced options and can be skipped for the current implementation of the connector.

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].

11.2.4 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.

- Section 11.2.4.1, "Creating a Reconciliation Schedule Task"
- Section 11.2.4.2, "Creating a Reconciliation Profile"
- Section 11.2.4.3, "Setting a Reconciliation Action Rule"
- Section 11.2.4.4, "Creating Reconciliation Mapping"
- Section 11.2.4.5, "Defining a Reconciliation Matching Rule"

11.2.4.1 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. Section 11.2.4.1.1, "Defining the Schedule Task"
2. Section 11.2.4.1.2, "Creating a Scheduled Job"

11.2.4.1.1 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 11–4. Make sure to update database value to your database.

Example 11–4 Deployment Manager XML with Scheduled Task Details

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<xl-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>
```

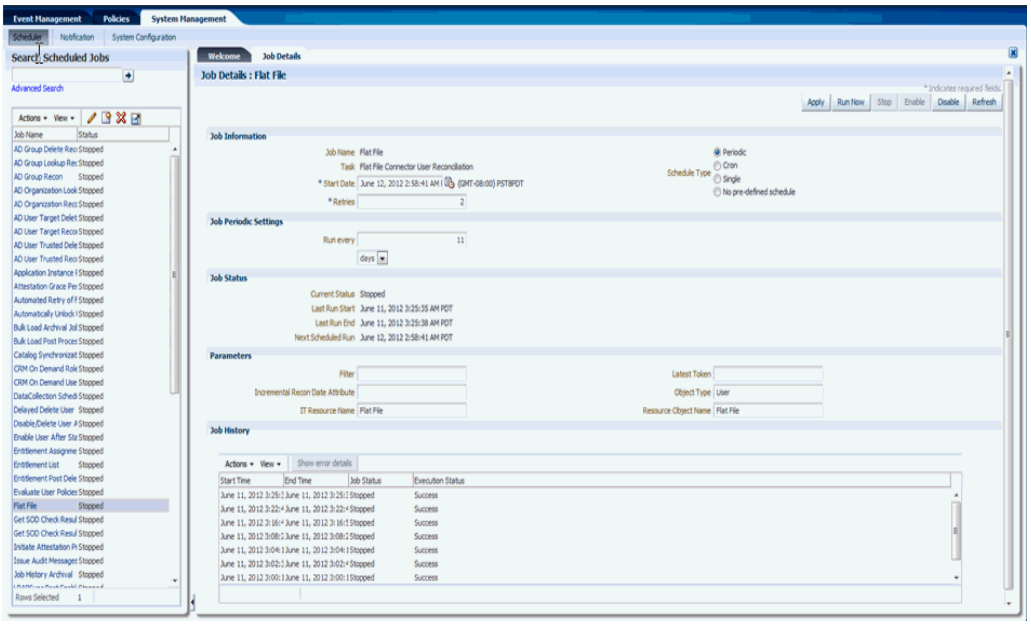
2. Save the file as Flat File Connector User Reconciliation.xml.
3. Login into the Identity System Administration. Under System Management, click **Import**.
4. Select the Flat File Connector User Reconciliation.xml file, and click **Import**.
5. Complete the steps in the wizard.

11.2.4.1.2 Creating a Scheduled Job 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. Click **New** for creating a new scheduled job. After that provide the job name as **Flat File** and in the Task field, select the value as **Flat File Connector User Reconciliation** from the lookup. Once the job is created, provide the values in the job as shown in Figure 11–15.

4. Add a scheduled task and add Flat File Connector User Reconciliation as the type as illustrated in Figure 11–15.

Figure 11–15 Scheduled Task Screen in Advanced Console



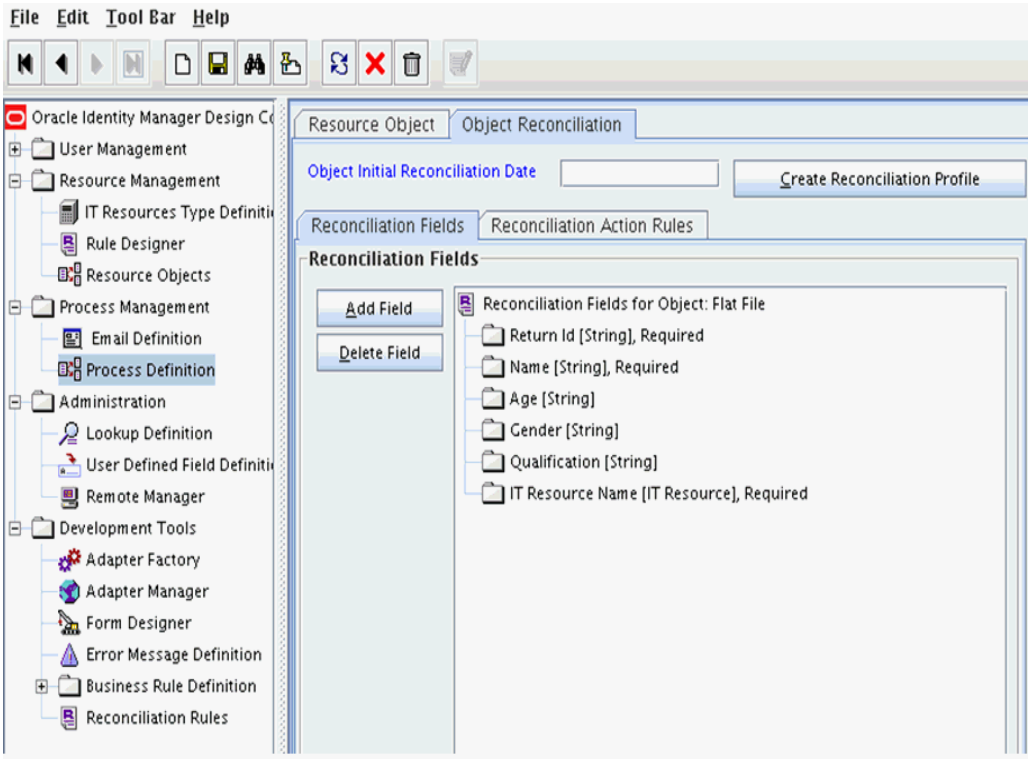
5. Set the parameters as follows:
   - IT Resource Name takes a value of Flat File.
   - Resource Object Name takes a value of FLATFILE.
   - Object Type takes a value of User.
6. Click **Apply**.

11.2.4.2 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 Flat File resource object.
4. Click the **Object Reconciliation** tab as illustrated in Figure 11–16.

Figure 11–16 Object Reconciliation in Design Console



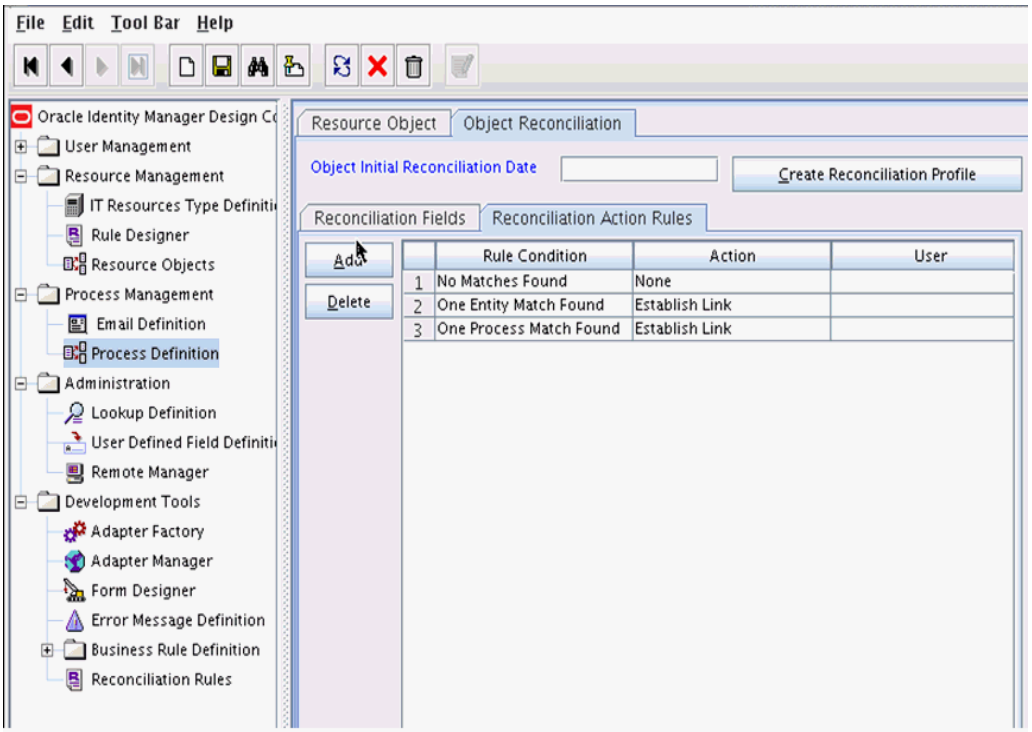
5. Add following reconciliation fields:
   - Return Id [String] , Required]
   - Name [String] , Required
   - Gender [String]
   - Age [String]
   - Gender [String]
   - IT Resource Name [IT Resource] , Required
6. Save the configuration.

11.2.4.3 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 Flat File resource object.
2. Click the **Object Reconciliation** tab.
3. Click the **Reconciliation Action Rules** tab in the right frame.

Figure 11–17 Reconciliation Action Rules in Design Console



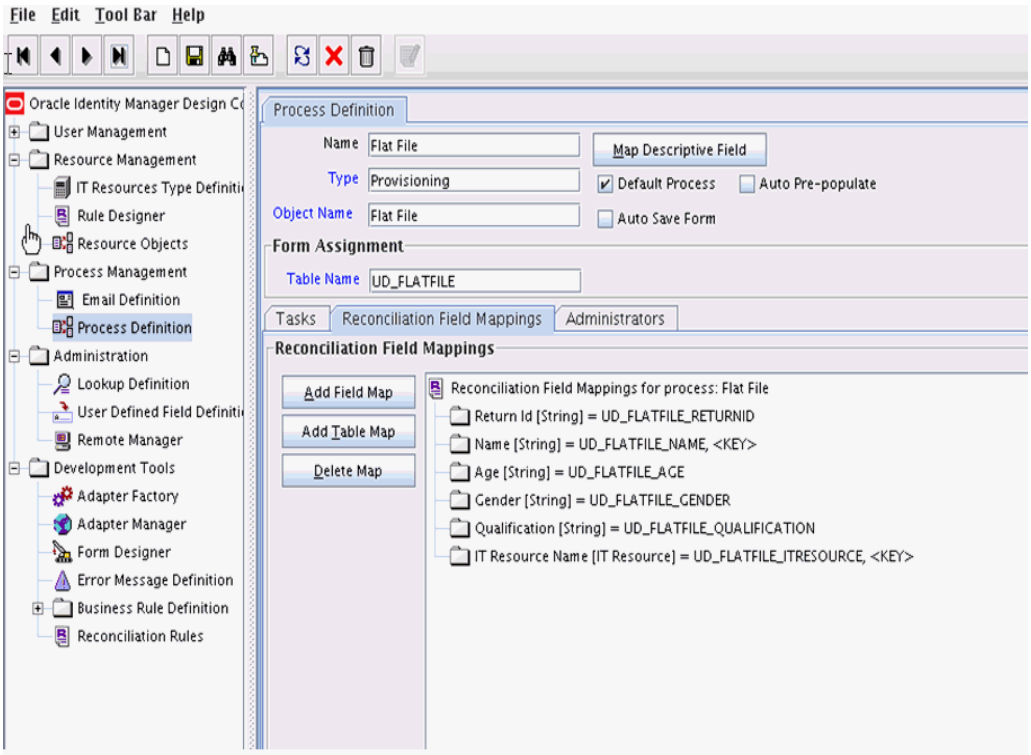
4. Add an action rule defined as One Process Match Found (Rule Condition) and Establish Link (Action).
5. Add an action rule defined as One Entity Match Found (Rule Condition) and Establish Link (Action).
6. Click **Create Reconciliation Profile**.
7. Click **Save**.

11.2.4.4 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 11–18.

Figure 11–18 Reconciliation Field Mapping in Design Console



5. Add mappings between the reconciliation profile fields and the process form fields.
   - ReturnId[String] = UD_FLATFILE_RETURNID
   - Name[String] = UD_FLATFILE_NAME, <KEY>
   - Age[String] = UD_FLATFILE_AGE
   - Gender[String] = UD_FLATFILE_GENDER
   - Qualification[String] = UD_FLATFILE_QUALIFICATION
   - IT Resource Name[IT Resource] = UD_FLATFILE_ITRESOURCE,<KEY>
6. Save the configuration.

11.2.4.4.1 Field Flags Used in the Reconciliation Attributes Map 9

> **Note:** These properties are advanced options and can be skipped for the current implementation of the connector

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].

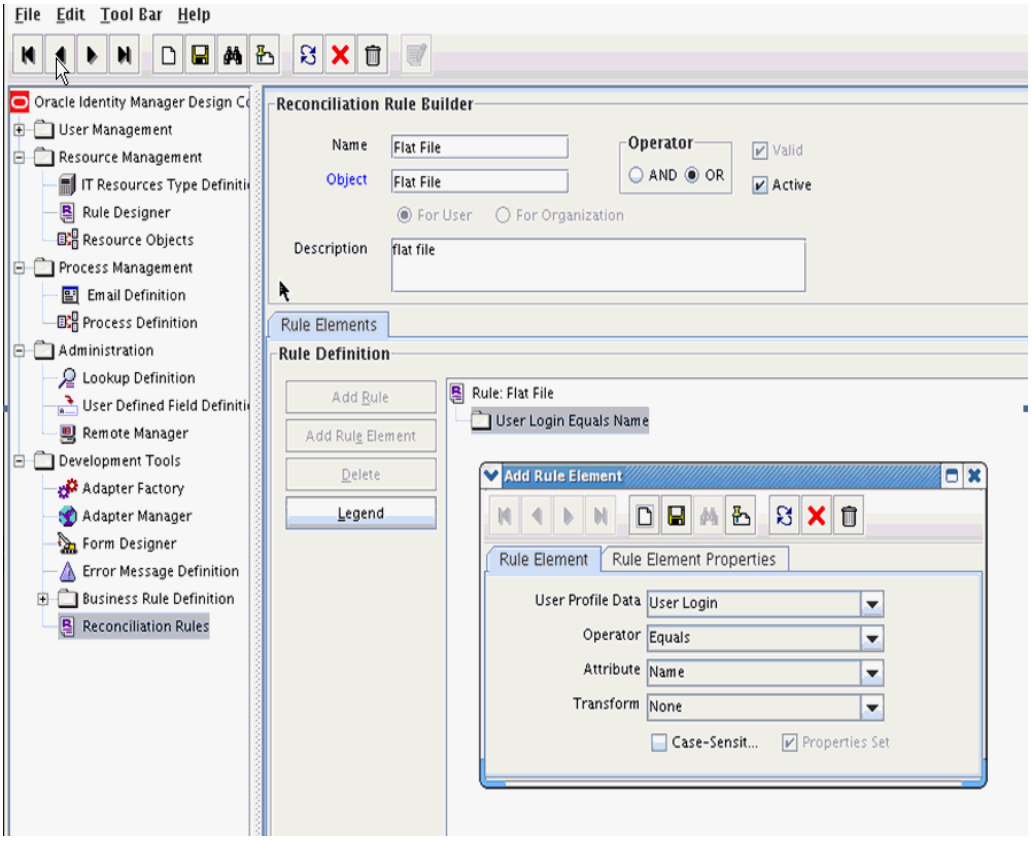
- **DATE:** In Recon Attribute Map, this must be specified for fields whose type need to be considered as Date. For example, the code key for a field with label Today whose value needs to be displayed in the date format must look similar to Today[DATE].

11.2.4.5 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 11–19 Adding Reconciliation Matching Rule



4. Select resource object Flat File.
5. Once the reconciliation rule element is added, make sure to check Active flag so that the reconciliation rule is made active.
6. Save and add the rule element.

   User Login from the user profile data equals the Name resource attribute.
7. Save the rule.

> **Note:** You must recreate the reconciliation profile whenever you make any changes to the reconciliation rule.

11.3 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.FlatFile.Configuration as appropriate.

12

Integrating ICF with Oracle Identity Manager

Oracle Identity Manager's goal is to manage the business logic of Identity administration, and delegate the execution of provisioning and reconciliation operations to Identity Connector Framework (ICF). ICF with Oracle Identity Manager (OIM) unites all the scheduled tasks and the provisioning tasks for all ICF based connectors.

This chapter contains conceptual information about ICF-OIM integration in the sections:

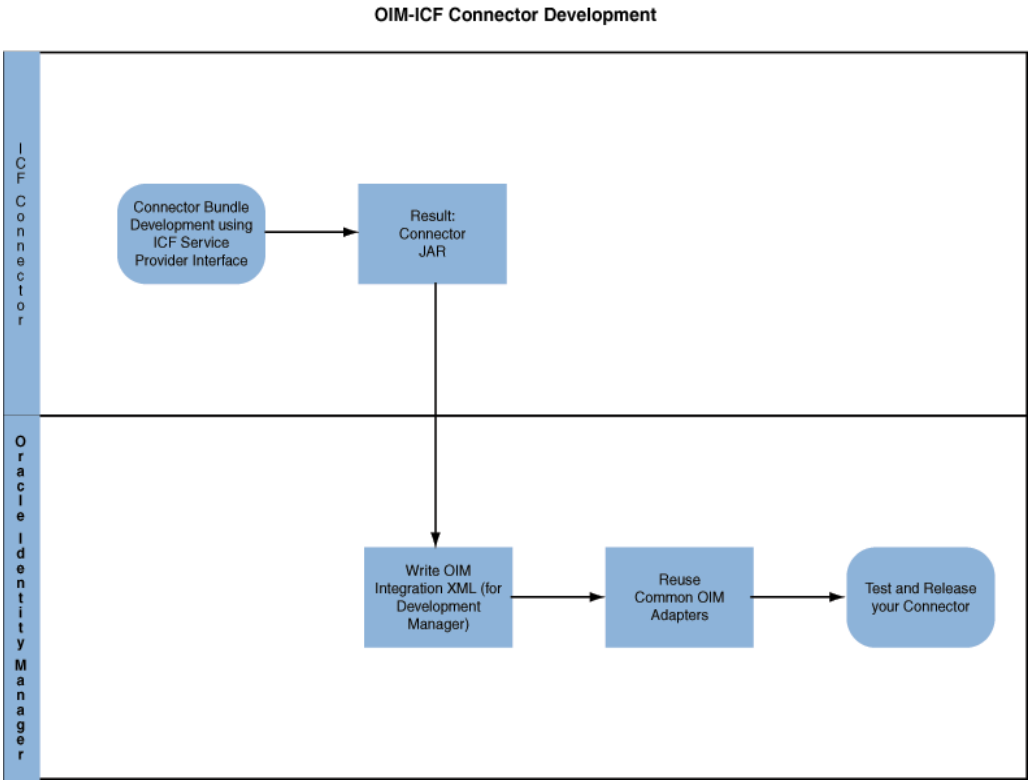
12.1 ICF Common

OIM ICF Integration Layer is an implementation of ICF API on one side and invokes OIM APIs (icf-oim-intg.jar) on the other side. This reduces the complexity of the connector developer as it provides API abstraction. It also support provisioning and reconciliation operations. See Section 12.5, "Provisioning" and Section 12.6, "Concepts of Reconciliation in ICF Common" for more information about provisioning and reconciliation using ICF Common.

12.2 Integration Architecture

Figure 12–1 is the ICF-OIM integration architecture.

Figure 12–1 OIM-ICF Connector Development Architecture



12.3 Global Oracle Identity Manager Lookups

Lookups is used to store OIM configuration metadata. IT Resource parameter Configuration Lookup points to main Configuration Lookup that encapsulates all the Oracle Identity Manager specific configuration information.

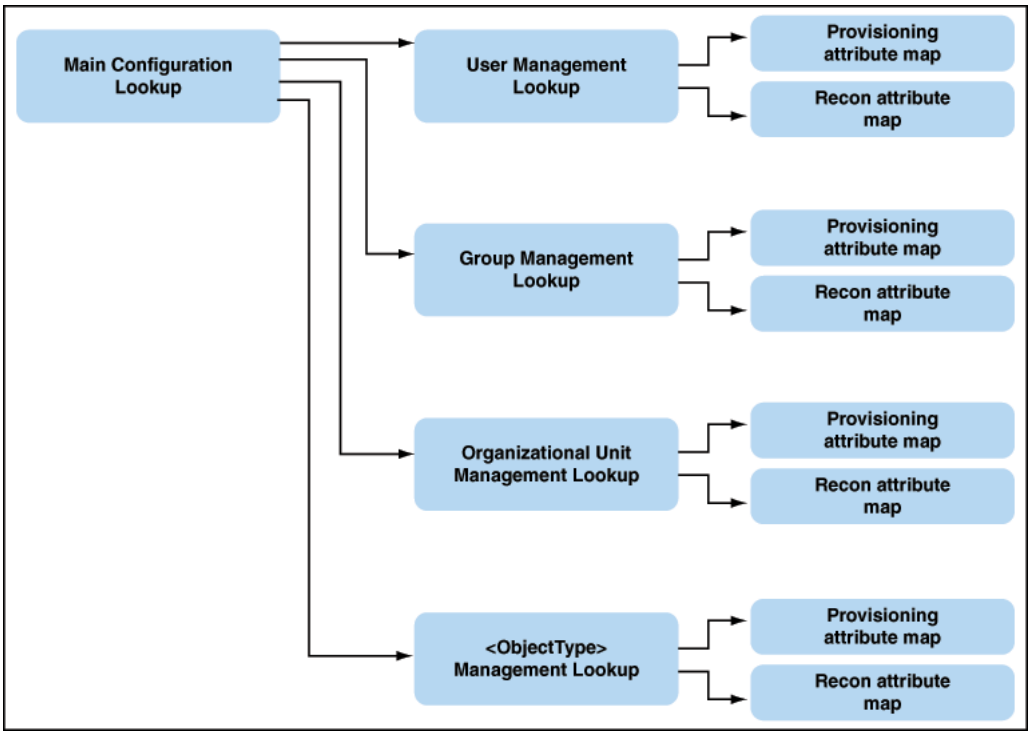
Based on the lookup configuration, you can classify your properties into the following three classes:

- IT Resource: connectivity properties: contains all properties that are used for making a connection to the target system.
- Main Configuration Lookup Configuration Properties: contains non-connectivity properties that alter the mode of reconciliation or provisioning, and are not required for connection. There is a thin line of difference between connectivity and configuration properties, therefore one property can be assigned to both of them.
- Object Type: specific lookups (for example, user management configuration), mapping lookups for specific object type (for example, User, Group, Organizational Unit).

Note: LOADFROMURL flag can be used in IT Resource or Main Configuration Lookup in the code (key) field, for example, sampleProperty[LOADFROMURL]. For properties marked as this, the value (decode value) is a URL. ICF integration will read the contents of the file stored in the given URL and use it as the value of given property at runtime. This is useful for large values that cannot fit directly into a lookup.

Figure 12–2 illustrates the global Oracle Identity Manager lookups from which most of the Connectors use the User Management Lookups.

Figure 12–2 Oracle Identity Manager Connector Lookup Hierarchy



This section discusses the following topics:

- Section 12.3.1, "Main Lookup Configuration"
- Section 12.3.2, "User Management Configuration"
- Section 12.3.3, "Recon Transformation Lookup (Lookup.CONNECTOR_NAME.UM.ReconTransformation)"
- Section 12.3.4, "Recon Validation Lookup (Lookup.CONNECTOR_NAME.UM.ReconValidation)"
- Section 12.3.5, "Optional Defaults Lookup"

12.3.1 Main Lookup Configuration

IT Resource parameter Configuration Lookup points to Main Configuration Lookup, which encapsulates all the OIM specific configuration information.

Configuration lookup, denoted as Lookup.CONNECTOR_NAME.Configuration, is the top level entry that refers to subordinate lookups for reconciliation and provisioning. The configuration lookup has the following structure:

Table 12–1 Lookup Configuration for Connector

Configuration Key	Value	Description
Connector Name	org.identityconnectors.CONNECTOR_NAME.Connector	Identity Connector Main Class. This is the class that implements SPI operations of ICF framework.

Table 12–1 (Cont.) Lookup Configuration for Connector

Configuration Key	Value	Description
Bundle Name	org.identityconnectors.CONNECTOR_NAME	Identity Connector bundle name
Bundle Version	11.1.1.5.x	Identity Connector bundle version
User Configuration Lookup **Note**: Other object types may be defined, for example, for Generic LDAP connector: Group Configuration Lookup, OU Configuration Lookup.	Lookup.CONNECTOR_NAME.UM.Configuration	Link to User specific configuration lookup. **Note**: User should be the object type. If you need to support any other object type, you can use OBJECT_TYPE Configuration Lookup as the key.

12.3.2 User Management Configuration

These lookups control the mapping for provisioning and reconciliation. In addition, these lookups might also configure transformation and validation.

This lookup contains the following keys:

- Before Create Action Language: This key if present in the Lookup.CONNECTOR_NAME.UM.Configuration, which informs ICF that there is a script whose language is the value of this key. The value of this key (Groovy/cmd) informs the language of the script that needs to be executed by ICF before create operation.
- Before Create Action File: This key if present in the Lookup.CONNECTOR_NAME.UM.Configuration, informs ICF that a script represented by the value of this key needs to be executed by ICF before create action. This script must be accessible to Oracle Identity Manager Server.
- Before Create Action Target: This key if present in the Lookup.CONNECTOR_NAME.UM.Configuration, informs ICF that script as defined by previous two keys must be executed either on resource or on connector. Depending on this configuration the ICF API runScriptOnConnector or runScriptOnResource will be executed.

Table 12–1 describes the User Management lookup configuration.

Table 12–2 User Management Lookup Configuration for Connector

Configuration Key	Value	Mandatory Field Type	Description
Provisioning Attribute Map	Lookup.CONNECTOR_NAME.UM.ProvAttrMap	Y	This lookup contains the mapping between Oracle Identity Manager fields and identity connector attributes. The mapping is used during provisioning.
Recon Attribute Map	Lookup.CONNECTOR_NAME.UM.ReconAttrMap	Y	This lookup contains the mapping between Oracle Identity Manager reconciliation fields and identity connector attributes. The mapping is used during reconciliation.

Table 12–2 (Cont.) User Management Lookup Configuration for Connector

Configuration Key	Value	Mandatory Field Type	Description
Recon Attribute Defaults	Lookup.CONNECTOR_NAME.UM.ReconDefaults	N	This mapping contains the default values for OIM attributes, that are substituted, if no value is provided by connector during reconciliation.
Recon Transformation Lookup	Lookup.CONNECTOR_NAME.UM.ReconTransformation	N	Lookup for Transformation by doing Reconciliation Task. Transformation is used in all Reconciliation Tasks except LookupReconTask.
Recon Validation Lookup	Lookup.CONNECTOR_NAME.UM.ReconValidation	N	Lookup used for Validation by running Reconciliation Task. Validation is used in all Reconciliation Tasks except LookupReconTask.
Recon Exclusion List	Lookup.CONNECTOR_NAME.UM.ReconExclusionList	N	Exclusion list is a way to address un-managed accounts for the connector. While reconciliation/provisioning. Any match from the exclusion list will not be processed by OIM. There are two types of rules supported by the exclusion list: ■ Matching rules **Direct Matching Rule** Code Key: Reconciliation field name Decode Key: Excluded field value ■ Pattern Matching Rule Suffix with [PATTERN] tag to enable pattern matching Code Key: ReconFieldName[PATTERN] Decode Key: Exclusion pattern Exclusion patterns should follow the nomenclature defined in java.util.regex.Pattern See the Recon Exclusion List key in this table.
Provisioning Exclusion List	Lookup.CONNECTOR_NAME.UM.ProvExclusionList	N	In provisioning, code key is the Form label name, and decode key is the excluded value/pattern.

Table 12–2 (Cont.) User Management Lookup Configuration for Connector

Configuration Key	Value	Mandatory Field Type	Description
Provisioning Validation Lookup	Lookup.CONNECTOR_NAME.UM.ProvValidation	N	Lookup for Validation by Provisioning. ICF defines the concept of OperationOption, it is an extra parameter list, that can be sent to any operation. It is up to the connector implementation to define the use of these operation options.
Operation Options Map	Lookup.CONNECTOR_NAME.UM.OperationOptions	N	The code key is a constant Operation Options Map. The decode value name of lookup that will be used as a map of operation options. For example, in Lookup.Domino.UM.OperationOptions the code key is CACertifier[UPDATE,DELETE] and the decode value is CACertifier, which means that this attribute will be sent to calls of Update and Delete operations as an extra operation option. If you want to configure the action run, then you need to provide three parameters for scripting: ■ Language ■ File ■ Target
Scripting Attributes			The triggering time of the script is controlled by these labels in your lookup key: ■ Before ■ After The provisioning operation type that the script is attached on is controlled by these labels: ■ Create ■ Update ■ Delete
Before Create Action Language	SCRIPTING_LANGUAGE_NAME	N	Language of the Action which will be executed, for example, Groovy/cmd. If you want to configure the action run, then you need to provide three options, Language/File/Target You can configure Before/After actions for the following provisioning operations: Create/Update/Delete.

Table 12–2 (Cont.) User Management Lookup Configuration for Connector

Configuration Key	Value	Mandatory Field Type	Description
Before Create Action File	FILE_PATH	N	File containing script which needs to be executed. This file needs to be accessible to Oracle Identity Manager Server.
Before Create Action Target	Connector or Resource	N	Target of the action, can be Connector or Resource. Depending on this configuration the ICF API runScriptOnConnector or runScriptOnResource will be used.

12.3.3 Recon Transformation Lookup (Lookup.CONNECTOR_NAME.UM.ReconTransformation)

Transformation code is in an external Oracle Identity Manager Java Task, used in all Reconciliation Tasks except LookupReconTask. It is a Java class uploaded (transforming data coming from Target System during reconciliation) to Oracle Identity Manager repository.

The Java class performing transformation needs to have a method with the signature public Object transform(HashMap arg0, HashMap arg1, String arg2) implemented. ICF would look for this method with the exact signature.

Transform java class template is as follows:

```
public class MyTransformer implements
oracle.iam.connectors.common.transform.Transformation {
  public Object transform(java.util.HashMap hmUserDetails, java.util.HashMap
hmEntitlementDetails, String sField) {
    String sFirstName= (String)hmUserDetails.get("First Name");
    String sLastName= (String)hmUserDetails.get("Last Name");
    String sFullName=sFirstName+"."+sLastName;
    return sFullName;
  }
}
```

The name of lookup storing the Recon Transformation Lookup is defined in Main Configuration Lookup (Lookup.CONNECTOR_NAME.Configuration) as shown in Table 12–3.

Table 12–3 Reconciliation Transformation Lookup

Key	Value	Description
Recon Field Name	<transformationClassName> com.validationexample.MyTransform	Java class which performs transformation for this recon field.

12.3.4 Recon Validation Lookup (Lookup.CONNECTOR_NAME.UM.ReconValidation)

Validation code is in an external Oracle Identity Manager Java task, Used for validating data coming from Target System during Reconciliation. It is a Java class uploaded (transforming data coming from Target System during reconciliation) to Oracle Identity Manager repository.

The Java class performing validation needs to have a method with the signature public boolean validate (HashMap arg0, HashMap arg1, String arg2) implemented. ICF would look for this method with the exact signature.

The validation Java class template is as follows:

```
public class MyValidator implements
oracle.iam.connectors.common.validate.Validator {
  public boolean validate(java.util.HashMap hmUserDetails, java.util.HashMap
hmEntitlementDetails,String sField) throws
oracle.iam.connectors.common.ConnectorException {
    boolean isValid = false;
    // validation code goes HERE
    return isValid;
  }
}
```

The name of lookup storing the Recon Validation Lookup is defined in main configuration lookup (Lookup.CONNECTOR_NAME.Configuration) as shown in Table 12–4.

Table 12–4 Reconciliation Validation Lookup

Key	Value	Description
Recon Field Name	<transformationClassName>	Java class which performs validation for this recon field.
	com.validationexample.MyValidator	

12.3.5 Optional Defaults Lookup

Missing values for reconciliation are substituted by default values defined in the following table. User Type is a required OIM attribute, that typically is not contained on the target resource. You can set the default value in here.

For example, trusted reconciliation requires a set of attributes from the connector to have a non-empty value. However, not all resources can supply all of these attribute types, so we need to provide some default values. Table 12–5 lists all required attributes for reconciliation, and possible default values for them.

If connector can supply all attributes needed in reconciliation, then this table becomes optional.

Table 12–5 Lookup.CONNECTOR_NAME.UM.Recon.Defaults.Trusted Attributes

Key	Value
Last Name	CONNECTOR_DEPENDENT_VALUE
Organization	Xellerate users
User Type	End-User
Employee Type	Full-Time
First Name	CONNECTOR_DEPENDENT_VALUE

Note: These default values are supported only for single valued fields, which means the multivalued or child table attributes are not supported.

12.4 IT Resource

IT Resource contains connectivity parameters for Target System. These parameters are required for all the connectors using ICF integration.

Table 12–6 describes the common IT Resource parameters.

> **See Also:** The documentation for the connector you are deploying for information about the IT Resource parameters of the target system and the Connector Server

Table 12–6 IT Resource Parameter

Parameter	Description
Connector Server Name	IT Resource name of Connector Server. The IT Resource needs to be of type Connector Server. This field is a mandatory field, but the value is optional.
Configuration Lookup	Name of the main configuration lookup. This field is a mandatory field

12.5 Provisioning

The section contains the following topics:

- Section 12.5.1, "ICF Provisioning Manager"
- Section 12.5.2, "Provisioning Lookup"
- Section 12.5.3, "Non-User Object Types"
- Section 12.5.4, "Optional Lookups for Provisioning"
- Section 12.5.5, "Optional Flags in Lookups for Provisioning Attribute Map"
- Section 12.5.6, "Compound attributes in Provisioning Attribute Map"

12.5.1 ICF Provisioning Manager

ICF Provisioning Manager unites the access to provisioning methods of connectors into one Java Task that serves all connectors.

The public methods are divided into four groups:

- Section 12.5.1.1, "APIs for Provisioning"
- Section 12.5.1.2, "Account Related Operations"
- Section 12.5.1.3, "Multivalued Operations"
- Section 12.5.1.4, "Other operations"

12.5.1.1 APIs for Provisioning

The following are the single-valued CRUD object types.

createObject

Creates object of a specified type on the target resource, the values are taken from the current Form.

Signature: public String createObject(String objectType)l

deleteObject

Deletes object of a specified type on the target resource.

Signature: public String deleteObject(String objectType)

updateAttributeValue

Updates object on target resource, only the attribute with the provided label is updated.

Signature: public String updateAttributeValue(String objectType, String attrFieldName)

updatePassword

Use this method in Adapter ONLY if you need to provide old password value, currently there is no way to get the old value using the formAPI. If you don't need old password value to change the password, use #updateAttributeValue(String, String) method instead.

Signature: public String updatePassword(String objectType, String pswdFieldLabel, String oldPassword)

12.5.1.2 Account Related Operations

The following are the account related provisioning operations.

enableUser

Deprecated, use enableObject() instead.

Signature: public String enableUser()

disableUser

Deprecated, use disableObject() instead.

Signature: public String disableUser()

enableObject

Example usage for User: enableObject("User").

Signature: public String enableObject(String objectType)

disableObject

Signature: public String disableObject(String objectType)

12.5.1.3 Multivalued Operations

The following are the multivalued operations used in provisioning.

updateAttributeValues

Use this method if there is a group update of fields. This will be useful when a set of attributes have to updated together.

Signature: public String updateAttributeValues(String objectType, String[] labels) public String updateAttributeValues(String objectType, Map<String, String> fields) public String updateAttributeValues(String objectType, Map<String, String> fields, Map<String, String> oldFields)}}}

addChildTableValue

Updates the target by adding the newly added row in child table.

Signature: public String addChildTableValue(String objectType, String childTableName, long childPrimaryKey)

removeChildTableValue

Updates the target by removing the row which was just deleted from child table.

Signature: public String removeChildTableValue(String objectType, String childTableName, Integer taskInstanceKey)

updateChildTableValue

Updates the target by removing the deleted row and adding the newly created row.

Signature: public String updateChildTableValue(String objectType, String childTableName, Integer taskInstanceKey, long childPrimaryKey)

updateChildTableValue

Updates values provided in child table on target resource.

Signature: public String updateChildTableValues(String objectType, String childTableName)

12.5.1.4 Other operations

The following is the other operation used in provisioning.

setEffectiveITResourceName

If the connector needs to use different IT Resource for provisioning operations, it can be set by this method.

Signature: public void setEffectiveITResourceName(String itResourceName)

12.5.2 Provisioning Lookup

Lookup.CONNECTOR_NAME.UM.ProvAttrMap contains basic attribute mapping for two classes of attributes:

- Single valued attributes: simple string key + value pairs.
- Multivalued attributes (Child tables in Oracle Identity Manager): These are further divided by the depth of hierarchy:
  - Simple multivalued attributes: represent records of data stored in child table, see second row in Table 12–7.
  - Complex multivalued attributes: multiple levels of embedded objects, see last row in Table 12–7.

Table 12–7 Provisioning Lookup Attributes

Key	Value	Description
Form Field Label	ConnectorAttributeName	This is a basic mapping type, simple Form Label Name to single value Connector Attribute Name

Table 12–7 (Cont.) Provisioning Lookup Attributes

Key	Value	Description
Child Form Name>~<Child Form Field Label	ConnectorAttributeName	This maps child form field to multivalued ConnectorAttributeName
Child Form Name>~<Child Form Field Label	ConnectorAttributeName>~<EmbeddedObjectClass>~<EmbeddedAttributeName	This maps child form field to EmbeddedAttribute of the embedded object, which object class is EmbeddedObjectClass and it is included in ConnectorAttributeName

12.5.3 Non-User Object Types

There are number of other entities that can be provisioned for example LDAP Organizational Unit (also called OU), or LDAP Group or Group. In this case you need to fill in the OBJECT_TYPE in the following examples:

Main Configuration Lookup Lookup.CONNECTOR_NAME.Configuration

Table 12–8 Configuration Lookup for Connector

Key	Value	Description
objectType Configuration Lookup	Lookup.<ConnectorName>.<objectType>.ProvAttrMap	
Group Configuration Lookup	Lookup.LDAP.Group.ProvAttrMap	Example for LDAP Group

Provisioning Lookup Lookup.CONNECTOR_NAME.OBJECT_TYPE.ProvAttrMap

Key	Value	Description
FORM_FIELD_LABEL_ON_THE_PROCESS_FORM	Target system attribute name	Attribute mapping between Oracle Identity Manager and the connector.

12.5.4 Optional Lookups for Provisioning

Key	Value	Description
FORM_FIELD_NAME [Create, Update, Delete]	ConnectorOperationOptionName	This field is used for generic definition. For example, where the field is mapped to operation option for CreateOp that is sent to connector named as myOperationOption.
myField[Create]	myOperationOption	

12.5.4.1 Provisioning Validation Lookup

Validation code is in an external OIM Java Task, it is a Java class uploaded to OIM repository. Validation java class template:

```
public class MyValidator implements
oracle.iam.connectors.common.validate.Validator {
  public boolean validate(java.util.HashMap hmUserDetails, java.util.HashMap
hmEntitlementDetails,String sField) throws
oracle.iam.connectors.common.ConnectorException {
    boolean isValid = false;
    // validation code goes HERE
    return isValid;
  }
}
```

The name of lookup storing the Recon Validation Lookup is defined in Main Configuration Lookup (Lookup.CONNECTOR_NAME.Configuration).

Key	Value	Description
Form Field Label	validatorClassName com.validationexample.MyValidator	Java class which performs validation for this recon field.

12.5.5 Optional Flags in Lookups for Provisioning Attribute Map

ICF-OIM Integration offers some advanced flags that modify the way provisioning is done. The following is the example for formats of flags in look up key:

```
<key value>[<flag>]
<key value>[<flag1, flag2, flag3>]
```

Let us assume we have a Group OIM attribute that is mapped to UnixGroup Connector attribute. This OIM attribute is populated by a UI lookup. The correct row in Provisioning lookup will be:

```
nullLookup key: Group[LOOKUP]
Lookup value: UnixGroup }}}
```

The following is the list of flags and their effects.

Provisioning Lookup Flag: TRUSTED

For some attributes (for example trusted reconciliation of __ENABLE__ attribute), you need to pass on different values for trusted and target mode of operation. For most of the connectors which support status Reconciliation use code key: Status[Trusted], and decode value: __ENABLE__.

Provisioning Lookup Flag: IGNORE

An attribute marked as IGNORE, will be ignored during provisioning.

Provisioning Lookup Flag: WRITEBACK

If a field has WRITEBACK property, then update of that form field is:

1. update the value on the target system
2. query the value back from the target system (in order to get a normalized value)
3. update this normalized value on the user form.

Provisioning Lookup Flag: DATE

Use this flag to mark date fields. Oracle Identity Manager will apply the localized date format to these fields.

Provisioning Lookup Flag: PROVIDEONPSWDCHANGE

Use this flag to mark additional attributes that are needed for password change operation. By default only __PASSWORD__ attribute is sent, if no flag is applied.

12.5.6 Compound attributes in Provisioning Attribute Map

ICF Common enables to use Groovy expressions on the right hand side, so that provisioned attribute can be computed based on multiple fields. For example, in Active Directory Connector, decode value for the name field is: .

```
__NAME__=CN=${Common_Name},${Organization_Name}
```

12.6 Concepts of Reconciliation in ICF Common

ICF Common leverages the definition and types of reconciliation defined by Oracle Identity Manager server. IT Resource Name / Resource Object Name and Object Type are mandatory attributes reconciliation using ICF Common. Any target system attribute can be used as Latest Token Attribute.

This section contains the following topics:

- Section 12.6.1, "Types of Reconciliation"
- Section 12.6.2, "List of Reconciliation Artifacts in Oracle Identity Manager"

12.6.1 Types of Reconciliation

Reconciliation involves pulling identities from resource (also referred as target) to destination (Oracle Identity Manager). Reconciliation can be classified based on following criteria:

- Destination type: trusted, target recon.
- Scope: full, incremental recon.

Table 12–9 illustrates the common reconciliation parameters.

Table 12–9 ICF Common Reconciliation Parameters

Parameter	Field Setting	Description
Filter	Optional	Filter to limit the number of reconciled accounts, or to select specific set of users.
IT Resource Name	Mandatory	Name of IT Resource instance to reconciliation.
Object Type	Constant	User object class
Resource Object Name	Constant	Determines what OIM Resource Object to use for reconciliation.

12.6.1.1 Target and Trusted Reconciliation

Scheduled task name include keywords such as trusted, target, to determine the type of destination. By choosing the scheduled task, it is determined whether trusted or target reconciliation is launched.

12.6.1.2 Full, Incremental Reconciliation

Full reconciliation involves reconciling all existing user records from the target system into Oracle Identity Manager. During Full Reconciliation, scheduled task is launched for the first time, it is run in full reconciliation mode and from next runs happen in incremental mode. It is possible to switch manually between full/incremental reconciliation modes by emptying the Latest Token field on the scheduled task.

If no value is supplied in Incremental Recon Date Attribute and Incremental Recon Attribute, reconciliation is considered as Target Recon.

The following scheduled tasks offer optional incremental reconciliation:

- Connector Target User Reconciliation
- Connector Trusted User Reconciliation

12.6.1.3 Advanced Incremental Reconciliation

The format of Latest Token is altered by setting the Recon Date Format scheduled task parameter. The formatting string needs to follow standard pattern used in Java. For more information about formatting string used in Java, see Java Doc on Oracle Technology Network.

By default the `Latest Token is` long value that holds Unix/POSIX time.

12.6.1.4 Delete Reconciliation

Some connectors supports both trusted and target reconciliation of deleted accounts. Target reconciliation evaluate which OIM users have lost their account on the resource, and unassign this resource in Oracle Identity Manager. Trusted Delete Reconciliation goes further, and deletes the OIM User.

12.6.1.5 Group Lookup Reconciliation

Some connectors may support reconciliation of Groups, or other object classes to Lookups.

Before the first use of provisioning with the connector, it is advised to launch Lookup reconciliation. This reconciliation populate the `Lookup.CONNECTOR_NAME.ObjectType` table with groups available on an IT Resource that is being reconciled. The reconciliation is performed by the Connector Lookup Reconciliation scheduled task.

You need to set the IT resource parameter name, the rest of the parameters are constant as shown in Table 12–9.

Table 12–10 illustrates the common reconciliation parameters.

Table 12–10 Common Group Lookup Parameters

Code Key	Decode Key	Object Type
Form field name	Connector attribute	Group, or other

For example, the list of names returned by the connector is used to populate the lookup for provisioning. When a new user is provisioned, the group field can display the list of available groups.

12.6.2 List of Reconciliation Artifacts in Oracle Identity Manager

In Oracle Identity Manager, there are two methods of control over reconciliation:

- Lookups for Reconciliation: they define mapping, transformation of the attributes.

- Scheduled tasks - they define the way reconciliation is executed on connector side, or determine account/lookup mode of reconciliation.

12.6.2.1 Lookups for Reconciliation

The following are the lookups for reconciliation:

Reconciliation Attribute Map Lookup

The reconciliation attribute map contains the following pairs:

- Code key: Resource Object reconciliation field name
- Decode: Target system attribute name

Table 12–11 illustrates this mapping (Lookup.CONNECTOR_ NAME.UM.ReconAttrMap) used by Scheduled tasks that perform reconciliation.

> **Note:** Resource Objects are different for Trusted and Target mode of reconciliation.

Table 12–11 Attribute Mapping for Lookup.CONNECTOR_NAME.UM.ReconAttrMap

Key	Value	Description
Recon Field Name	ConnectorAttributeName	This is a basic mapping type, single value Connector Attribute Name to simple Recon Field.
Recon Field Name~Child Recon Field Name	ConnectorAttributeName	This maps multivalued ConnectorAttributeName to child recon field.
Recon Field Name~Child Recon Field Name	ConnectorAttributeName~EmbeddedObjectClass~EmbeddedAttribute	This maps embedded attribute to child recon field

Example showing Design Console updates to setup reconciliation with child table

The following is the example showing Design Console updates to setup reconciliation with child table:

- Child table name: UD_FF_CHILD
- Column name: UD_FF_CHILD_ROLE
- Field label: Role

  To set up reconciliation with the above child table:

  1. Open Resource Object under Resource Management.
  2. Create a new Reconciliation Data field under Object Reconciliation tab.

> **Note:** While creating a new Reconciliation Data field, you must ensure that the field name be Roles and Field Type be Multi-Valued Attribute. This represents the child table as a whole UD_FF_CHILD.

3. Right click on the newly created Reconciliation Data Field and define a new property field as **Role**. This represents the actual column of the child table UD_FF_CHILD_ROLE.
4. Open **Reconciliation Field Mapping** under Process Definition.
5. Click on **Add Table Map**.
6. Select Field Name as **Roles**.
7. Select Table Name as **UD_FF_CHILD**.
8. Right click on the newly created field name Roles, click on **Define proper field name**.
9. Select **Role** for field name.
10. Select Process data field as **UD_FF_CHILD_ROLE**.
11. Update Lookup.CONNECTOR_NAME.UM.ReconAttrMap to include new lookup field with code key = Roles~Role and decode = Role (this should be connector side attribute name).
12. Go back to Resource Object and create reconciliation profile.
13. Clear cache.

12.7 Predefined Scheduled Tasks

ICF-OIM integration provides the following list of predefined scheduled tasks that a connector supports:

- Section 12.7.1, "LookupReconTask"
- Section 12.7.2, "SearchReconTask"
- Section 12.7.3, "SearchReconDeleteTask"
- Section 12.7.4, "SyncReconTask"

12.7.1 LookupReconTask

This scheduled task is based on ICF SearchOp based reconciliation. Oracle Identity Manager form field of type lookup stores a set of predefined values. These values originate from the connector's search query. The Code Key Attribute is the form field's name, and the Decode Attribute is the name of attribute on the target system (also called Connector).

Internally, this task invokes a search operation on the connector for the given Object Type that is translated to ICF Object Class eventually.

Table 12–12 Identity Connector Lookup Reconciliation Attributes

Key	Value
IT Resource Name	Specifies the name of the IT resource for target system installation.
Object Type	User
Lookup Name	This attribute holds the name of the lookup definition that maps each lookup definition with the data source from which values must be fetched.
Decode Attribute	Specifies the Decode Key column of the lookup definition.

Table 12–12 (Cont.) Identity Connector Lookup Reconciliation Attributes

Key	Value
Code Key Attribute	Specifies the Code Key column of the lookup definition.
Filter	Allows to create sophisticated filtration expressions in order to speed up/refine scheduled task execution.

12.7.2 SearchReconTask

The following is the ICF SearchOp based reconciliation.

Table 12–13 Identity Connector Target Search Reconciliation Attributes

Key	Value
IT Resource Name	Specifies the name of the IT resource for target system installation.
Resource Object Name	Specifies the name of the Resource Object used for reconciliation.
Object Type	User
Filter	Allows to create sophisticated filtration expressions in order to speed up/refine scheduled task execution.
Latest Token	Used in Filter as one of the criteria in incremental reconciliation. Any target system attribute can be used as Latest Token Attribute. This value is calculated as follows: If a reconciliation has fetched 100 records and Timestamp is chosen as a Incremental Recon Attribute, then Latest Token = Max Timestamp of all 100 records. It is not the Schedule task execution end timestamp.
Incremental Recon Date Attribute (optional, type Date)	Attribute used to update Latest Token. **Note**: If no value is supplied in Incremental Recon Date Attribute, then reconciliation is considered as Target Reconciliation.
Incremental Recon Attribute (optional, type long)	Attribute used to update Latest Token. **Note**: If no value is supplied in Incremental Recon Attribute , then reconciliation is considered as Target Reconciliation.

12.7.3 SearchReconDeleteTask

This scheduled task is used for ICF SearchOp based reconciliation.

Table 12–14 Identity Connector Target Search Delete Reconciliation Attributes

Key	Value
IT Resource Name	Specifies the name of the IT resource for target system installation.
Resource Object Name	Specifies the name of the Resource Object used for reconciliation
Object Type	User
Filter	Allows to create sophisticated filtration expressions in order to speed up/refine scheduled task execution.

12.7.4 SyncReconTask

This scheduled task is used for `ICF SyncOp` based reconciliation. The Sync Token field persists the token of last synchronization.

Table 12–15 Identity Connector Target Sync Reconciliation Attributes

Key	Value
IT Resource Name	Specifies the name of the IT resource for target system installation.
Resource Object Name	Specifies the name of the Resource Object used for reconciliation
Object Type	User
Filter	Allows to create sophisticated filtration expressions in order to speed up/refine scheduled task execution.
Sync Token	Token of last synchronization.

12.8 ICF Filter Syntax

GroovyFilterBuilder allows to create sophisticated filtration expressions in order to speed up/refine scheduled task execution.

WARNING: The GroovyFilterBuilder uses the connector attribute name for filtration. See Connector documentation for the attribute name.

Examples

The following example could limit the number of reconciled accounts to only those, where account name starts with letter "a", this filter is denoted by the following expression:

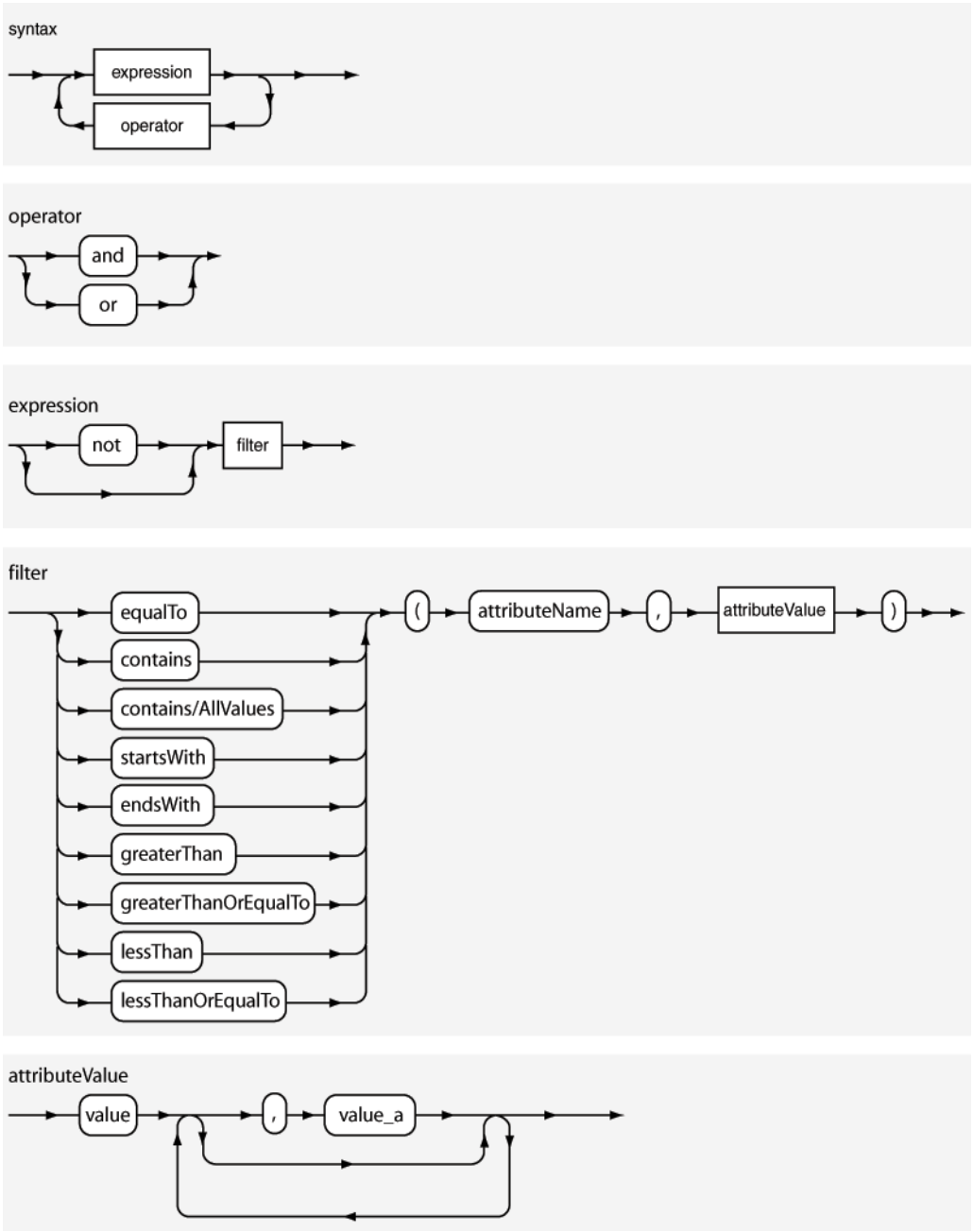
```
startsWith('__NAME__', 'a')
```

Some more advanced search could require to filter only those account names, which end with "z" letter, therefore the filter is:

```
startsWith('__NAME__', 'a') & endsWith('__NAME__', 'z')
```

Figure 12–3 shows the graphical scheme of Filter Syntax.

Figure 12–3 Graphical Representation of Filter Syntax



It is also possible to use a shortcut for and/or operators.

For example, <filter1> & <filter2> instead of and (<filter1>, <filter2>) , analogically replace or with |.

Definition in EBNF format:

The following is the Extended Backus–Naur Form (EBNF) description of the expression language used for Search Filters in reconciliation.

```
syntax = expression ( operator expression )*
operator = 'and' | 'or'
expression = ( 'not' )? filter
filter = ('equalTo' | 'contains' | 'containsAllValues' | 'startsWith' | 'endsWith'
```

```
| 'greaterThan' | 'greaterThanOrEqualTo' | 'lessThan' | 'lessThanOrEqualTo' )  '('
'attributeName' ',' attributeValue ')'
attributeValue = singleValue  |  multipleValues
singleValue = 'value'
multipleValues = '[' 'value_1' (',' 'value_n')* ']'
```

Table 12–16 lists the filter syntax that you can use and the corresponding description and sample values.

Note: Filters with wildcard characters are not supported.

Table 12–16 Keywords and Syntax for the Filter Attribute

Filter Syntax	Description
String Filters	
startsWith('*ATTRIBUTE_NAME*','*PREFIX*')	Records whose attribute value starts with the specified prefix are reconciled. **Example:** `startsWith('userPrincipalName','John')` In this example, all records whose userPrincipalName begins with 'John' are reconciled.
endsWith('*ATTRIBUTE_NAME*','*SUFFIX*')	Records whose attribute value ends with the specified suffix are reconciled. **Example:** `endsWith('sn','Doe')` In this example, all records whose last name ends with 'Doe' are reconciled.
contains('*ATTRIBUTE_NAME*','*STRING*')	Records where the specified string is contained in the attribute's value are reconciled. **Example:** `contains('displayName','Smith')` In this example, all records whose display name contains 'Smith' are reconciled.
containsAllValues('*ATTRIBUTE_NAME*',['*STRING1*','*STRING2*', . . . ,'*STRINGn*'])	Records that contain all the specified strings for a given attribute are reconciled. **Example:** `containsAllValues('objectClass',['person','top'])` In this example, all records whose objectClass contains both "top" and "person" are reconciled.
Equality and Inequality Filters	
equalTo('*ATTRIBUTE_NAME*','*VALUE*')	Records whose attribute value is equal to the value specified in the syntax are reconciled. **Example:** `equalTo('sAMAccountName','Sales Organization')` In this example, all records whose sAMAccountName is Sales Organization are reconciled.

Table 12–16 (Cont.) Keywords and Syntax for the Filter Attribute

Filter Syntax	Description
greaterThan('ATTRIBUTE_NAME','VALUE')	Records whose attribute value (string or numeric) is greater than (in lexicographical or numerical order) the value specified in the syntax are reconciled. **Example 1:** `greaterThan('cn','bob')` In this example, all records whose common name is present after the common name 'bob' in the lexicographical order (or alphabetical order) are reconciled. **Example 2:** `greaterThan('employeeNumber','1000')` In this example, all records whose employee number is greater than 1000 are reconciled.
greaterThanOrEqualTo('*ATTRIBUTE_NAME*','*VALUE*')	Records whose attribute value (string or number) is lexographically or numerically greater than or equal to the value specified in the syntax are reconciled. **Example 1:** `greaterThanOrEqualTo('sAMAccountName','S')` In this example, all records whose sAMAccountName is equal to 'S' or greater than 'S' in lexicographical order are reconciled. **Example 2:** `greaterThanOrEqualTo('employeeNumber','1000')` In this example, all records whose employee number is greater than or equal to 1000 are reconciled.
lessThan('*ATTRIBUTE_NAME*','*VALUE*')	Records whose attribute value (string or numeric) is less than (in lexicographical or numerical order) the value specified in the syntax are reconciled. **Example 1:** `lessThan('sn','Smith')` In this example, all records whose last name is present after the last name 'Smith' in the lexicographical order (or alphabetical order) are reconciled. **Example 2:** `lessThan('employeeNumber','1000')` In this example, all records whose employee number is less than 1000 are reconciled.
lessThanOrEqualTo('*ATTRIBUTE_NAME*','*VALUE*')	Records whose attribute value (string or numeric) is lexographically or numerically less than or equal to the value specified in the syntax are reconciled. **Example 1:** `lessThanOrEqualTo('sAMAccountName','A')` In this example, all records whose sAMAccountName is equal to 'A' or less than 'A' in lexicographical order are reconciled. **Example 2:** `lessThanOrEqualTo('employeeNumber','1000')` In this example, all records whose employee numer is less than or equal to 1000 are reconciled.
Complex Filters	

Table 12–16 (Cont.) Keywords and Syntax for the Filter Attribute

Filter Syntax	Description
<*FILTER1*> & <*FILTER2*>	Records that satisfy conditions in both filter1 and filter2 are reconciled. In this syntax, the logical operator & (ampersand symbol) is used to combine both filters. **Example:** `startsWith('cn', 'John') & endsWith('sn', 'Doe')` In this example, all records whose common name starts with John and last name ends with Doe are reconciled.
<*FILTER1*> | <*FILTER2*>	Records that satisfy either the condition in filter1 or filter2 are reconciled. In this syntax, the logical operator | (vertical bar) is used to combine both filters. **Example:** `contains('sAMAccountName', 'Andy') | contains('sn', 'Brown')` In this example, all records that contain 'Andy' in the sAMAccount Name attribute or records that contain 'Brown' in the last name are reconciled.
not(<*FILTER*>)	Records that do not satisfy the given filter condition are reconciled. **Example:** `not(contains('cn', 'Mark'))` In this example, all records that does not contain the common name 'Mark' are reconciled.

13

Using Java APIs for ICF Integration

To build a custom connector, you must first implement the Identity Connector (ICF-based connector) by implementing the ICF SPI. After this, you need to create OIM artifacts to integrate the Identity Connector to Oracle Identity Manager, which can reuse ICProvisioningManager (part of ICF integration) in their Adapter Tasks to invoke provisioning operations on Identity Connector, and also can reuse Reconciliation Tasks that are implemented in ICF integration. Therefore, by using ICF integration, you need not write any integration code in java, ICF integration uses ICF APIs to access the Identity Connectors.

For information about Java APIs related to ICF integration, see *Oracle Fusion Middleware Java API Reference for Identity Connector Framework.*

For information about Java APIs related to Oracle Identity Manager, see *Oracle Fusion Middleware Java API Reference for Oracle Identity Manager*.

14

Configuring ICF Connectors

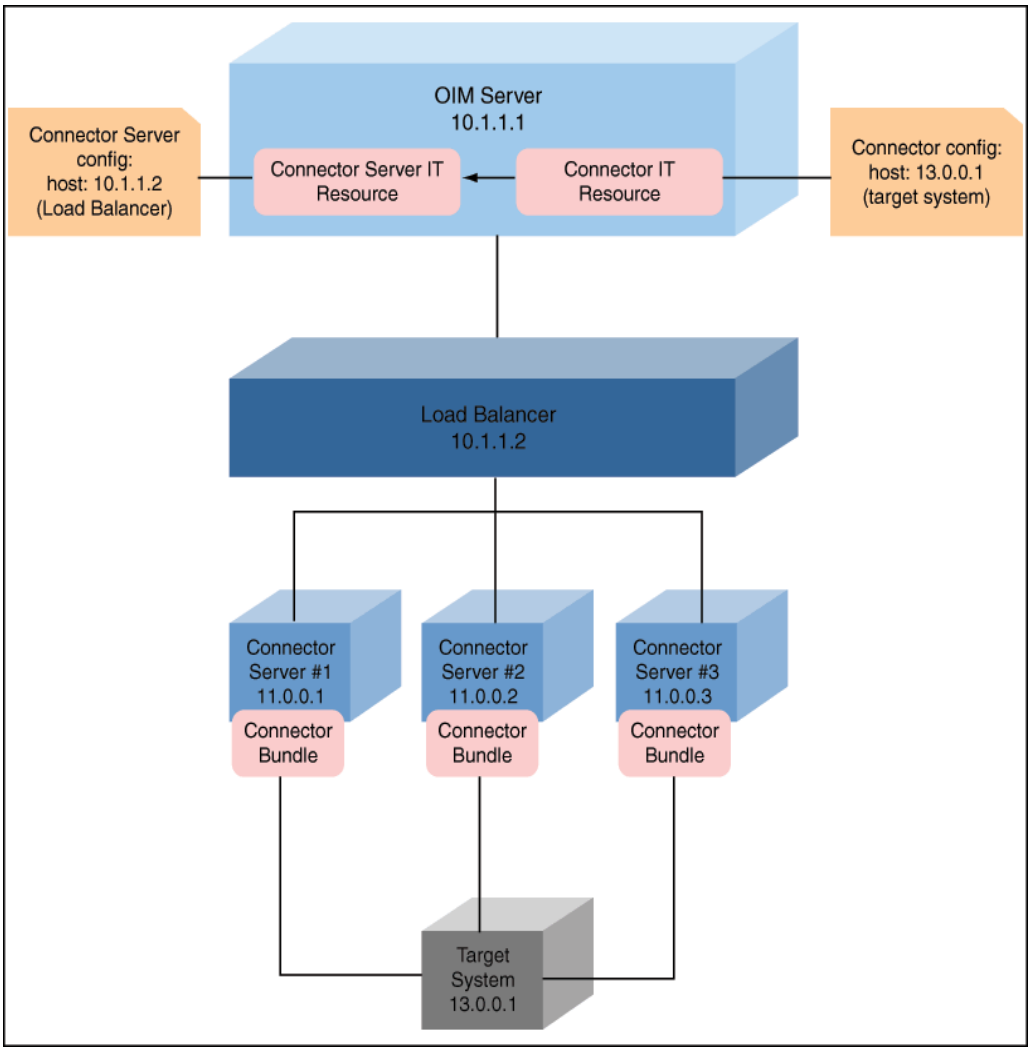
This chapter provides the information about the common customization procedures that needs to be performed for all ICF connectors.

The following are the topics discussed under this chapter:

14.1 Configuring Connector Load Balancer

A connector server is an application that enables remote execution of an Identity Connector. If there are multiple connector servers, then you must ensure the high availability of the connector server for the remote execution of the Identity connector and failover management. Therefore, you must configure a load balancer for a connector server. Figure 14–1 depicts the typical configuration for a cluster of connector servers. The flow in the figure is based on the assumption that the required connector bundle is deployed across all the connector servers.

Figure 14–1 Connector Server Load Balancer



To configure the load balancer for a connector server:

1. Install connector server on nodes including the connector bundle. This involves copying and running the server binaries on all nodes.
2. Setup your load balancer so that every request on port 8759 (default for connector server, which is configurable) is being load balanced across the nodes created in Step 1.
3. Create a connector server IT resource, and point it to your host deployed with load balancer.
4. Configure your connector IT resource with the following details:
   - host: target address
   - connector server name: use the name created in Step 3.

Note: You must make sure to double-check that the incoming port of load balancer is same as the one given in connector server IT resource. In addition, you must check that the ports set up for cluster nodes match the one used for configuring your load balancer.

14.2 Configuring Validation of Data During Reconciliation and Provisioning

The Lookup.CONNECTOR_NAME.ProvValidations and Lookup.CONNECTOR_NAME.UM.ReconValidations lookup definitions hold single-valued data to be validated during provisioning and reconciliation operations, respectively.

For example, you can validate data fetched from the First Name attribute to ensure that it does not contain the number sign (#). In addition, you can validate data entered in the First Name field on the process form so that the number sign (#) is not sent to the target system during provisioning operations.

> **Note:** The Lookup.CONNECTOR_NAME.UM.ProvValidations and Lookup.CONNECTOR_NAME.UM.ReconValidations lookup definitions are optional and do not exist by default.
>
> You must add these lookups as decode values to the Lookup.CONNECTOR_NAME.UM.Configuration lookup definition to enable exclusions during provisioning and reconciliation operations. See the respective connector guide for more information about setting up the lookup definition for user operations.

To configure validation of data:

1. Write code that implements the required validation logic in a Java class with a fully qualified domain name (FQDN), such as `org.identityconnectors.CONNECTOR_NAME.extension.CONNECTOR_NAMEValidator`.

   This validation class must implement the validate method. The following sample validation class checks if the value in the First Name attribute contains the number sign (#):

```
package com.validationexample;

import java.util.HashMap;

public class MyValidator {
    public boolean validate(HashMap hmUserDetails, HashMap
hmEntitlementDetails, String sField) throws ConnectorException {

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

```
                valid = false;
                break;
            }
        }
        return valid;

    }
}
```

2. Log in to the Design Console.
3. Create one of the following new lookup definitions:
   - To configure validation of data for reconciliation:

     `Lookup.CONNECTOR_NAME.UM.ReconValidations`
   - To configure validation of data for provisioning:

     `Lookup.CONNECTOR_NAME.UM.ProvValidations`
4. In the **Code Key** column, enter the resource object field name that you want to validate. For example, `Alias`.
5. In the **Decode** column, enter the class name. For example, `org.identityconnectors.CONNECTOR_NAME.extension.CONNECTOR_NAMEValidator`.
6. Save the changes to the lookup definition.
7. Search for and open the **Lookup.CONNECTOR_NAME.UM.Configuration** lookup definition.
8. In the **Code Key** column, enter one of the following entries:
   - To configure validation of data for reconciliation:

     `Recon Validation Lookup`
   - To configure validation of data for provisioning:

     `Provisioning Validation Lookup`
9. In the **Decode** column, enter one of the following entries:
   - To configure validation of data for reconciliation:

     `Lookup.CONNECTOR_NAME.UM.ReconValidations`
   - To configure validation of data for provisioning:

     `Lookup.CONNECTOR_NAME.UM.ProvValidations`
10. Save the changes to the lookup definition.
11. Create a JAR with the class and upload it to the Oracle Identity Manager database as follows:

    Run the Oracle Identity Manager Upload JARs utility to post the JAR file created in Step 7 to the Oracle Identity Manager database. This utility is copied into the following location when you install Oracle Identity Manager:

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

For Microsoft Windows:

OIM_HOME/server/bin/UploadJars.bat

For UNIX:

OIM_HOME/server/bin/UploadJars.sh

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

See Also: "Migrating JARs and Resource Bundle" on page 37-5 for detailed information about the Upload JARs utility

12. Run the PurgeCache utility to clear content related to request datasets from the server cache.
13. Perform reconciliation or provisioning to verify validation for the field, for example, Alias.

14.3 Configuring Transformation of Data During User Reconciliation

The Lookup.CONNECTOR_NAME.UM.ReconTransformations lookup definition holds single-valued user data to be transformed during reconciliation operations. For example, you can use First Name and Last Name values to create a value for the Full Name field in Oracle Identity Manager.

Note: The Lookup.CONNECTOR_NAME.UM.ReconTransformations lookup definition is optional and does not exist by default.

You must add this lookup as decode value to the Lookup.CONNECTOR_NAME.UM.Configuration lookup definition to enable exclusions during provisioning and reconciliation operations. See the respective connector guide for more information about setting up the lookup definition for user operations.

To configure transformation of single-valued user data fetched during reconciliation:

1. Write code that implements the required transformation logic in a Java class with a fully qualified domain name (FQDN), such as `org.identityconnectors.CONNECTOR_NAME.extension.CONNECTOR_NAMETransformation`.

   This transformation class must implement the transform method. The following sample transformation class creates a value for the Full Name attribute by using values fetched from the First Name and Last Name attributes of the target system:

```
package com.transformationexample;

import java.util.HashMap;


public class MyTransformer {
    public Object transform(HashMap hmUserDetails, HashMap
hmEntitlementDetails, String sField) throws ConnectorException {
        /*
```

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

    }
}
```

2. Log in to the Design Console.
3. Create a new lookup definition, **Lookup.CONNECTOR_NAME.UM.ReconTransformations.**
4. In the **Code Key** column, enter the resource object field name you want to transform. For example, `Alias`.
5. In the **Decode** column, enter the class name. For example, `org.identityconnectors.CONNECTOR_NAME.extension.CONNECTOR_NAMETransformation`.
6. Save the changes to the lookup definition.
7. Search for and open the **Lookup.CONNECTOR_NAME.UM.Configuration** lookup definition.
8. In the **Code Key** column, enter `Recon Transformation Lookup`.
9. In the **Decode** column, enter `Lookup.CONNECTOR_NAME.UM.ReconTransformations`.
10. Save the changes to the lookup definition.
11. Create a JAR with the class and upload it to the Oracle Identity Manager database as follows:

    Run the Oracle Identity Manager Upload JARs utility to post the JAR file created in Step 7 to the Oracle Identity Manager database. This utility is copied into the following location when you install Oracle Identity Manager:

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

    For Microsoft Windows:

    OIM_HOME/server/bin/UploadJars.bat

    For UNIX:

    OIM_HOME/server/bin/UploadJars.sh

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

See Also: "Migrating JARs and Resource Bundle" on page 37-5 for detailed information about this utility.

12. Run the PurgeCache utility to clear content related to request datasets from the server cache.
13. Perform reconciliation to verify transformation of the field, for example, Alias.

14.4 Configuring Resource Exclusion Lists

The Lookup.CONNECTOR_NAME.UM.ProvExclusionList and Lookup.CONNECTOR_NAME.UM.ReconExclusionList lookup definitions hold user IDs of target system accounts for which you do not want to perform provisioning and reconciliation operations, respectively.

Note: The Lookup.CONNECTOR_NAME.UM.ProvExclusionList and Lookup.CONNECTOR_NAME.UM.ReconExclusionList lookup definitions are optional and do not exist by default.

You must add these lookups as decode values to the Lookup.CONNECTOR_NAME.UM.Configuration lookup definition to enable exclusions during provisioning and reconciliation operations. See the respective connector guide for more information about setting up the lookup definition for user operations.

The following is the format of the values stored in these lookups:

Code Key	Decode	Sample Values
User Login Id resource object field name	User ID of a user	Code Key: User Login Id Decode: User001
User Login Id resource object field name with the [PATTERN] suffix	A regular expression supported by the representation in the `java.util.regex.Pattern` class	Code Key: User Login Id[PATTERN] To exclude users matching any of the user ID 's User001, User002, User088, then: Decode: User001 | User002 | User088 To exclude users whose user ID 's start with 00012, then: Decode: 00012* **See Also:** For information about the supported patterns, visit `http://download.oracle.com/javase/6/docs/api/java/util/regex/Pattern.html`

To add entries in the lookup for exclusions during provisioning operations:

1. On the Design Console, expand **Administration** and then double-click **Lookup Definition.**
2. Create a new lookup definition, **Lookup.CONNECTOR_NAME.UM.ProvExclusionList.**

Note: To specify user IDs to be excluded during reconciliation operations, create a new lookup definition called Lookup.CONNECTOR_NAME.UM.ReconExclusionList and add entries to that lookup.

3. Click **Add.**
4. In the Code Key and Decode columns, enter the first user ID to exclude.

Note: The Code Key represents the resource object field name on which the exclusion list is applied during provisioning operations.

5. Repeat Steps 3 and 4 for the remaining user IDs to exclude.

   For example, if you do not want to provision users with user IDs User001, User002, and User088 then you must populate the lookup definition with the following values:

Code Key	Decode
User Login Id	User001
User Login Id	User002
User Login Id	User088

You can also perform pattern matching to exclude user accounts. You can specify regular expressions supported by the representation in the `java.util.regex.Pattern` class.

See Also: For information about the supported patterns, visit `http://download.oracle.com/javase/6/docs/api/java/util/regex/Pattern.html`

For example, if you do not want to provision users matching any of the user IDs User001, User002, and User088, then you must populate the lookup definition with the following values:

Code Key	Decode
User Login Id[PATTERN]	User001 | User002 | User088

If you do not want to provision users whose user IDs start with 00012, then you must populate the lookup definition with the following values:

Code Key	Decode
User Login Id[PATTERN]	00012*

6. Click **Save**.

14.5 Setting SSL for Connector Server and OIM

To set up the SSL communication between Connector Server and Oracle Identity Manager:

1. Generate a new SSL key (or you can reuse your existing key):

   ```
   keytool -genkey -alias keyconnserv -keyalg dsa -keystore <yourKeyStore.jks>
   -storepass <yourPassword> -validity 360
   ```

2. Export the newly generated public key:

   ```
   keytool -export -keystore <yourKeyStore.jks> -storepass <yourPassword> -alias
   keyconnserv -file icfkey-public.cer
   ```

3. Configure your Connector Server for SSL, and start using the new keystore set in Step 1.
4. Import the public key generated in Step 2 (icfkey-public.cer) to OIM keystore.
5. Configure IT Resource such as host, port, and so on. These parameters will be passed on to Connector Server (an extra field of IT Resource).
6. Configure Connector Server, using SSL:
   a. Deploy an SSL certificate to the Connector Server's system.
   b. Configure your Connector Server to provide SSL sockets.
   c. 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.

7. Import the public key generated in Step 2 to OIM keystore.

   If you follow to choose the default Weblogic keystore, perform the following:

```
keytool -import -trustcacerts -alias icfkey -file icfkey-public.cer -keystore
<pathToYouKeystore>
```

For example default Weblogic keystores are: server/lib/DemoTrust.jks and server/lib/DemoIdentity.jks.

14.5.1 Troubleshooting SSL

The following is an example of exception in connector server logs:

java.net.SocketException: Default SSL context init failed: null

This means that the path to keystore is incorrect. To handle this exception, make sure you provide the following full/absolute path:

For UNIX

./connectorserver.sh /run "-J-Djavax.net.ssl.keyStore=/path/to/mykeystore.jks" "-J-Djavax.net.ssl.keyStorePassword=changeit"

For Windows

./connectorserver.sh /run "-J-Djavax.net.ssl.keyStore=C:\path\to\mykeystore.jks" "-J-Djavax.net.ssl.keyStorePassword=changeit"

You must also ensure the following check points:

- Check your configuration folder for the setting of connector server configuration to use SSL
- Restart your WLS after importing public keys from the connector server, if the public key present in OIM keystore

14.6 Adding Target System Attributes

Adding target system attributes includes the following subsections:

- Section 14.6.1, "Adding Target System Attributes for Provisioning"
- Section 14.6.2, "Adding Target System Attributes for Target Reconciliation"
- Section 14.6.3, "Adding Target System Attributes for Trusted Reconciliation"

Note: If you add an attribute with a Date type field, make sure that you add the `[Date]` suffix in the Lookup definition code key.

For example, if you add _LAST_PASSWORD_CHANGE_DATE_, when you make changes in the code key for Lookup.CONNECTOR_NAME.UM.ReconAttrMap or Lookup.CONNECTOR_NAME.UM.ProvAttrMap, specify the attribute as:

```
_LAST_PASSWORD_CHANGE_DATE_[Date]
```

14.6.1 Adding Target System Attributes for Provisioning

By default, the target system attributes are mapped for provisioning between Oracle Identity Manager and the target system. If required, you can map additional attributes for provisioning by performing these steps.

Note: In this section, the term "attribute" refers to the identity data fields that store user data.

To add a target system attribute for provisioning, follow these steps:

1. Add a new form field. To add a new field to the Process form:
   a. Open the Form Designer form. This form is in the Development Tools folder of the Oracle Identity Manager Design Console.
   b. Query for the UD_CONNECTOR_NAMECON form.
   c. Click **Create New Version**. The Create a New Version dialog box is displayed.
   d. In the Label field, enter the name of the version.
   e. Click **Save** and close the dialog box.
   f. From the Current Version box, select the version name that you entered in the Label field in Step 4.
   g. On the Additional Columns tab, click **Add**.
   h. Specify the new field name and other values.
   i. Click **Save**.
   j. Click **Make Version Active** to make the new form field visible to the user.

      Now, if you go to Oracle Identity Manager and try to provision a new user to Connector, you should see the new form field. Next, you must add the new form field to the Provisioning Mapping Lookup.
2. Add the new field to the Provisioning Mapping Lookup. After creating a new form field, you must add that field to the Provisioning Mapping Lookup, as follows:
   a. Expand **Administration** and then double-click **Lookup Definition**.
   b. In the Lookup Definition window, search for CONNECTOR_NAME.

      The Design Console returns Lookup.CONNECTOR_NAME.UM.ProvAttrMap.
   c. Select the Lookup Definition Table tab, and select **Lookup.CONNECTOR_NAME.UM.ProvAttrMap**.

      The Lookup Code Information tab maps the Oracle Identity Manager form field names and the CONNECTOR_NAME Identity Connector attributes. Where the Code Key column contains the Oracle Identity Manager field labels and the Decode column contains the attribute names supported by the CONNECTOR_NAME identity connector.
   d. Add a new record for the new form field. Type the new form field name into the Code Key column and type the CONNECTOR_NAME identity connector attribute name into the Decode column.
   e. Click **Save**.

      Now, when you create a new CONNECTOR_NAME user, the connector will get the new attribute as part of the create operation.

      At this point, the process task only handles creates. Next, you must change the process task to also handle updates. Instructions are described in the next steps.

3. Change the process task to handle updates, as follows:

   a. In the Design Console, expand **Process Management** and then double-click **Process definition**.

   b. Search for and select process CONNECTOR_NAME User.

   c. In the Task column, look for an update task that is similar to the one you want to add and select that entry.

   d. Click **Add**.

   e. In the Creating New Task dialog, select the General tab and enter a Task Name and a Task Description.

      The Task Name is important because it will be the form name field. Be sure to include the event you want the task to handle. For example, if you add the Building field for provisioning, then add the Building Updated task. Now, this update event will be triggered when the Building field is updated.

   f. In the Task Properties section, set the following properties as noted:

      -Conditional: Enabled

      -Required for Completion: Disabled

      -Disable Manual Insert: Disabled

      -Allow Cancellation while Pending: Enabled

      -Allow Multiple Instances: Enabled

      You do not have to change any of the remaining properties.

   g. Save your changes.

   h. To add an Event Handler, select the Integration tab, and then click **Add**.

   i. When the Handler Select dialog box displays, select Adapter as the handler type and then perform the following steps:

      Select adapter **adpCONNECTOR_NAMECONNECTORUPDATEATTRIBUTEVALUE** and click **Save**.

      Map all of the variables that are configured for the event adapter.

      In the Adapter Variables section, double-click a variable name to open the Edit Data Mapping For Variable dialog box. Specify the following values for each variable in turn. Be sure to save your changes after each mapping.

Variable Name	Map To	Qualifier	Literal Value
itResourceFieldName	Literal	String	UD_CONNECTOR_NAMECON_SERVER
processInstanceKey	Process Data	Process Instance	
Adapter return value	Response Code		
objectType	Literal	String	User
attrName	Literal	String	Enter your new label

   j. Save and close the Creating New Task dialog.

k. Check the Task column on the Process Definition tab to verify that the new process task is listed. Also verify that the new form field is available and working in Oracle Identity Manager.

14.6.2 Adding Target System Attributes for Target Reconciliation

By default, the target system attributes are mapped for reconciliation between Oracle Identity Manager and the target system. If required, you can map additional attributes for target reconciliation as described in this section.

Note:

- Perform this procedure only if you want to add new target system attributes for reconciliation.
- In the following steps, a new attribute called BUILDING will be added, its connector attribute name is BUILDING, and the form field name is Building. Names are case-sensitive.

To add a new target system attribute for target reconciliation, follow these steps:

1. In the resource object definition, add a reconciliation field corresponding to the new attribute, as follows:
   a. Open the Resource Objects form. This form is in the Resource Management folder.
   b. Click **Query for Records**.
   c. On the Resource Objects Table tab, double-click the **CONNECTOR_NAME User** resource object to open it for editing.
   d. On the Object Reconciliation tab, click **Add Field** to open the Add Reconciliation Field dialog box.
   e. Specify a value for the field name that is the name of the new Attribute on your Form.

      For example: Building
   f. From the Field Type list, select a data type for the field.

      For example: String
   g. Save the values that you enter, and then close the dialog box.
   h. If required, repeat Steps d through g to map more fields.
   i. Click **Create Reconciliation Profile**. This copies changes made to the resource object into the MDS.
2. If a corresponding field does not exist in the process form, then add a new column in the process form, as follows:
   a. Open the Form Designer form. This form is in the Development tools folder.
   b. Query for the UD_CONNECTOR_NAMECON form.
   c. Click **Create New Version**. The Create a New Version dialog box is displayed.
   d. In the Label field, enter the name of the version.
   e. Click **Save** and close the dialog box.

f. From the Current Version box, select the version name that you entered in the Label field in Step 3.

g. On the Additional Columns tab, click **Add**.

h. In the Name field, enter the name of the data field and then enter the other details of the field.

Note: Repeat Steps g and h if you want to add more attributes.

i. Click Save and then click Make Version Active.

3. Modify the process definition to include the mapping between the newly added attribute and the corresponding reconciliation field:

a. Open the Process Definition form. This form is in the Process Management folder of the Design Console.

b. Click the **Query for Records** icon.

c. On the Process Definition Table tab, double-click the **CONNECTOR_NAME User** process definition.

d. On the Reconciliation Field Mappings tab, click **Add Field Map** to open the Add Reconciliation Field Mapping dialog box.

e. From the Field Name list, select the name of the resource object that you added in Step 2e.

f. Double-click Process Data Field and select the corresponding process form field from the Lookup dialog box. Then, click **OK**.

g. Click **Save** and close the dialog box.

h. If required, repeat Steps c through g to map more fields.

4. Go to the reconciliation lookup, Lookup.CONNECTOR_NAME.UM.ReconAttrMap, and add a new record for the new attribute using the following values:

- Code Key - Name of the reconciliation field
- Decode - Name of the CONNECTOR_NAME attribute

5. In the Design Console, regenerate the reconciliation profile for the Resource Object.

14.6.3 Adding Target System Attributes for Trusted Reconciliation

By default, the attributes for trusted source reconciliation are mapped between Oracle Identity Manager and the target system. If required, you can map additional attributes for trusted reconciliation as described in this section.

Note:

- Perform this procedure only if you want to add new target system attributes for reconciliation.
- In the following steps, a new attribute called BUILDING will be added, its connector attribute name is BUILDING, and the form field name is Building. Names are case-sensitive.

To add a new target system attribute for trusted reconciliation, follow these steps:

1. In the resource object definition, add a reconciliation field corresponding to the new attribute, as follows:
   a. Open the Resource Objects form. This form is in the Resource Management folder.
   b. Click **Query for Records**.
   c. On the Resource Objects Table tab, double-click the **CONNECTOR_NAME Trusted User** resource object to open it for editing.
   d. On the Object Reconciliation tab, click **Add Field** to open the Add Reconciliation Field dialog box.
   e. Specify a value for the field name that is the name of the new Attribute on your Form.

      For example: Building
   f. From the Field Type list, select a data type for the field.

      For example: String
   g. Save the values that you enter, and then close the dialog box.
   h. If required, repeat Steps d through g to map more fields.
   i. Click **Create Reconciliation Profile**. This copies changes made to the resource object into the MDS.
2. If a corresponding field does not exist in the process form, then add a new column in the process form, as follows:
   a. Open the Form Designer form. This form is in the Development tools folder.
   b. Query for the UD_CONNECTOR_NAMECON form.
   c. Click **Create New Version**. The Create a New Version dialog box is displayed.
   d. In the Label field, enter the name of the version.
   e. Click **Save** and close the dialog box.
   f. From the Current Version box, select the version name that you entered in the Label field in Step 3.
   g. On the Additional Columns tab, click **Add**.
   h. In the Name field, enter the name of the data field and then enter the other details of the field.

      Note: Repeat Steps g and h if you want to add more attributes.
   i. Click **Save** and then click **Make Version Active**.
3. Modify the process definition to include the mapping between the newly added attribute and the corresponding reconciliation field:
   a. Open the Process Definition form. This form is in the Process Management folder of the Design Console.
   b. Click the **Query for Records** icon.
   c. On the Process Definition Table tab, double-click the **CONNECTOR_NAME Trusted User** process definition.
   d. On the Reconciliation Field Mappings tab, click **Add Field Map** to open the Add Reconciliation Field Mapping dialog box.

e. From the Field Name list, select the name of the resource object that you added in Step 2e.

f. Double-click Process Data Field and select the corresponding process form field from the Lookup dialog box. Then, click **OK**.

g. Click **Save** and close the dialog box.

h. If required, repeat Steps c through g to map more fields.

4. Go to the reconciliation lookup, Lookup.CONNECTOR_NAME.UM.ReconAttrMap.Trusted, and add a new record for the new attribute using the following values:

   - Code Key - Name of the reconciliation field
   - Decode - Name of the CONNECTOR_NAME attribute

15

Understanding ICF Best Practices and FAQs

This chapter enlists the best practices and Frequently Asked Questions (FAQ) on ICF. The list is discussed in the following sections:

- Section 15.1, "Best Practices for ICF"
- Section 15.2, "FAQs on ICF"

15.1 Best Practices for ICF

The following are the best practices that you need to follow while using ICF:

- Use common Scheduled tasks, and ICProvisioningManager.
- Keep IT Resource parameters count to minimum, IT Resource should contain connectivity related parameters only, the rest needs to be in the Main Connector Configuration Lookup.
- Logging in ICF Connectors:

  ICF Integration for Oracle Identity Manager logs all the input/output parameters of all calls to ICF Connector interfaces. You must ensure that the following points are taken care while logging:

  - If required, you can enhance the logging with detailed logging messages.
  - You must not log messages that involves password information or sensitive data.
  - In case you encounter ConnectorException error, then you must wrap the target specific exception, and provide any additional details.
  - Turn on Logging for ICF Common by switching on logging for `oracle.iam.connectors.icfcommon`.
- Connector Load Balancer
  - In order to use SSL-encrypted communication between Oracle Identity Manager and connector servers, you need to copy the SSL keystore on all connector server nodes, and maintain its consistency if SSL key changes.
  - Connector server uses a proprietary (non-HTTP) protocol, and SSL encryption.
  - All connector server nodes under the load balancer should contain the same set of bundles.

15.2 FAQs on ICF

The following are the FAQs on ICF:

- Why lookup reconciliation data contains tilda (~)?

  Tilda (~) notation in lookup reconciliation is to separate Lookups for different IT Resources. In the following example, Key will be a programmatic key, whereas Value will be a user-friendly display name:

  Lookup Key: <IT Resource Key>~<Lookup key>

  Lookup Value: <IT Resource Name>~<Lookup value>

- What is bulk attribute update and how to set it up?

  Bulk attribute update in OIM means that all the changed attributes will be sent to the connector in one method call, instead of updating each attribute individually (default option).

  In order to enable your connector for bulk attribute update, make sure:

  – all your attributes have their respective process tasks for individual update, typically named as, ATTRIBUTE_NAME updated.

  – you have an extra process task named, UD_FORM_NAME updated. This task will be used for bulk update.

- Search-based versus sync-based reconciliation: when to use what?

  It is based on the capabilities of connector/target resource. Most connectors support search, some of them (LDAP) support sync operation too. Where available, sync-based reconciliation is preferred due to higher efficiency.

  Sync-based reconciliation is more efficient than search-based reconciliation because, it can process both additions/removals in one run. With search-based reconciliation, you need to run search reconciliation first and then run search delete reconciliation, which is double the effort.

- How to configure Connector Pooling?

  See Release 11.1.1.5.0 version of the Connector documentation for information about Connector Pooling and its configuration.

- How to use Groovy to extend connector functionality?

  In order to have an extendable connector, you need to implement `ScriptOnConnector` or `ScriptOnResource` ICF SPIs. Connectors might support various scripting languages, based on target resource capabilities. By default, ICF supports groovy scripts with ScriptOnConnector for all java based connectors. You must always refer the connector documentation to understand the scripting languages for a given connector. See Chapter 14, "Configuring ICF Connectors" for more information about how to customize the connector.

- What are the basic requirements (such as memory, disk space, CPU, and so on) for Connector Server?

  The connector server can run in any Java 6 environment and above. The requirements are same as of those of Java and Oracle Identity Manager.

  See Release 11.1.1.5.0 version of the Connector documentation for the supported versions of JDK and Oracle Identity Manager.

- Does one connector server version support all ICF Connector versions?

Connector Server version equals ICF version. ICF is backward compatible with previously released connector versions.

- How to troubleshoot connector server related issues?

  Set up log level to `FINEST` in logging configuration file of the Connector Server. If the default port 8759 is taken, than set a different port number in the Connector Server configuration.

- When to deploy connector on Connector Server and when to deploy connector locally into Oracle Identity Manager?

  Only .NET connectors require Connectors Server, others can be deployed directly into Oracle Identity Manager.

16

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:

16.1 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 16.2, "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 18, "Creating Custom Providers for Generic Technology Connectors" to create the custom providers that call the Java APIs exposed by the target application.

16.2 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:

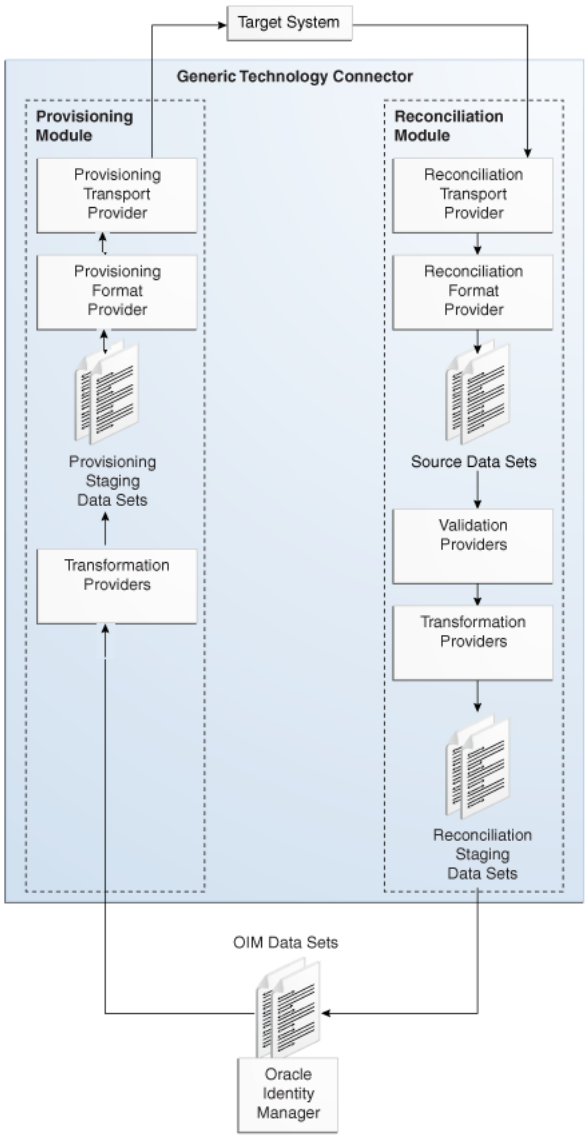
- Establishes a data flow path between fields of two data sets for use either in provisioning or reconciliation.

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 16–1 shows the functional architecture of a generic technology connector.

Figure 16–1 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

16.2.1 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.

16.2.2 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.

16.2.3 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.

16.3 Features of Generic Technology Connectors

The following sections discuss the features of generic technology connectors:

- Features Specific to the Reconciliation Module
- Other Features

16.3.1 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

16.3.1.1 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 5, "Developing Provisioning Processes".

> **Note:** In trusted source reconciliation, the reconciliation of multivalued (child) data is not supported.

16.3.1.2 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 5.3.2.2, "Reconciliation Field Mappings Tab".

16.3.1.3 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.

16.3.1.4 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.

16.3.1.5 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.

16.3.1.6 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.

16.3.2 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

16.3.2.1 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.

16.3.2.2 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.

16.3.2.3 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.

16.3.2.4 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

16.3.2.5 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.

16.4 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.

16.4.1 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.

- Application instance

The combination of IT resource (target connectivity and connector configuration) and resource object (provisioning mechanism). This is a provisionable entity.

- 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 19.2.6, "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 19.2.5, "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

16.4.2 Only Reconciliation Is Selected

See "Both Reconciliation and Provisioning Are Selected" on page 16-9 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.

16.4.3 Only Provisioning Is Selected

See "Both Reconciliation and Provisioning Are Selected" on page 16-9 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

16.5 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 17, "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 18, "Creating Custom Providers for Generic Technology Connectors" explains the role of providers during provisioning and reconciliation, and describes how to create custom providers.
- Chapter 19, "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 20, "Troubleshooting Generic Technology Connectors" describes general and configuration issues related to Generic Technology Connectors and how to troubleshoot the issues.

17

Predefined Providers for Generic Technology Connectors

The following predefined providers are shipped 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 Oracle Identity System Administration.

17.1 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 16-8 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 19-13, 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 19-28, 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 19-6.

  > **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.

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
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: ■ 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.

Storage Entity	Access Permission	Reason for Access Permission Requirement
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.

17.2 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.

17.3 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:

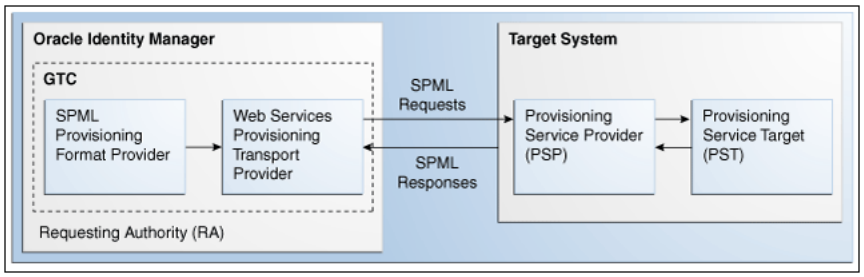
OIM_HOME/GTC/Samples/spml

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 17–1 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 17–1 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 19-13, 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

17.3.1 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).

17.3.2 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/wsdl20/`

- **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>
```

17.3.3 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

17.3.4 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`

17.4 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 `wsdlsoap: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.

17.4.1 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

17.5 Transformation Providers

Note: Use the information provided in this section while performing the instructions given in Section 19.2.4.3, "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 available in Oracle Identity Manager:

- Concatenation Transformation Provider
- Translation Transformation Provider

17.5.1 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 19.2.4.3.1, "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**.

17.5.2 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 7.2.1, "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

Code Key	Decode
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 19.2.4.3.1, "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.

17.5.2.1 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 5.3.2.2.1, "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 7.2, "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)
Target system status value for a user account that is in the Enabled state	`Active`	`Enabled`
Target system status value for a user account that is in the Disabled state	`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 19, "Creating and Managing Generic Technology Connectors". While creating the generic technology connector, perform the following steps on the Step 3: Modify Connector Configuration page:

   Note: These steps are a condensed version of the procedure described in Section 19.2.4.3.1, "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 19.2.4.3.1, "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 19.2.4.3.1, "Adding or Editing Fields in Data Sets".

17.6 Validation Providers

Table 17–1 describes the validation providers that are shipped with 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 17–1 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
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 `GenericValidator` class in the Apache Jakarta Commons API.

18

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:

18.1 Role of Providers

The following sections discuss the role of providers during generic technology connector creation and use:

18.1.1 Role of Providers During Generic Technology Connector Creation

You create a generic technology connector by using the Identity System Administration. 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.

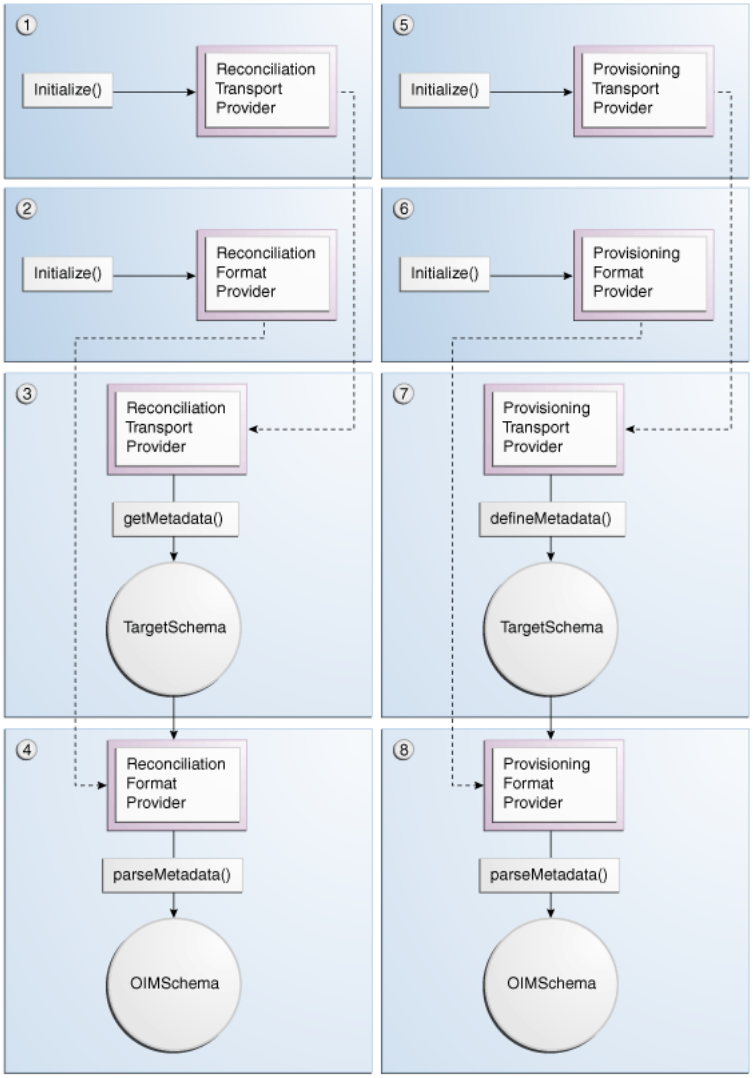
In the generic technology connector context, the term **metadata** refers to the set of identity fields that constitute the user account information. The term **metadata detection** refers to the reading and parsing of target system metadata by the providers.

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 18–1 shows the metadata detection process.

Figure 18–1 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 16.2.1, "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.

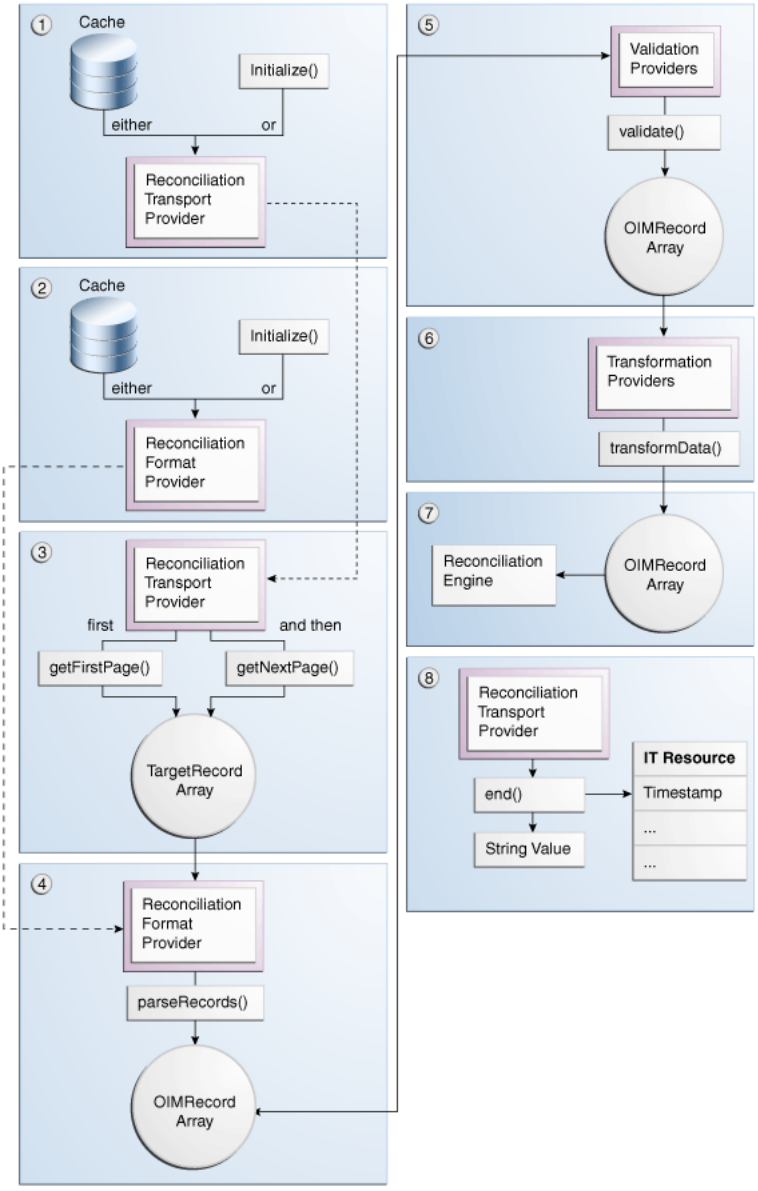
> **Note:** After a provider is initialized, it is stored in the Oracle Identity Manager cache until any one of the following events occurs:
>
> - Cache is purged.
> - Oracle Identity Manager is restarted.
> - The generic technology connector is modified after it is created.
>
> The validation providers 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.

18.1.2 Role of Providers During Reconciliation

Figure 18–2 shows the role of providers during reconciliation.

Figure 18–2 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 Identity System Administration 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.

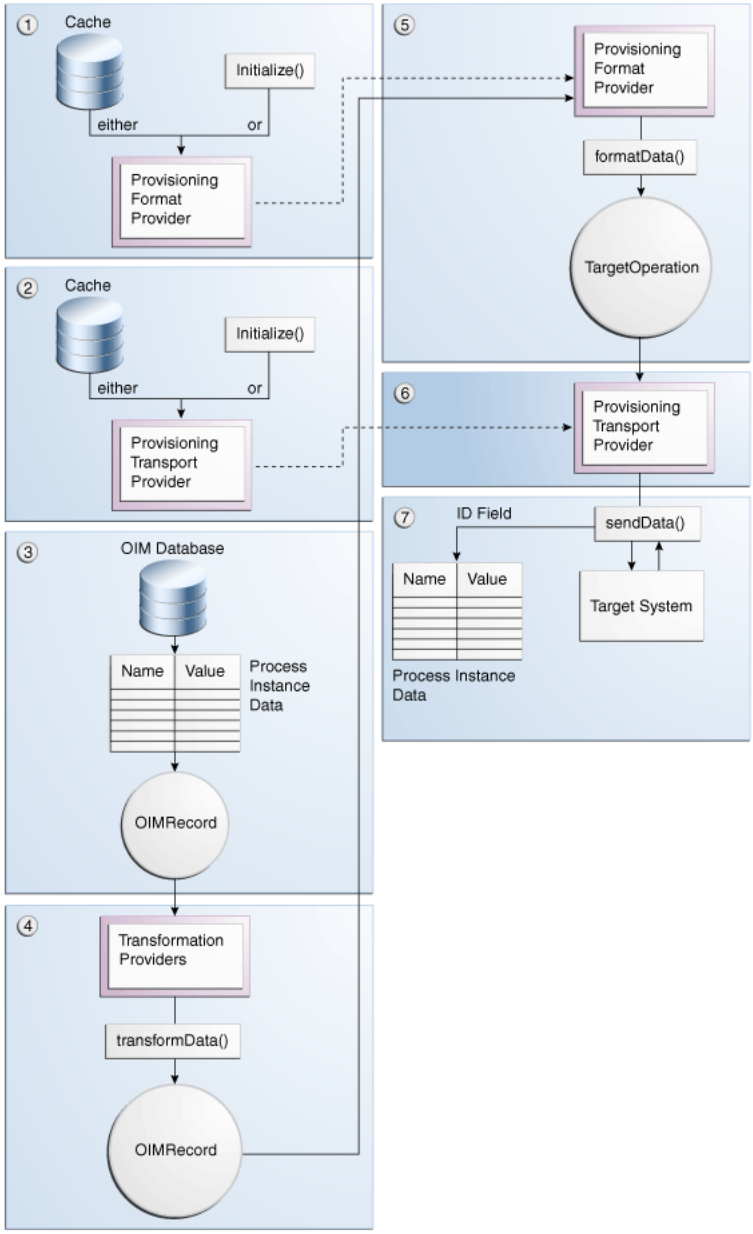
   If you do not select validation providers while creating the generic technology connector, Step 5 is not performed and each element of the `OIMRecord` value objects array is passed on to Step 6.
6. While creating the generic technology connector, if you selected transformation providers to modify data 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 16.2.1, "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.

18.1.3 Role of Providers During Provisioning

Figure 18–3 shows the role of providers during provisioning.

Figure 18–3 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 16.2.3, "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 16.2.2, "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 Identity System Administration.
   - If the operation fails at any stage after the name-value pairs are created, an error message is displayed on the Identity System Administration.

   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 Identity System Administration.
   - 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.

18.2 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

18.2.1 Determining Provider Requirements

Guidelines for determining provider requirements are as follows:

- Determining the Reconciliation Provider Requirements
- Determining the Provisioning Provider Requirements

18.2.1.1 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 18.3, "Reusing Providers"

18.2.1.2 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 18-15

18.2.2 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.

18.2.3 Developing Java Code Implementations of the Value Objects

Develop the Java code implementations of the value objects listed in Table 18–1. 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 18–1 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 18–1 (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`

18.2.4 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.

18.2.5 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 18–2 lists the modules that are specific to the supported provider types.

Table 18–2 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.

18.2.6 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 18–3 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 18–3 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: ■ `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: ■ `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: ■ `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: ■ `name`: Name of the provider ■ `class`: Name of the Java class of the provider implementation

Table 18–3 (Cont.) Elements of the Provider XML File

Element	Description
`Configuration`	Parent element of the configuration elements of any type of provider
`Parameter`	Element that provides information about a parameter of a provider The `Parameter` element has the following attributes: ■ `type`: Type of parameter, either `Runtime` or `Designtime` ■ `datatype`: Data type of parameter, either `String` or `Boolean`. Any parameter whose data type is not Boolean must be represented as a string. ■ `required`: Specifies whether or not the parameter is mandatory ■ `encrypted`: Specifies whether or not the parameter display must be encrypted ■ `name`: Name of the parameter ■ `datalength`: Character length of the parameter value
`DefaultAttribute`	Child element of the `Configuration` element This element must be included only in the `ProvFormatProvider` element. It has the following attributes: ■ `datatype`: Data type of parameter, either `String` or `Boolean`. Any parameter whose data type is not Boolean must be represented as a string. ■ `name`: Name of the parameter ■ `encrypted`: Specifies whether or not the parameter display must be encrypted ■ `size`: Size of the default attribute 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 `DefaultAttribute` element in the provisioning format provider XML file.
`Response`	Child element of the `Configuration` element This element must be included only in the `ProvFormatProvider`, `ProvTransportProvider`, `TransformationProvider`, and `ValidationProvider` elements. It represents the response returned from the providers that are called by the provisioning engine. This response is displayed on the Identity System Administration. This element has the following attributes: ■ `code`: Corresponds to the Oracle Identity Manager process task response code to be generated ■ `description`: Corresponds to the description of the Oracle Identity Manager process task response code to be generated **Note:** For a provisioning format provider or provisioning transport provider, you must ensure that the sum of the number of characters of the `name` attribute of the `ProvFormatProvider` or `ProvTransportProvider` element and the number of characters of the `Response` 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.

See Also: "Migrating User Modifiable Metadata Files" on page 37-1 for detailed information importing and exporting files and modifying Oracle Identity Manager metadata

18.2.7 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 Identity System Administration. 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 18.2.6, "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:

  ```
  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_
  required_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.description=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.

18.2.8 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 18-17 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 Oracle Identity System Administration.

ii. Under Configuration, click **Generic Connector**. 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 "Migrating JARs and Resource Bundle" on page 37-5 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.

18.3 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

18.3.1 Reusing Reconciliation Providers

As described in Section 18.1.2, "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.

18.3.2 Reusing Provisioning Providers

As described in Section 18.1.3, "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.

18.4 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 36, "Understanding Customization Types" 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 37, "Deploying and Undeploying Customizations" for information about the upload resource bundles and JAR utilities.

19

Creating and Managing Generic Technology Connectors

This chapter explains how to create and maintain Generic Technology Connectors. It contains these sections:

19.1 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 17, "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 18, "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 Chapter 8, "Using the Adapter Factory".

19.2 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 Identity System Administration to Create the Connector
- Configuring Reconciliation

- Configuring Provisioning
- Creating the Form and Publishing the Application Instance
- Enabling Logging

19.2.1 Determining Provider Requirements

As mentioned in Chapter 17, "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 16.2, "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 19.2.1, "Determining Provider Requirements"

19.2.2 Selecting the Providers to Include

Identify the predefined providers that can be used to meet your provider requirements. See Chapter 17, "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 18, "Creating Custom Providers for Generic Technology Connectors" for information about creating custom providers.

19.2.3 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`
- 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 Identity System Administration page for creating generic technology connectors.

Note: In an Oracle Identity Manager deployment that is integrated with Access Manager (OAM), the OIMSignatureAuthenticator authentication provider is not configured by default. If you use Oracle Identity Manager 9.x connectors, such as GTC, or if your custom code uses signature-based OIMClient login, then you must enable the OIMSignatureAuthenticator authentication provider.

For information about enabling OIMSignatureAuthenticator, see "OIMSignatureAuthenticator Not Configured for Oracle Identity Manager Domain Security Realm" in the *Oracle Fusion Middleware Release Notes*.

For authentication from custom code, see "Using OIMClient and tcUtilityFactory in Integrated Deployments" on page 31-2.

19.2.4 Using Identity System Administration to Create the Connector

To navigate to the first Identity System Administration page for creating a generic technology connector, login to Identity System Administration, and click **Generic Connector** under Configuration. In the Manage Connectors page, click **Create**.

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

19.2.4.1 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 16.4, "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 20.2.1, "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 17.4.1, "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 19–1 lists sample entries for the GUI elements on the Step 1: Provide Basic Information page.

Table 19–1 Sample Entries for 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

19.2.4.2 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 17, "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 17, "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 17-21 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 19.2.4.3.1, "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 19–2 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 19–2 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		Section 17.1, "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		Section 17.4, "Web Services Provisioning Transport Provider"
Web Service URL field	http://acme123:8080/spmlws/services/HttpSoap11	NA

Table 19–2 (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
Run-Time Parameters of the SPML Provisioning Format Provider		Section 17.3, "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		Section 17.1, "Shared Drive Reconciliation Transport Provider"
File Encoding field	Cp1251	NA
Design Parameters of the Web Services Provisioning Transport Provider		Section 17.4, "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		Section 17.3, "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
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.fffffffff	NA
Batch Size field	All	NA
Stop Reconciliation Threshold field	None	NA
Stop Threshold Minimum Records field	None	NA

Table 19–2 (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
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

19.2.4.3 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 18, "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.

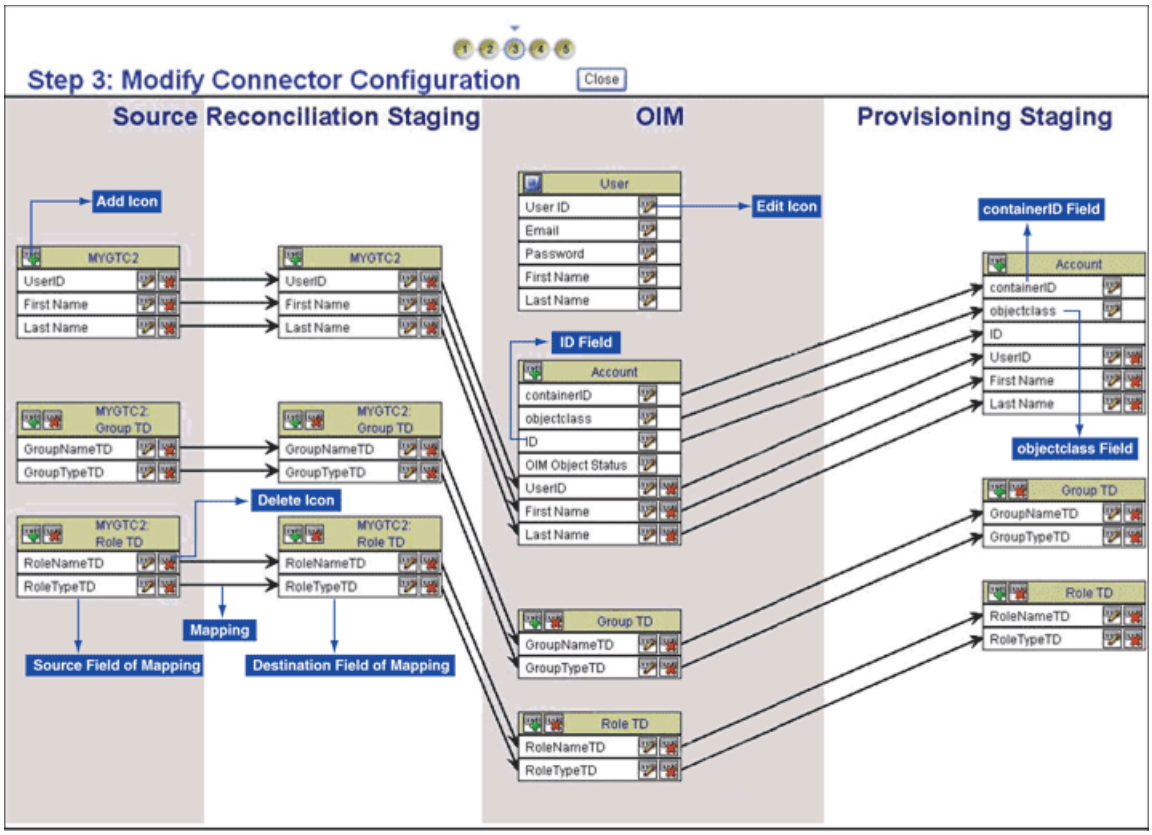
   The Web Services provisioning transport provider and SPML provisioning format provider cannot detect metadata from the target system. If you want custom providers to be able to detect metadata, you must ensure that:

   - 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 19–1 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 19–1 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. To overcome this issue, you must perform the following steps:

  1. Open the generic technology connector and navigate to Jgraph screen.
  2. In the Reconciliation Staging of the Jgraph screen, modify the field data type to **Date** for all the fields which holds date value.

3. Save the connector.

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 19–3, 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 19–3 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 *not* taken place	**Scenario 1** The following data sets are displayed: ■ Source ■ Reconciliation Staging ■ Oracle Identity Manager The fields that constitute the data sets are *not* displayed. 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.	**Scenario 2** The following data sets are displayed: ■ Source ■ Reconciliation Staging ■ Oracle Identity Manager ■ Provisioning Staging The fields that constitute the data sets are *not* displayed.	**Scenario 3** The following data sets are displayed: ■ Oracle Identity Manager ■ Provisioning Staging The fields that constitute the data sets are *not* displayed.
Metadata detection has taken place	**Scenario 4** The following data sets are displayed: ■ Source ■ Reconciliation Staging ■ Oracle Identity Manager The fields that constitute the data sets are displayed. 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.	**Scenario 5** The following data sets are displayed: ■ Source ■ Reconciliation Staging ■ Oracle Identity Manager ■ Provisioning Staging The fields that constitute the data sets are displayed.	**Scenario 6** The following data sets are displayed: ■ Oracle Identity Manager ■ Provisioning Staging The fields that constitute the data sets are displayed.

See Also: Section 20.1.2, "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

19.2.4.3.1 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 19.2.4.2, "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 19.2.4.3, "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 17.5, "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 19.2.4.3, "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 19.5.3.3, "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 19–4.

Table 19–4 Lookup Properties

Lookup Property	Value
Column Names	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. **Note:** If you select the Lookup Column Name property, you must also select the Column Names property, which is described later in this table. After you enter a value in the **Property Value** field, click **Submit**. The following SQL query can be used to illustrate how the Column Names and Lookup Column Name properties are used: `SELECT USR_FIRST_NAME, USR_LOGIN, USR_LAST_NAME FROM USR` Suppose you set the following as the values of the two properties: - Column Names: `USR_FIRST_NAME, USR_LAST_NAME` - Lookup Column Name: `USR_LOGIN` 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.
Column Captions	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**. After you enter a value in the **Property Value** field, click **Submit**.

Table 19–4 (Cont.) Lookup Properties

Lookup Property	Value
Column Widths	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. If the lookup window is going to display multiple columns, enter multiple column widths separated by commas. After you enter a value in the **Property Value** field, click **Submit**.

Table 19–4 (Cont.) Lookup Properties

Lookup Property	Value

Lookup Query	To specify a value for the Lookup Query property:
	1. In the **Property Value** field, enter the SQL query (without the `WHERE` 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 `WHERE` clause for the `SELECT` statement that you specify in Step 1:
	- Filter Column
	- Source
	- Field Name
	From the values that you select, the `WHERE` clause is created as follows:
	`WHERE Filter_Column=Source.Field_Name`
	4. Click **Save**.
	To correctly display the data returned from a query, you must add a `lookupfield.header` property to the `xlWebAdmin_`*`locale`*`.properties` file.
	For example, consider the following SQL query:
	`SELECT usr_status FROM usr`
	To view the data returned from the query, you must add the following entry to the `xlWebAdmin_`*`locale`*`.properties` files:
	`lookupfield.header.users.status=User Status`
	If the `xlWebAdmin_locale.properties` file does not contain a `lookupfield.header` property for your specified query, the Identity System Administration displays a lookup window after you click the corresponding lookup icon.
	The syntax for a `lookupfield.header` property is as follows:
	`lookupfield.header.`*`column_code`*`=display value`
	The *`column_code`* portion of the entry must be lowercase and any spaces must be replaced by underscore characters (_).
	By default, the following entries for lookup field column headers are already available in the `xlWebAdmin_locale.properties` file:
	`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`

Table 19–4 (Cont.) Lookup Properties

Lookup Property	Value
Lookup Code	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). To enter a lookup code, open the Lookup Definition form, query for the required code, and copy the code into the Property Value field. After you enter a value in the **Property Value** field, click **Submit**. **Note:** 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. If you want to implement lookup fields reconciliation, create a scheduled task that populates the lookup code.
Lookup Column Name	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. **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. After you enter a value in the **Property Value** field, click **Submit**.
Auto Complete	If you enter `True` 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 `New` in the field. When the user double-clicks the State lookup field, only states that begin with `New` (for example, New Hampshire, New Jersey, New Mexico, and New York) are displayed in the lookup window. If you do not want to let users filter the display of values in the lookup field, enter `False` in the **Property Value** field. The default value of the **Auto Complete** property is `False`. After you enter a value in the **Property Value** field, click **Submit**.

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 17.5.2, "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.

19.2.4.3.2 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.

19.2.4.3.3 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.

19.2.4.3.4 Removing Child Data Sets

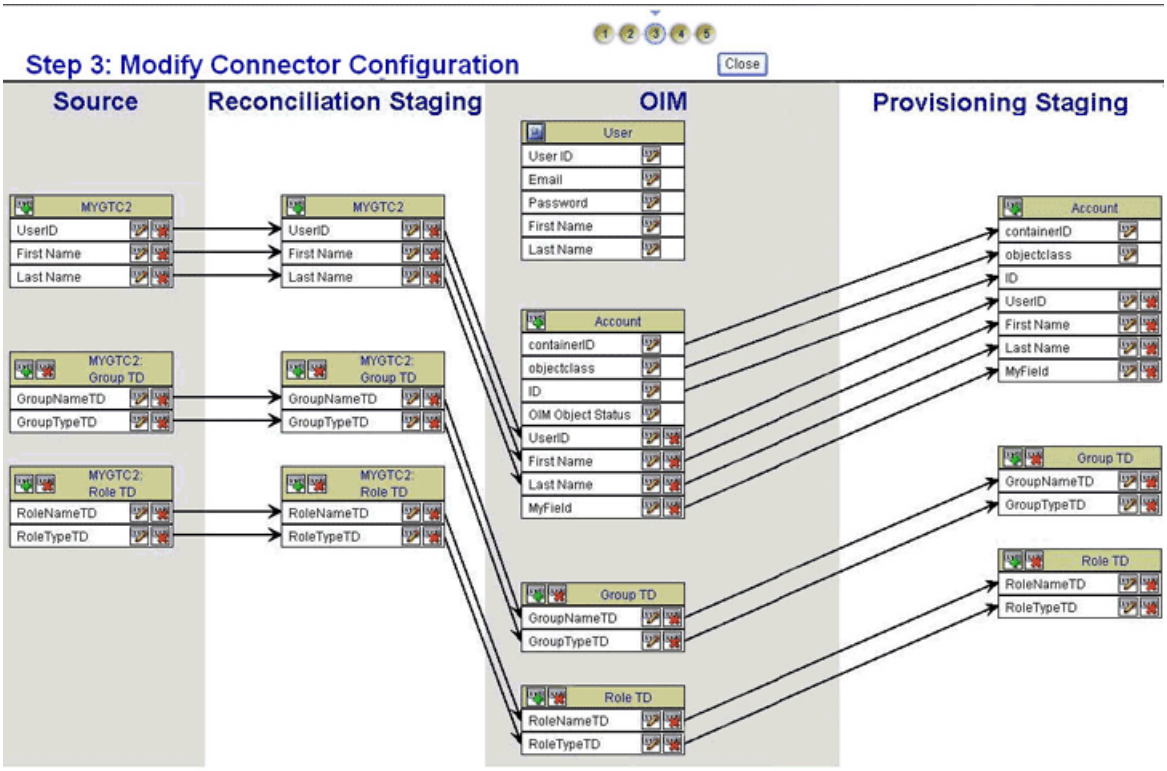
To remove a child data set:

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 19–2 shows the Step 3: Specify Connector Configuration page after the `MyField` field was added to the OIM - Account and provisioning staging data sets.

Figure 19–2 Step 3: Modify Connector Configuration Page After Addition of a Field



19.2.4.4 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 16.4, "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.

19.2.4.5 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 16.4, "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 20.1.1, "Creation Issues".

See Section 20.2.3, "Errors During Connector Creation" for a listing of error messages related to the creation process.

19.2.5 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.

19.2.6 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 5-15.

19.2.7 Creating the Form and Publishing the Application Instance

To create the form and publish the application instance, which is created when you select both the provisioning and reconciliation options on the Step 1: Basic Information page, perform the following steps:

1. Create a form specific to the GTC resource object.
2. Attach the form to the GTC application instance.
3. Publish the GTC application instance to the required organizations.

Note: To view a provisioned account in the new UI, the process form should have a field for IT resource. The value for this IT resource field should be populated during a reconciliation run.

19.2.8 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.

19.3 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

19.3.1 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.

- 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 16.4, "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 20, "Troubleshooting Generic Technology Connectors" discusses both these points.

To modify a generic technology connector:

1. Login to the Identity System Administration.
2. Under Configuration, click **Generic Connector**.
3. 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.
4. In the results that are displayed, click the generic technology connector that you want to modify.
5. 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, 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 20.2.2, "Step 3: Modify Connector Configuration Page".

19.3.2 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**.

19.3.3 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 18, "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 inSection 20.2.1, "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 19.5.7, "Modifying Generic Technology Connectors".

19.4 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

19.4.1 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 19–5 describes key methods of `ResourceConnection`:

Table 19–5 Methods of ResourceConnection

Method	Description
Create Connection	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 `itResourceInfoMap` is available as parameter with ITResource values to this method. The method returns the `ResourceConnection` which is the actual physical connection to the target.
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	This method returns `true` or `false` to indicate whether the connection is still valid. The Generic Connection Pool invokes the method if "validate connection on borrow" is set to `true`. It is invoked for connections that have been in the pool for some time. If the method returns `false`, the pool will discard that connection, create a new connection, and return to the requester.

19.4.2 Defining Additional ITResource Parameters

Table 19–6 lists other `ITResource` parameters for which you should provide appropriate values:

Table 19–6 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. *Note*: 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 19–5 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)

Table 19–6 (Cont.) ITResource Parameters

Field	Description	Sample Value and Notes
Connection pooling supported	Denotes whether pooling is supported. If pooling is not supported, returned connections will not be pooled connections. Recommended default is `true`.	true or false.
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: ■ 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 *not* 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 Identity System Administration or through the APIs.
- Avoid updating values when the pool is in use.

19.4.3 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);
```

19.4.4 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:** It 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 19–7 lists the methods invoked for the GenericPool interface:

Table 19–7 Methods of the GenericPool Interface

Method	Purpose
initializePool(PoolConfiguration poolConfig)	To initialize the pool. The PoolConfiguration data object contains all pool-related parameters.
borrowConnectionFromPool()	To request a connection.
returnConnectionToPool(Resource Connection resConn)	To return a connection to the pool.
refreshPool(PoolConfiguration newPoolConfig)	To refresh the pool with updated values.
destroyPool()	To remove the pool (for example when ITResource is deleted).

19.4.5 Example: Implementation of ResourceConnection

This example demonstrates an implementation of the `ResourceConnection` interface. Key methods are highlighted.

Example 19–1 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.ResourceConnectionValidationxception;
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;
               }
        }
}
```

19.5 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

19.5.1 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 17.1, "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 Identity System Administration for the same Oracle Identity Manager installation.

19.5.2 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.

19.5.3 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

19.5.3.1 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.

19.5.3.2 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.

19.5.3.3 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 Identity Self Service, Identity System Administration, or database.

  See also Section 20.1, "General Issues for Generic Technology Connectors".

19.5.3.4 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.

19.5.3.5 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.

19.5.4 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.

19.5.5 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.

19.5.6 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.

- 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 19.5.6, "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.

19.5.7 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 Identity System Administration for the same Oracle Identity Manager installation.

20

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

20.1 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

20.1.1 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.

  - 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.

- **Summery:**

  When you create a new GTC based on Database Application Table, configure connection parameters, and select the custom transformation provider in the reconciliation staging to apply a concatenation to the attribute, such as ID_ EMPLEADO, the custom transformation provider is visible and you can select it,

but when you click Continue to provide mapping information, an error is logged, and you are not able to provide parameters to the Transformation. The error is similar to the following:

```
<Sep 29, 2011 6:31:32 PM CDT> <Error>
<org.apache.struts.tiles.taglib.InsertTag> <BEA-000000> <ServletException in
'/gc/ModifyConnectorTransParamsTiles.jsp': null
javax.servlet.ServletException: java.lang.reflect.InvocationTargetException
```

Description:

The possible reason of this error is missing entries in the properties files. To resolve this issue, the resource bundles must be added in the following files:

- $*MW_HOME*/Oracle_IDM1/server/customResources/customResources_en.properties
- The corresponding translations for the other locales used

20.1.2 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:

  Some of the static text displayed on the Step 3: Modify Connector Configuration page are not 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`

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, see "Migrating User Modifiable Metadata Files" on page 37-1.

   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
```

20.1.3 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 Identity System Administration or database.

- 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.
- 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.

20.2 Configuration Issues for Generic Technology Connectors

This section contains these topics:

20.2.1 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 .

20.2.2 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.

  Therefore, 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.

- **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.

20.2.3 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.

20.2.4 Errors During Reconciliation

Table 20–1 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 20–1 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: ■ 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 17.1, "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.

Table 20–1 (Cont.) Common Errors Encountered During Reconciliation

Problem Description (Error Message)	Solution
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 17.1, "Shared Drive Reconciliation Transport Provider" for information about the header format. Fix the problem, and then retry reconciliation.
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.

20.2.5 Errors During Provisioning

Table 20–2 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 20–2 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.

Table 20–2 (Cont.) Common Errors Encountered During Provisioning

Problem Description	Solution
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.
Response Code: SPML Template Context Processing Error **Response Description:** An error was encountered while processing the template context for generation of SPML request.	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 Operation Name Missing **Response Description:** The operation name for provisioning is missing.	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.
Response code: SPML Provisioning Child Name Missing **Response Description:** The child name is missing.	You may have been trying to perform provisioning for one particular type (for example, role or membership) of multivalued attribute when this error occurred. 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.
Response code: SPML Provisioning Child Meta-Data Null **Response Description:** The child metadata list is null.	You may have been trying to perform provisioning for one particular type (for example, role or membership) of multivalued attribute when this error occurred. 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.
Response code: SPML Provisioning Child Metadata Problem **Response Description:** An error was encountered while sorting the child metadata list.	You may have been trying to perform provisioning for one particular type (for example, role or membership) of multivalued attribute when this error occurred. 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.
Response code: SPML Provisioning ID Missing **Response Description:** The unique ID is missing.	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.
Response code: SPML Provisioning Target ID Missing **Response Description:** The unique Target ID is missing.	Check the provider parameters that have been entered. TargetID may be missing.

Table 20–2 (Cont.) Common Errors Encountered During Provisioning

Problem Description	Solution
Response code: OIM API Error **Response Description:** An error was encountered in the Oracle Identity Manager API layer.	Check if Oracle Identity Manager is operating correctly for other operations. Check the connectivity between the Oracle Identity Manager front end and the database. **Note:** This error is not related to the providers that you use.
Response code: OIM Process Form Not Found **Response Description:** The process form was not found in Oracle Identity Manager.	The generic technology connector may not have been created correctly. Try creating another connector by using the same set of configurations. Try direct provisioning. **Note:** This error is not related to the providers that you use.
Response code: OIM Process Form Instance Not Found **Response Description:** The process form instance was not found for the specified form during update.	The provisioned instance information in the Oracle Identity Manager database may have become corrupted. Try direct provisioning. 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. **Note:** This error is not related to the providers that you use.
Response code: OIM Atomic Process Instance Not Found **Response Description:** The process instance found is not an atomic process.	The provisioned instance information in the Oracle Identity Manager database may have become corrupted. Try direct provisioning. 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. **Note:** This error is not related to the providers that you use.
Response code: Column Not Found **Response Description:** An expected column was not found in the result set.	The generic technology connector may not have been created correctly. Try creating another connector by using the same set of configurations. Try direct provisioning. **Note:** This error is not related to the providers that you use.
Response code: Invalid Provider **Response Description:** The provider name specified is invalid.	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: PROVIDER_DEF_XSD_LOCATION = "/db/GTC/Schema"; PROVIDER_DEF_XML_LOCATION = "/db/GTC/ProviderDefinitions";
Response code: IT Resource Instance Not Found **Response Description:** The IT resource instance was not found in Oracle Identity Manager.	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. **Note:** This error is not related to the providers that you use.

Table 20–2 (Cont.) Common Errors Encountered During Provisioning

Problem Description	Solution
Response code: Version Not Found **Response Description:** The required process form version was not found in Oracle Identity Manager.	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. **Note:** This error is not related to the providers that you use.
Response code: Version Not Defined **Response Description:** The required process form version was not defined in Oracle Identity Manager.	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. **Note:** This error is not related to the providers that you use.
Response code: Web Service Not Found **Response Description:** The Web service was not found on the target server. Check the service name and IP address.	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.
Response code: Web Service Connection Refused **Response Description:** The Web service connection could not be established. Check that the server is running and the specified port is correct.	Check if the Web service is running.
Response code: Web Service No Such Method **Response Description:** The Web service method could not be started. Check the operation name and parameters.	Check the operation name and parameters.
Response code: Web Service Null Parameter Value **Response Description:** The parameter value passed to the Web service is null.	Check if the provisioning process ran correctly. The provisioning format provider may not have run correctly and, therefore, may have generated NULL output.
Response code: Web Service HTTP Library Missing **Response Description:** The Web service HTTP library is not included in the classpath.	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: Web Service Null Result Value **Response Description:** The Web service result value is null.	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.

Table 20–2 (Cont.) Common Errors Encountered During Provisioning

Problem Description	Solution
Response code: Web Service Invocation Issue **Response Description:** An error was encountered while invoking the Web service.	Check the credentials of the Web service.
Response code: Web Service Target URL Missing **Response Description:** The Web service target URL required to invoke the Web service is missing.	Check the values of the provider parameters. The Web service URL may be missing. Modify the generic technology connector and provide this value again.
Response code: Web Service Target Method Name Missing **Response Description:** The Web service target method name required to invoke the Web service is missing.	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.
Response code: Web Service Response XML Parsing Error **Response Description:** An error was encountered during XML parsing of the Web service response.	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.
Response code: Web Service Response ID Error **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.	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 `psoID` created in the target system. The provider specification for the Web Service provider expects the return of the psoID field.
Response code: Web Service Protocol Connection Error **Response Description:** An error was encountered in the Oracle-SOAP HTTP connection.	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.
Response code: Web Service Protocol Processing Error **Response Description:** An error was encountered while calling the Oracle-SOAP API.	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 20–2 (Cont.) Common Errors Encountered During Provisioning

Problem Description	Solution
Response Code: Unable to parse the date **Response Description:** Error encountered while parsing the date.	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
Response Code: Data Access Error **Response Description:** A data access error occurred while executing the query or loading the result set.	Check if Oracle Identity Manager is operating correctly for other operations. Check the connectivity between the Oracle Identity Manager front end and the database. **Note:** This error is not related to the providers that you use.
Response Code: SSL Handshake Did Not Happen **Response Description:** An SSL handshake did not happen during the secure communication with the target Web service.	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.
Response Code: Error in Initialization of SSL-Related Properties **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.	Check the configuration entries corresponding to the `RMSecurity` element of the `oim-config.xml` file.
Response Code: Invalid Web Service Keystore or password **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.	Check the configuration entries corresponding to the `RMSecurity` element of the `oim-config.xml` file.
Response Code: Error Encountered During Web Service Keystore Initialization **Response Description:** Keystore initialization failed. Credentials of the keystore are mentioned in the `oim-config.xml` file under the "RMSecurity" element.	Check the configuration entries corresponding to the `RMSecurity` element of the `oim-config.xml` file.

Table 20–2 (Cont.) Common Errors Encountered During Provisioning

Problem Description	Solution
Response Code: Invalid ID **Response Description:** An invalid ID is present in the input SPML request.	Check the value specified for the `Target ID` parameter.
Response Code: Object already exists **Response Description:** This object already exists in the target system.	Check if the object that you are trying to create already exists on the target system.
Response Code: Operation Not Supported **Response Description:** The requested provisioning operation is not supported.	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.
Response Code: Invalid ID Type in Input SPML Request **Response Description:** An invalid ID type is present in the input SPML request.	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.
Response Code: ID in Input SPML Request Does Not Exist in the Target System **Response Description:** The ID in the input SPML request does not exist in the target system.	Ensure that the `psoID` value that was sent in the request exists in the target system.
Response Code: Requested Execution Mode Not Supported **Response Description:** The requested execution mode is not supported.	Ensure that the target system supports the execution of requests in synchronous mode.
Response Code: Invalid Container **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.	Check if a container corresponding to the container ID specified in the request exists on the target system.

Table 20–2 (Cont.) Common Errors Encountered During Provisioning

Problem Description	Solution
Response Code: Nonstandard SPML Error **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.	Check the value of the `errorMessage` element in the SPML response. This element contains the target system error message that was generated when the error was encountered.
Response Code: SPML Response Is for Asynchronous Mode **Response Description:** The SPML response is for asynchronous mode, which is not supported.	Ensure that the target system sends responses corresponding to the synchronous mode of request execution.
Response Code: Error Encountered While Parsing Constituent Elements of Web Service URL **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.	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.
Response Code: SPML Response failed V2 schema validation **Response Description:** SPML Response received is not compliant with the SPML V2 standard specifications.	Ensure that the SPML response returned by the target system conforms to the SPML V2 standard specification.

Part IV

Requests and Approval Processes

This part contains chapters describing how to configure requests and SOA composites.

It contains the following chapters:

- Chapter 21, "Developing Workflows for Approval and Manual Provisioning"
- Chapter 22, "Using Segregation of Duties (SoD)"

21

Developing Workflows for Approval and Manual Provisioning

This chapter describes the concepts, features, and architecture of workflows in Oracle Identity Manager. It provides use cases for workflow, and instructions for designing, implementing, and deploying your first workflow. In addition, this chapter describes how to extend the request management operations by using plug-in points.

This chapter contains the following sections:

21.1 Introducing Workflows

This section describes the key workflow concepts in the following sections:

21.1.1 Overview of Workflows

Managing user access and orchestrating the business process so that users get the correct access is a key identity governance function. The process of changing users' access can be initiated by the users through events in HR that trigger policies, or by administrators. Irrespective of how the change in access is initiated, organizations require the following:

- The business process that is initiated must be flexible, and must be able to meet changing business rules of the organization.

- The business process must be able to decide between granting access immediately versus introducing manual intervention steps and seeking approval prior to granting access.
- The business process must be able to perform validations, including Segregation of Duties (SoD) checks on what is being requested, by who, for whom, and in what context.
- If manual intervention is required, then the business process must have the ability to assign to users or groups of users and escalate, reassign, or expire if no response is received in a timely manner.
- For manual intervention, the business process must have the ability to gather information from the approvers, including comments and attachments.
- The business process must be able to interact with external systems, such as ticketing systems, when automated access grants are not possible, or the organization's rules require that access is granted manually.
- All decisions and actions must be audited and available in a reportable manner to allow the organization to measure performance of the process and also for auditors to fulfill compliance requirements

Oracle Identity Manager provides flexible and powerful access request capabilities that allow organizations to meet these requirements.

21.1.2 Workflow Concepts

The key concept of workflows in Oracle Identity Manager involves the following terminologies:

- **Request**

  In Oracle Identity Manager, a request refers to the business process that is invoked when an operation on an identity or an account has to be performed. Examples of these operations include creating a user, provisioning an account, and granting a role to a user. A request can either be fulfilled immediately (also known as direct operation) or can require manual intervention in the form of approvals (also known as request-based operation). When a user tries to perform an operation, Oracle Identity Manager determines whether the operation would be direct or request-based on the authorization policies of the logged-in user.

- **Request-level approval**

  A request has to go through two independent approval processes before it can proceed for fulfillment. These are request-level approvals and operational-level approvals. Request-level approvals are invoked first and are followed by operational-level approvals. Request-level approvals are primarily used for bulk requests, which involve multiple target users or multiple requested entities or any combination of both.

  For bulk approvals, when the request-level approval has been received, the request engine splits the request into individual target user and requested operation or entity combination, and invokes the operational approval for each combination.

- **Operation-level approval**

  Operation-level approval is the second approval process that is invoked as part of approving a request. Operational approvals are always for a single target user and a single operation or requested entity.

- **Approval policy**

  An approval policy is a rule that allows the request engine to pick a SOA composite to invoke. Approval policies can be configured at request level or operation level. They can use all the data available at the request and operational levels to construct a rule. The rule helps the request engine determine if the request should be auto-approved or a SOA composite should be invoked.

- **SOA composite**

  A SOA composite is an assembly of services, service components, and references designed and deployed together in a single application. Wiring between the service, service component, and reference enables message communication. The composite processes the information described in the messages.

- **Partner Link**

  A partner link enables you to define the external services with which the BPEL process service component is to interact. You can define partner links as services or references (for example, through a JCA adapter) in the SOA Composite Editor or within a BPEL process service component in Oracle BPEL Designer.

- **BPEL process**

  BPEL processes provide process orchestration and storage of synchronous or asynchronous processes. You design a business process that integrates a series of business activities and services into an end-to-end process flow.

- **IT provisioner**

  The IT provisioner, also known as fulfillment user or Help Desk user, is the persona responsible for fulfilling manual provisioning requests.

- **Request web service**

  The request web service is a web service that is shipped with Oracle Identity Manager. It allows customers to expose request, user, role, organization, account, entitlement, application instance, and catalog information so that approval workflows can make data-driven routing decisions.

- **Request callback**

  The request callback is a web service that is invoked by the SOA composite when an approval outcome (approve/ reject) has been received. When the request engine invokes a SOA composite for the purpose of approval, it suspends the request until the composite invokes the request callback and provides an approve or reject decision. This decision allows the request engine to proceed with fulfilling the request (if approved) or rejecting the request (if rejected).

- **Provisioning callback**

  The provisioning callback is a web service that is invoked as part of disconnected provisioning. When the IT provisioner or fulfillment user fulfills a disconnected provisioning request and marks the task as completed, the SOA composite invokes the provisioning callback and sends the provisioning status allowing the provisioning workflow to complete.

- **Request payload**

  The request engine invokes the SOA composite and passes it some basic information about the request, requester, and target user. This information is called the request payload.

- **Human Task**

Human tasks provide workflow modeling that describes the tasks for users or groups to perform as part of an end-to-end business process flow.

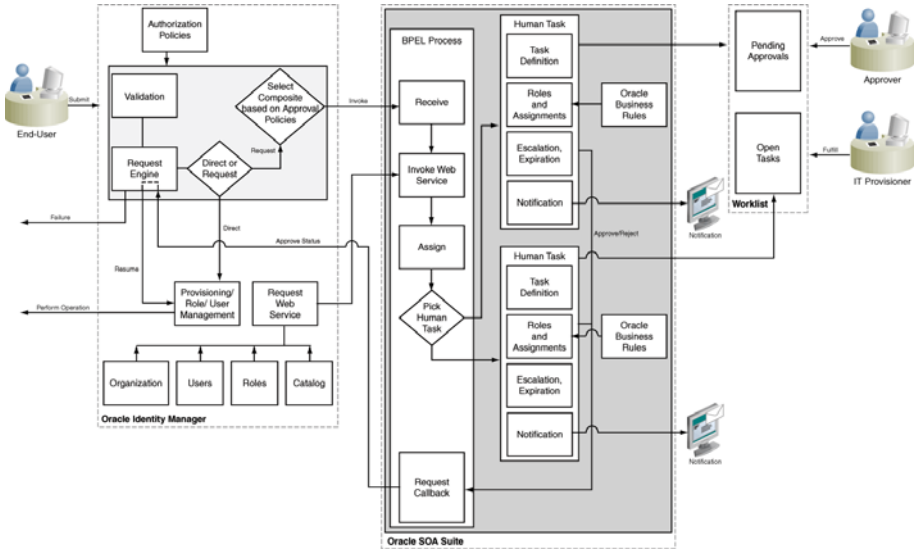
21.1.3 Workflow Architecture

Workflows are used in Oracle Identity Manager to:

- Route requests to approvers for approval
- Route manual provisioning tasks to IT provisioners or Help Desk for fulfillment

Figure 21–1 provides an overview of workflows in Oracle Identity Manager:

Figure 21–1 Workflow Architecture



In Figure 21–1, the following actions occur:

1. User initiates an operation that results in a request. Examples of such operations include:
   - Self-registration
   - User profile modification, excluding lock, unlock, and password management operations
   - Role grant operations
   - Application instance operations, including disconnected provisioning
   - Entitlement operations
   - Bulk operations
2. A request is created. After appropriate validation, the request engine evaluates approval policies and selects a SOA composite to be invoked.
3. If approval policies are not configured, then the default request-level SOA composite is selected for request-level approval, and default operational-level SOA composite is selected for operational-level approval.
4. The SOA composite involves the Business Process Execution Language (BPEL) process.

5. The BPEL process invokes a web service to get additional details about the request including:

   Note: This step is optional. This is required only if additional information related to various entities is required in BPEL Process.

   - Item details from the catalog
   - Target user information
   - Requester information

6. The BPEL process invokes additional logic to calculate properties such as priority, approvers, and notification.
7. When manual intervention is required, such as during approval and manual fulfillment, the process invokes a Human Task.

   A Human Task contains the logic to assign, expire, or escalate the approval task to users or roles. The Human Task can assign the users and roles statically or dynamically. For static assignments, the approvers can be determined in the BPEL process and passed as parameters to the Human Task. For dynamic assignments, rules created using Oracle Business Rules (OBR) are used to dynamically determine the approvers.

   Typically, the BPEL process contains one Human Task. In some instances, the BPEL process might invoke a decision point to pick one of multiple Human Tasks.
8. When the human task completes, a response of approve or reject (for approval) or complete (for manual fulfillment), is returned via a callback service to Oracle Identity Manager, which resumes the operation.

21.2 Predefined SOA Composites

Table 21–1 lists the predefined SOA composites in Oracle Identity Manager that can be used as approval processes.

Table 21–1 Predefined Workflow Composites

Workflow Composite	Description
DefaultRequestApproval	This is the default request-level approval. By default, the request-level approval goes to the SYSTEM ADMINISTRATORS role, for request-level approval. In addition, this composite is invoked by certification use cases. The task will have one of the following states: ■ Assigned to the beneficiary. Later, the task may be assigned to the beneficiary's manager based on the decision of the beneficiary. ■ Auto-approved if the certification requester is beneficiary's manager. **Note:** For information about the certification use cases, see "Managing Identity Certification" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager*.

Table 21–1 (Cont.) Predefined Workflow Composites

Workflow Composite	Description
DefaultOperationalApproval	This is the default operation-level approval. By default, the approval task is assigned to the SYSTEM ADMINISTRATORS role for operation-level approval. In addition, the composite is invoked by certification use cases, and the task will be auto-approved.
BeneficiaryManagerApproval	This requires approval from the beneficiary's manager. This can be associated with the following: ■ The request models that have a beneficiary. Examples of such request models are Provision Application Instance and Assign Roles. ■ All user models except Create User and Self-Register User. This composite must be associated at the operational level of approval because a request can have multiple beneficiaries at the request level.
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. **Note:** This composite cannot be associated with unauthenticated request models, such as Self-Register User.
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 models to provision or modify resources at the operational level if SoD check is required.
DisconnectedProvisioning	This SOA composite assigns the task to the System Administrator to fulfil the disconnected provisioning.
ProvideInformation	This SOA composite assigns the task to the requester seeking details of account/entitlement. For more information, see "Entitlement and Account Dependencies in Requests" in the *Oracle Fusion Middleware User's Guide for Oracle Identity Manager*.
CertificationProcess	This is the default Certification composite. This composite takes care of assigning the certification task to the certifier (user). This composite also manages the following certification task events: ■ Expiry ■ Proxy ■ Escalation ■ Re-assignment
CertificationOverseerProcess	This composite assigns a certification task to the certifier (user). In addition, the composite also handles routing the task to the overseer after the certifier completes the task. Oracle SOA Business Rules are used to handle the task routing. This composite handles the following certification task events: ■ Expiry ■ Proxy ■ Routing (Overseer) ■ Escalation ■ Re-assignment

21.3 Creating New SOA Composites

To create a new SOA composite that can be used as an approval process, 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

21.3.1 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 de 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.

21.3.2 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: The composite should be redeployed with a new version. If a composite is redeployed with the same version 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.

21.3.3 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*

21.4 Developing Workflows: Vision Request Tutorial

This section describes how to design your first workflow in the following sections:

- Introducing the Tutorial
- Prerequisites
- Creating the Application Instance
- Configuring FinApp in the Catalog
- Creating and Configuring the SOA Composite for Approval

21.4.1 Introducing the Tutorial

This tutorial is based on the following use case:

- Vision Corp uses FinApp, a mainframe-based application. The application does not have APIs that can be remotely invoked. Therefore, accounts are managed manually by the Help Desk.
- Vision Corp employees use the Access Request Catalog to request accounts and entitlements in the application.
- Approvals are based on the risk level of the access being requested. If the risk level is Low Risk, approval is required only from the beneficiary's manager. If the risk level is Medium Risk, approval is required from either the beneficiary's manager or certifier. If the risk level is High Risk, approval is required from the beneficiary's manager and the Audit Review team.
- After approval, the request has to be fulfilled by members of the Asset Management team.

This tutorial describes how to create the application and the workflow, and how to configure the approval and fulfillment for the application.

The result of the tutorial is:

- An application instance
- A SOA composite for approval consisting of:
  - A BPEL process
  - Multiple Human Tasks

21.4.2 Prerequisites

The following assumptions are made for this tutorial:

- Oracle SOA Suite is installed on a host on which the SOA infrastructure is configured.
- JDeveloper 11.1.1.7 with SOA Design Time 11.1.1.7 is available.
- Mandatory patches, if any, for SOA Jdeveloper extension have been applied. For information about the mandatory patches, see "Mandatory Patches Required for Installing Oracle Identity Manager" in the *Oracle Fusion Middleware Release Notes*.
- You are familiar with basic BPEL constructs, including BPEL activities and partner links, and basic XPath functions.
- You are familiar with the SOA Composite Editor and Oracle BPEL Designer, the environment for designing and deploying BPEL processes. However, for detailed information about SOA composites, refer to *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.
- Two roles, Audit Review Team and Asset Management Team, have been created and members have been assigned.
- An organization with name Vision is created.

21.4.3 Creating the Application Instance

This section contains the following topics:

21.4.3.1 Creating the FinApp Application Instance

To create the FinApp application instance:

1. Login to Oracle Identity System Administration.
2. Click Sandboxes to access sandbox management, create a sandbox, and activate it. See "Managing Sandboxes" on page 30-4 for information about sandboxes and how to create, activate, and publish sandboxes.
3. Under Configuration, click **Application Instances**. Click **Create** on the toolbar to open the Create Application Instance page.
4. Enter Name and Display Name as **FinApp**.
5. Select the **Disconnected** option to specify a disconnected application instance.
6. Click **Save**, and then click **OK** to confirm creation of the FinApp application instance.

21.4.3.2 Defining Application Instance Attributes and Creating a Form

To define the attributes of the application instance and create a form:

1. Under Configuration, click **Form Designer**.

2. Search and select the FinApp form. This form is automatically created when the disconnected application instance is saved.

   Note: You must be in an active sandbox to create and edit a form.

3. Click the **Fields** tab, and then click the Edit icon on the toolbar.

   By default, the following fields are created and are available for use:

Field	Description
IT Resource	The IT resource instance where the account is being created
Account Login	The login for the application
Password	The password that is used while logging in to the application
Account ID	The unique identifier generated by the application when the account is created
Service Account	A flag that is used during access request only

   Note: Attributes such as Account ID and IT Resource are typically not displayed in the access request user interface. Depending upon the use case, for example a mobile phone request, the attributes might not be relevant. To hide these attributes, you can customize the form. See "Configuring Custom Attributes" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager* for more information on how to customize the form.

4. Add additional attributes. In this example, add the following attribute:

   Account Description: Data type is Text.

   Note: See "Configuring Custom Attributes" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager* for more information on creating the custom attributes

5. After adding the attributes, verify that the configuration in the Fields tab is similar to the following table:

Display Label	Name	Type
Account Description	AccountDescription	Text
Account ID	UD_FINAPP_ACCOUNTID	Text
ITResource	UD_FINAPP_IT	Number
Account Login	UD_FINAPP_LOGIN	Text
Password	UD_FINAPP_PASSWORD	Text
Service Account	serviceaccount	Checkbox

6. To allow users to request entitlements, you must add a child object and add an attribute that is tagged as an Entitlement. To do so:

a. Click the **Child Objects** tab, and then click **Add** on the toolbar.

b. Enter the child object name, and click **OK** to create the child object.

c. Click the child object just created.

d. Select **Action**, **Create** to create a new attribute. From the popup window, select **Lookup**, and click **OK**. Enter values for the following fields:

   Name: Profile Name

   Display Name: Profile Name

e. Select **Use in Bulk** to allow requesters to specify a value when requesting access for multiple users.

f. Under Lookup Type, click **Create a New Lookup Type**.

g. Create the new Lookup and specify the values, as shown:

   Code: Lookup.FinApp.Profile

   Meaning: Lookup.FinApp.Profile

   Description: Lookup.FinApp.Profile

h. Create three lookup codes by using the values given in the following table:

Meaning	Code
FinApp User	FinAppUser
FinApp Administrator	FinAppAdministrator
FinApp Operator	FinAppOperator

Note: You can also populate the lookup definition by using a scheduled task and the lookup APIs.

i. Select the **Searchable**, **Entitlement**, and **Searchable Picklist** options.

7. Click **Save and Close**.
8. Click **Back to Parent Object**.
9. Click **Regenerate View** to re-create the UI form with the new attributes.
10. Close all tabs.
11. Publish the sandbox.

21.4.3.3 Publishing the Application Instance to One or More Organizations

To publish the application instance to one or more organizations:

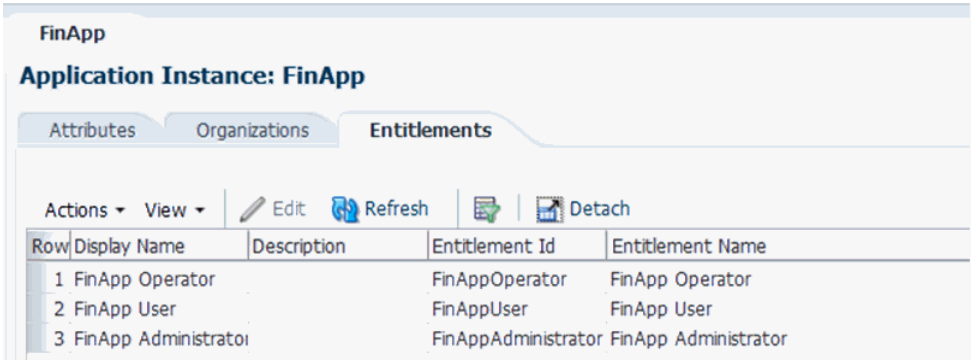
1. Open the FinApp application instance details page, and click the Organizations tab.
2. Click **Assign**. In the Select Organizations dialog box, select one or more organizations to publish the application instance to.
3. Select the **Hierarchy** option if you want the application instance to be published to the organization and its child organizations.
4. Click **OK**.

21.4.3.4 Linking Entitlements to the Application Instance

To link entitlements to the application instance:

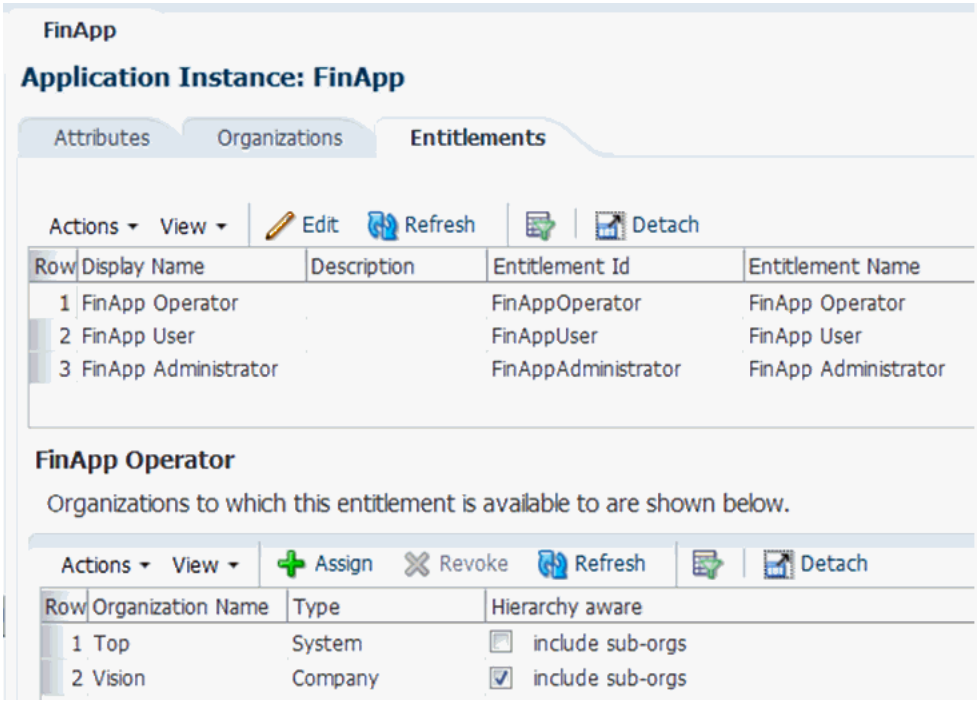
1. Under System Management, click **Scheduler**.
2. Search for the Entitlement List scheduled job, and click **Run Now**.
3. Under Configuration, click **Application Instances**, and navigate to the FinApp application instance.
4. Click the **Entitlements** tab, and verify that the entitlements are displayed, as shown in Figure 21–2:

Figure 21–2 Entitlements List



5. Select an entitlement, and verify that it is published to the same organizations as the application instance, as shown in Figure 21–3:

Figure 21–3 Entitlement Availability to Organizations



6. Edit one or more entitlements, and enter a business friendly description. If required, modify the display name as well.

21.4.3.5 Publishing the Application Instance With Entitlements to the Catalog

To publish the application instance and its entitlements to the catalog:

1. Under System Management, click Scheduler.
2. Search for the Catalog Synchronization scheduled job, and click **Run Now**.

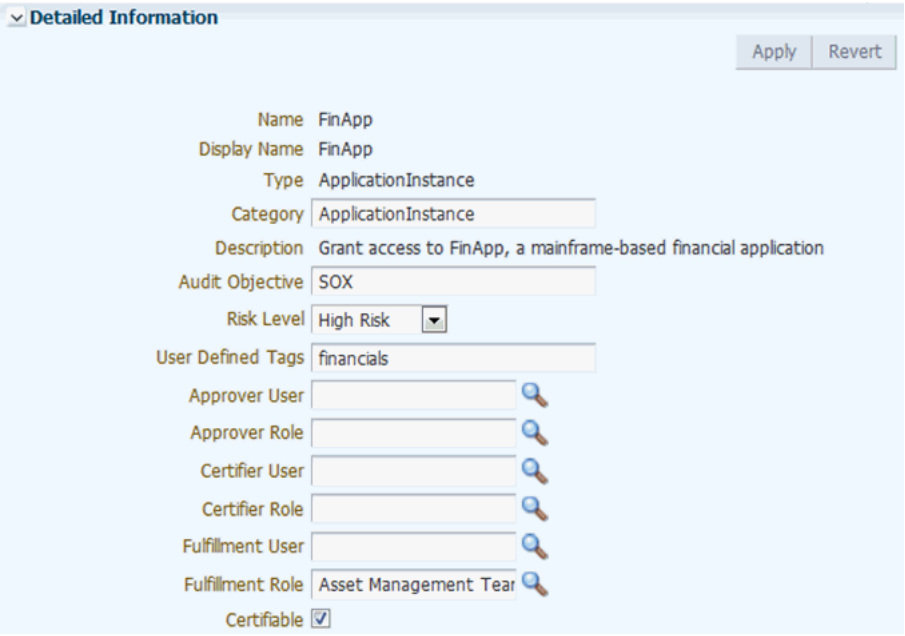
21.4.4 Configuring FinApp in the Catalog

To configure the application instance and its entitlements in the catalog:

1. Login to Oracle Identity Self Service as the Catalog Administrator.
2. Under Requests, click **Catalog**.
3. In the Catalog page, search for the application instance.
4. Select the application instance, and edit the catalog item details.
5. Provide values for the default attributes. Because this tutorial involves workflow routing based on risk level and manual fulfillment, you must provide a value for the Risk Level and Fulfillment Role attributes. However, it is recommended that you provide values for other attributes, especially User Defined Tags.

   Figure 21–4 shows the attributes of the catalog item.

Figure 21–4 Catalog Item Attributes



21.4.5 Creating and Configuring the SOA Composite for Approval

This section contains the following topics:

- Creating the Approval Workflow
- Making Request and Catalog Data Available to the BPEL Process
- Configuring Workflow Selection
- Configuring Human Tasks
- Configuring the Human Task and BPEL Mappings

- Deploying the SOA Composite
- Creating the Approval Policies

21.4.5.1 Creating the Approval Workflow

To create a new approval workflow:

1. Set the JAVA_HOME environment variable by running the setWLSEnv.sh script in the /server/bin/ subdirectory in the WebLogic Server installation directory.
2. Set the ANT_HOME environment variable to *MIDDLEWARE_HOME*/modules/org.apache.ant_1.7.1.
3. Set the PATH environment variable to $JAVA_HOME/bin:$ANT_HOME/bin:$PATH.
4. Navigate to *OIM_HOME*/server/workflows/new_workflow.
5. Run the following:

   ```
   ant-f new_project.xml
   ```

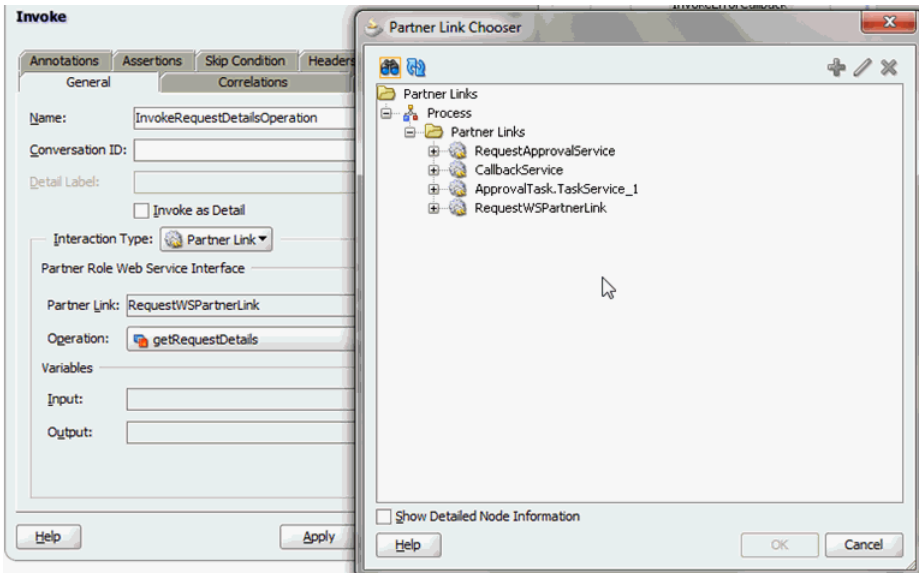
6. Provide the Application Name as AddAccessApprovalApplication.
7. Provide the Project Name as AddAccessApproval.
8. Provide the Service Name as AddAccess.
9. Wait for the utility to finish generating the new JDeveloper Workspace containing the Composite. The workspace is generated in /server/workflows/new-workflow/process-template.
10. Copy the directory to a location accessible to JDeveloper.

21.4.5.2 Making Request and Catalog Data Available to the BPEL Process

To make request and catalog data available to the BPEL process:

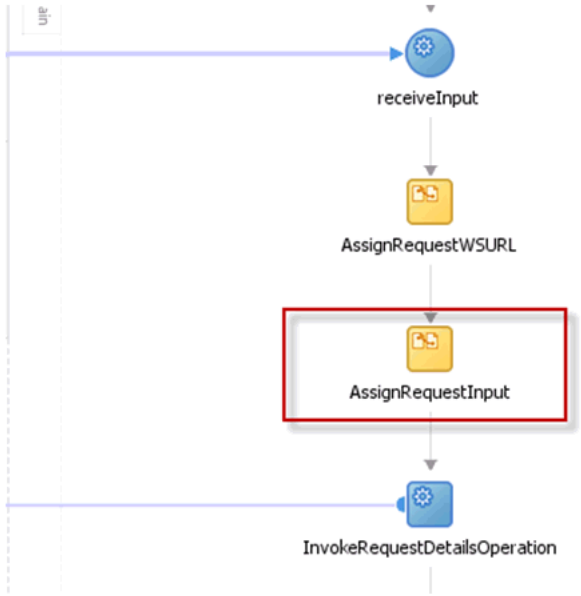
1. Switch to Design view of the BPEL process.
2. Drag the Invoke activity from the Component Palette and drop it below the AssignRequestWSURL activity. Rename it to `InvokeRequestDetailsOperation`.
3. Right-click **InvokeRequestDetailsOperation**, and select **Edit**.
4. Select partner link from the Partner Link Chooser as RequestWSPartnerLink, and operation as getRequestDetails, as shown in Figure 21–5.

Figure 21–5 Partner Link and Operation



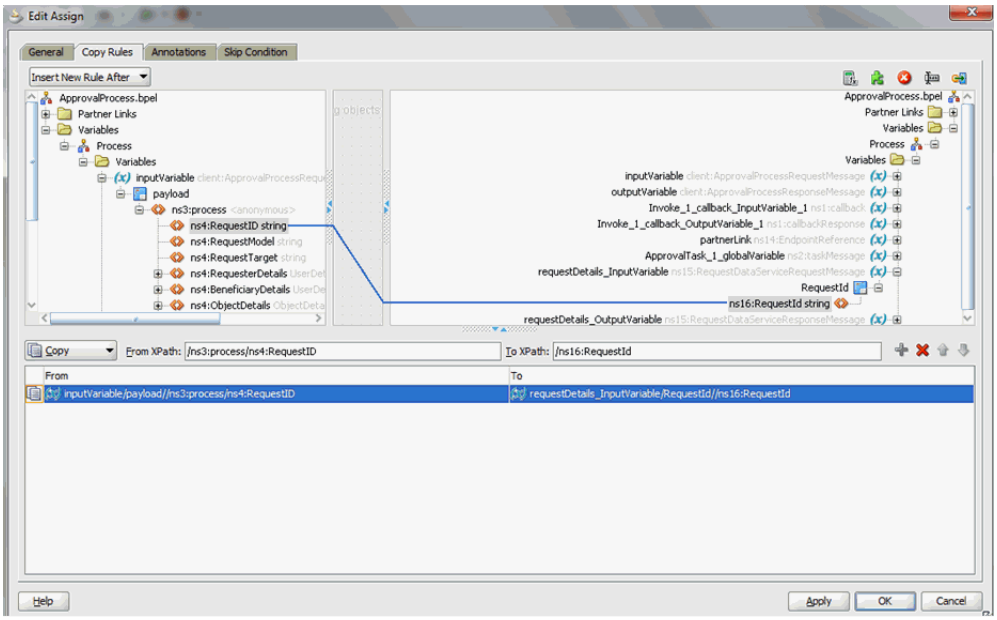
5. Under the Variables section, click the plus (+) icon for the Input and Output fields to create the input and output variables. Name the input and output variables as `requestDetails_InputVariable` and `requestDetails_OutputVariable` respectively. Then click **Apply** and **OK**.
6. Drag and drop an assign activity, rename it to `AssignRequestInput`, and place it above the InvokeRequestDetailsOperation invoke activity, as shown in Figure 21–6.

Figure 21–6 AssignRequestInput



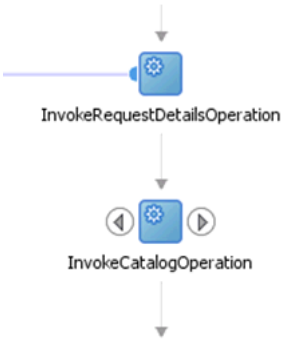
7. Right-click **AssignRequestInput**, and select **Edit** to map the input of the InvokeRequestDetailsOperation, as shown in Figure 21–7.

Figure 21–7 Input Mapping



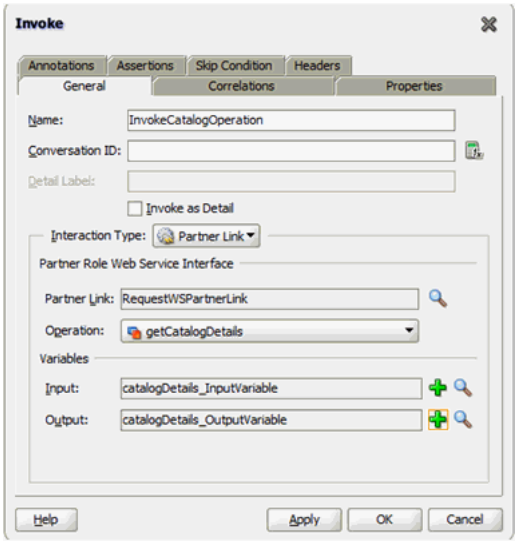
8. Add an Invoke activity after the InvokeRequestDetailsOperation, as shown in Figure 21–8. Name the activity `InvokeCatalogOperation`.

Figure 21–8 InvokeCatalogOperation



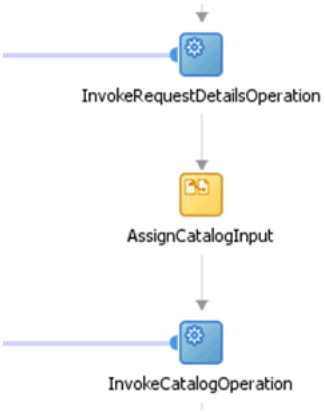
9. Edit the InvokeCatalogOperation, and configure it as shown in Figure 21–9.

Figure 21–9 InvokeCatalogOperation Configuration



10. Add an Assign activity above InvokeCatalogOperation, as shown in Figure 21–10. Name the activity as AssignCatalogInput.

Figure 21–10 AssignCatalogInput

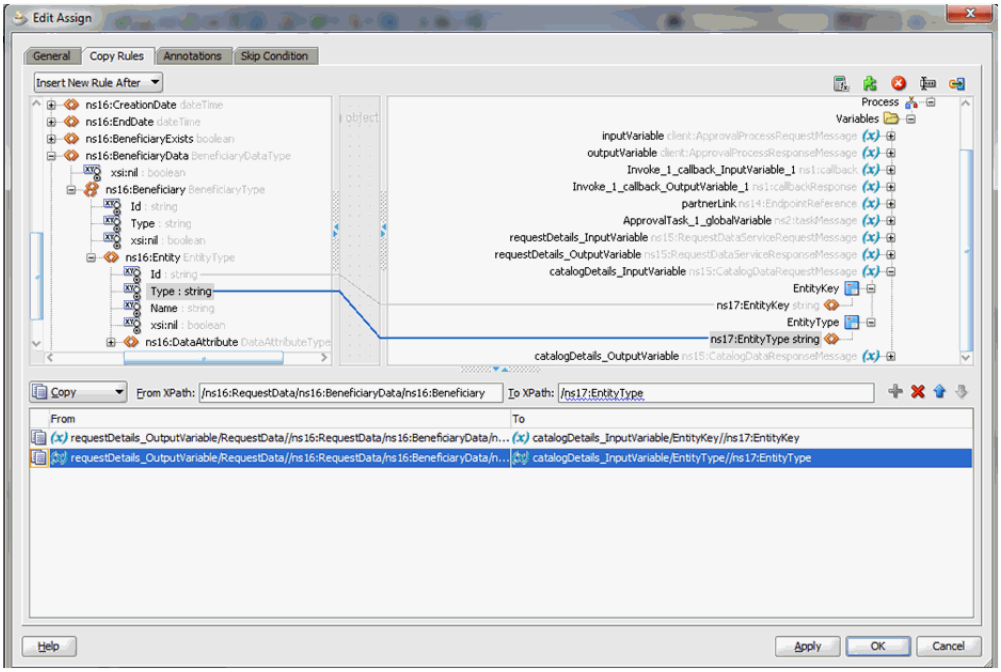


Note: The following attributes will be returned as custom attributes through the catalog detail method of the request web service:

- APPROVER_USER_FIRSTNAME
- APPROVER_USER_LASTNAME
- APPROVER_USER_DISPLAYNAME
- APPROVER_USER_EMAIL
- CERTIFIER_USER_FIRSTNAME
- CERTIFIER_USER_LASTNAME
- CERTIFIER_USER_DISPLAYNAME
- CERTIFIER_USER_EMAIL
- FULFILLMENT_USER_FIRSTNAME
- FULFILLMENT_USER_LASTNAME
- FULFILLMENT_USER_DISPLAYNAME
- FULFILLMENT_USER_EMAIL

11. Right-click and edit the assign activity to map the input to the InvokeCatalogOperation, as shown in Figure 21–11.

Figure 21–11 InvokeCatalogOperation Input Mapping



21.4.5.3 Configuring Workflow Selection

To define the workflow selection rules:

1. Define a variable called catalogData. To do so:

a. Click the **Variables** icon, and then click the **Create** icon on the Variable dialog box.

b. Choose **Type** as Element and click the **Search** icon next to the field.

c. In the dialog box, expand **Project Schema Files** and then *CatalogData.xsd* and select the CatalogData element. This variable will contain the catalog details returned as an output of the InvokeCatalogDetails step.

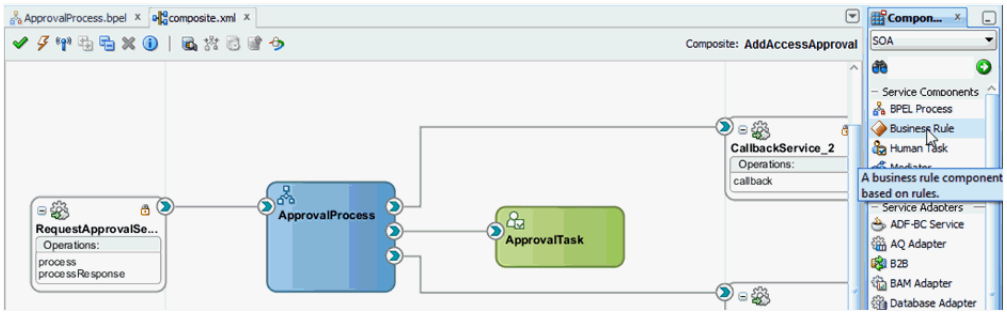
2. Define a variable called workflowtype. To do so:

a. Select type as **Element** and click on **Search** icon next to the field.

b. In the dialog box, expand P**roject Schema Files** and then *BusinessRule.xsd* and select the StageOutput element. This variable will contain the type of workflow to be invoked.

3. Navigate to the SOA Composite view, and add a Business Rule component, as shown in Figure 21–12.

Figure 21–12 Adding Business Rule Component



4. In the Create Business Rules dialog box, specify the name of the Rule Dictionary as WorkflowSelection.

5. Specify Input as CatalogData from CatalogData.xsd in Project Schema Files and Output as StageOutput from BusinessRule.xsd in Project Schema Files

6. Switch to the BPEL process.

7. Expand SOA Components and add a Business Rule component between the InvokeCatalogOperation and ApprovalTask_1 components.

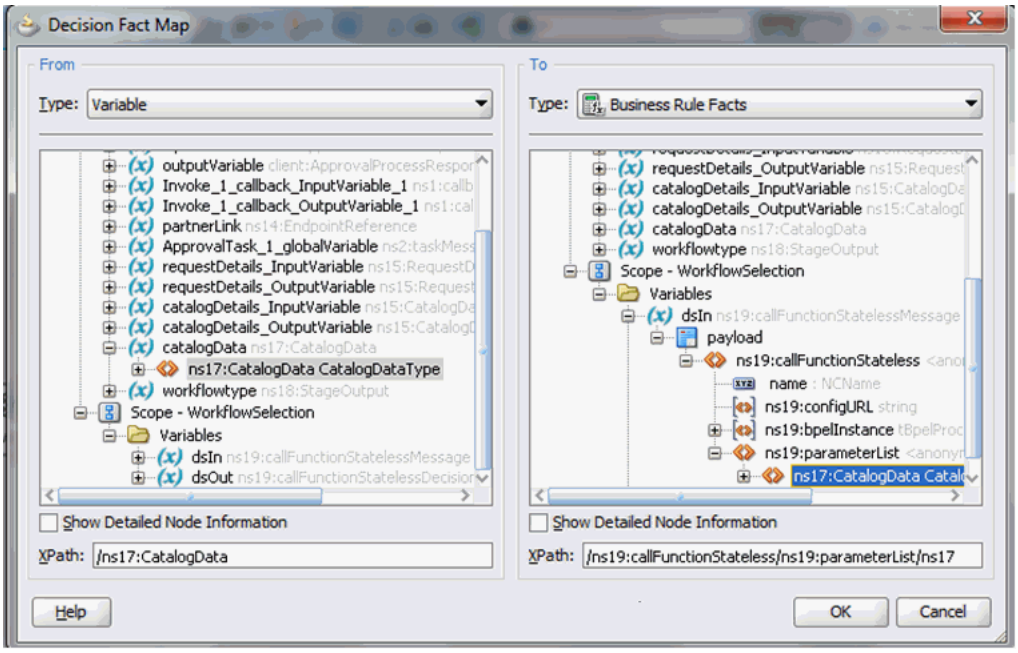
8. Edit the rule and rename it to `WorkflowSelection`.

9. In the Rule dialog box, click the Dictionary tab, and select the WorkflowSelection dictionary that you defined in step 4.

10. Select Assign Input Facts subtab in the Dictionary tab, and click the plus (+) icon.

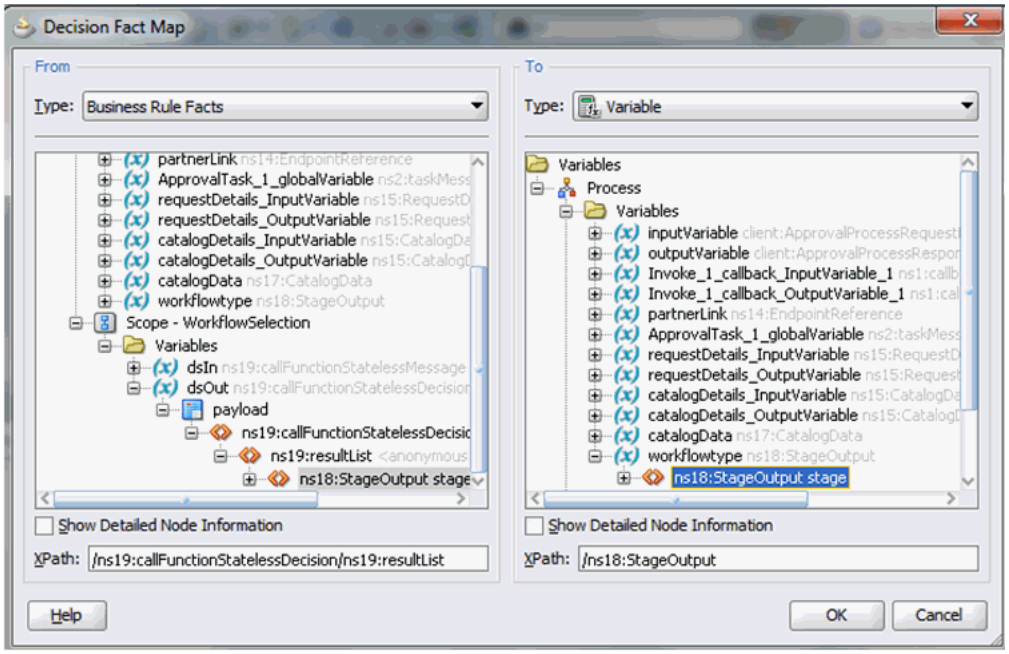
11. Map the catalogData variable to the input to the Rule, as shown in Figure 21–13.

Figure 21–13 catalogData Variable Input Mapping



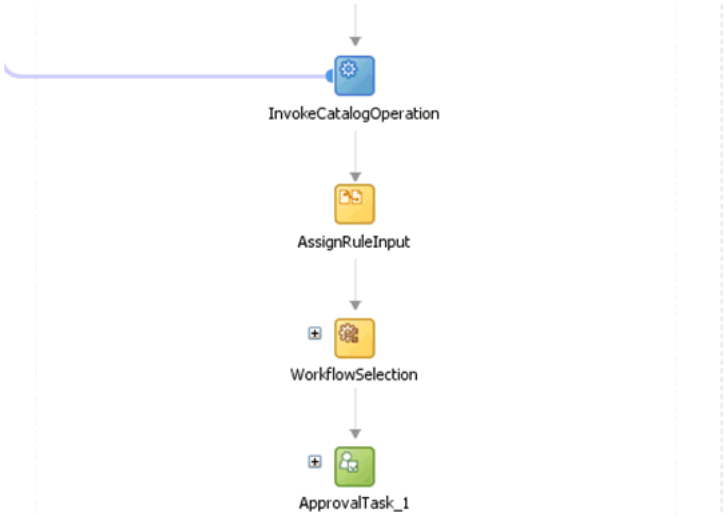
12. Select Assign Output facts subtab in Dictionary tab.
13. Map the workflowtype variable to the output to the Rule, as shown in Figure 21–14.

Figure 21–14 workflowtype Variable Output Mapping



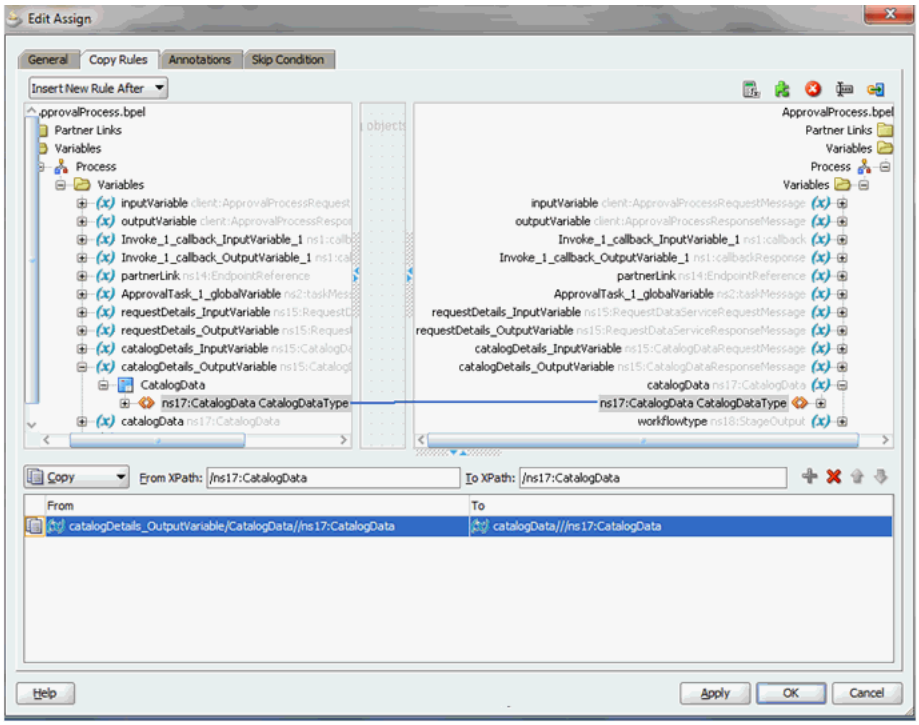
14. Add an Assign activity before the WorkflowSelection rule and rename it as AssignRuleInput, as shown in Figure 21–15.

Figure 21–15 AssignRuleInput



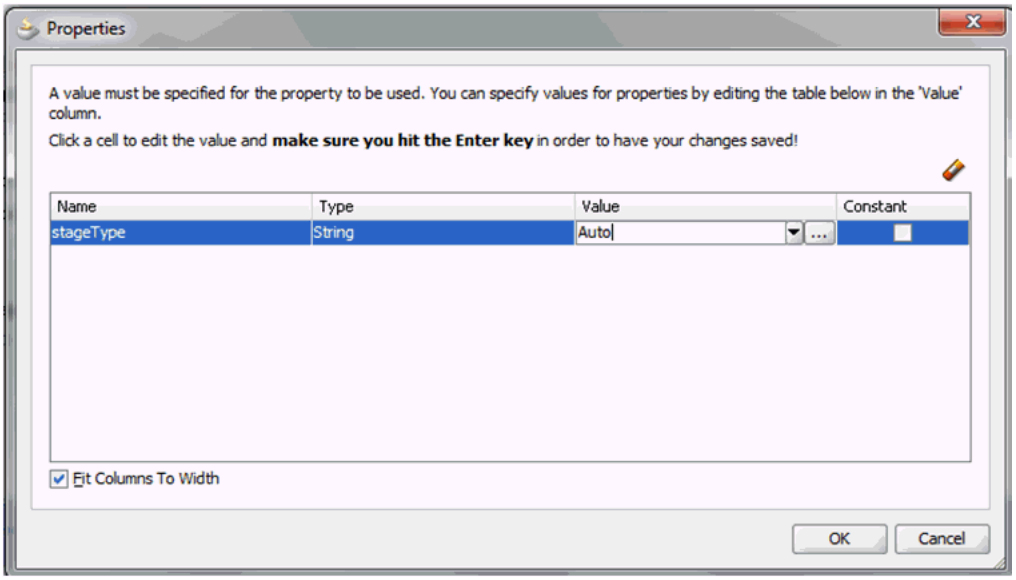
15. Map the output of the InvokeCatalogOperation to the catalogData variable, as shown in Figure 21–16.

Figure 21–16 catalogData Variable Output Mapping



16. Switch to the SOA Composite view.
17. Right-click the Business Rule component, and select **Edit**.
18. Click **Create Rule**.
19. Rename the rule from Rule1 to `Auto Approval`.
20. Edit the rule, and specify the stageType property in the Properties dialog box, as shown in Figure 21–17.

Figure 21–17 The stageType Property



To specify the stageType property:

a. Click the **<insert test>** action below **IF** (click it 3 times to add 3 conditions separated by Logical AND).

Change conditions as:

```
CatalogDataType.itemRisk != 3 and CatalogDataType.itemRisk != 7 and
CatalogDataType.itemRisk != 5
```

b. Click the **<insert action>** below **THEN**.

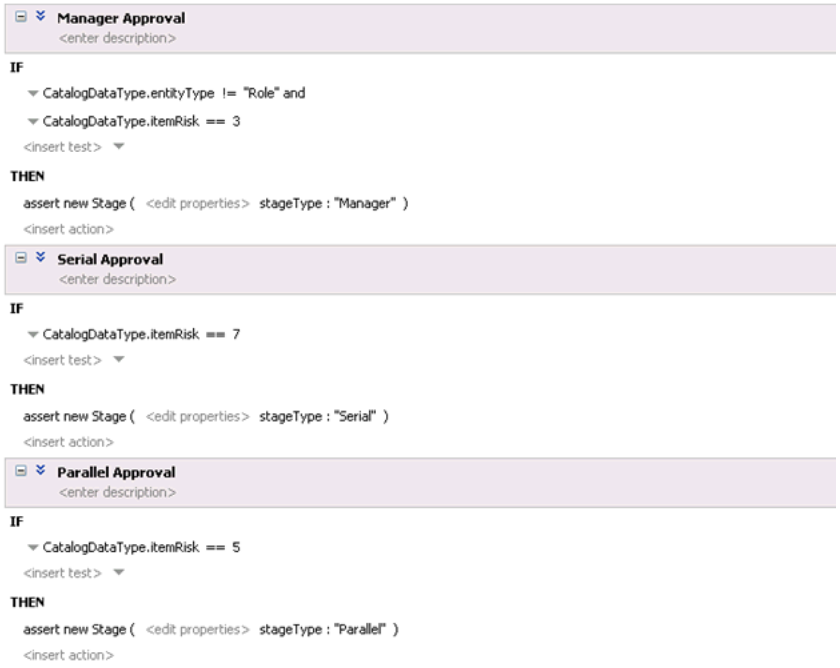
c. Select assert new.

d. Click **<target>** that is added next to assert new, and select **Stage**.

e. Click **<edit properties>**. The Properties dialog box is displayed, as shown in Figure 21–17.

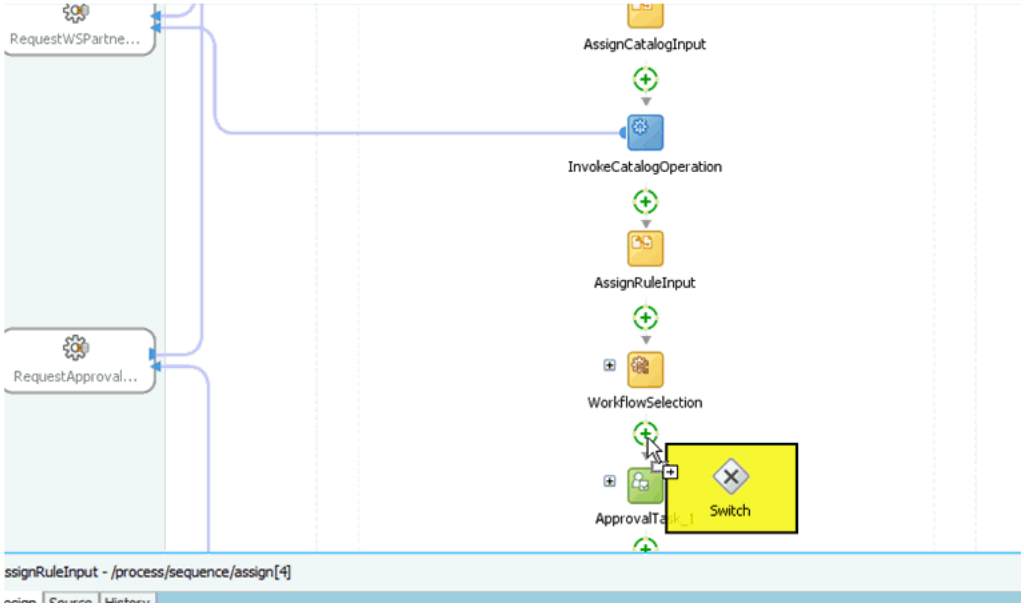
21. Similarly, create the Manager, Serial, and Parallel approval rules, as shown in Figure 21–18.

Figure 21–18 Approval Rules



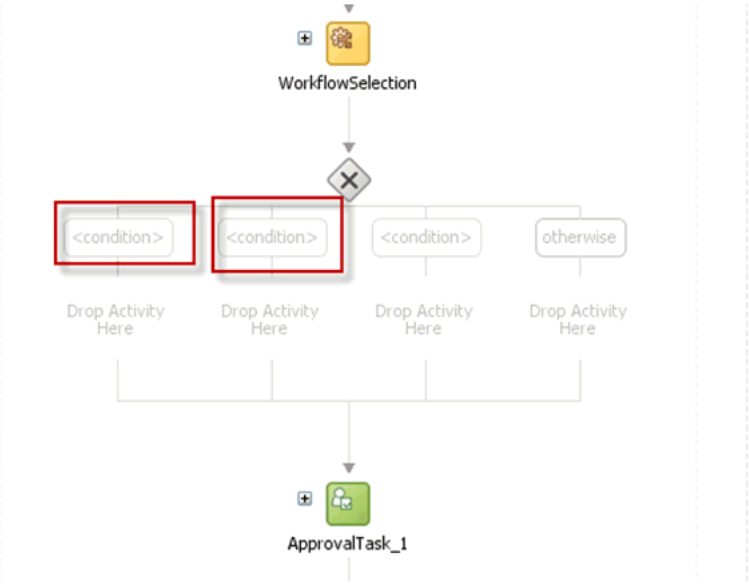
22. Switch to the BPEL process.
23. Add a switch activity after the WorkflowSelection rule, as shown in Figure 21–19.

Figure 21–19 Switch Activity



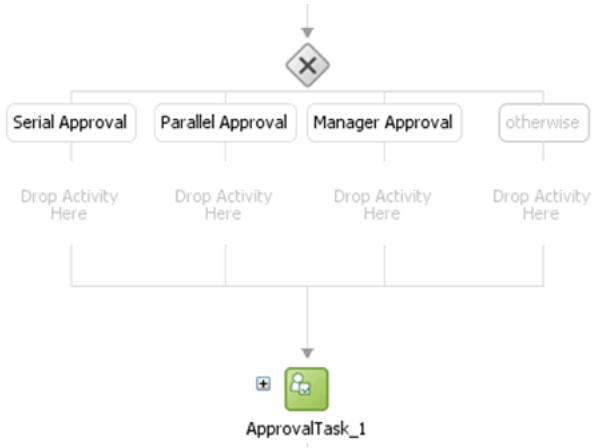
24. Select the Switch activity and add two Switch Case steps, as shown in Figure 21–20.

Figure 21–20 Switch Case Steps



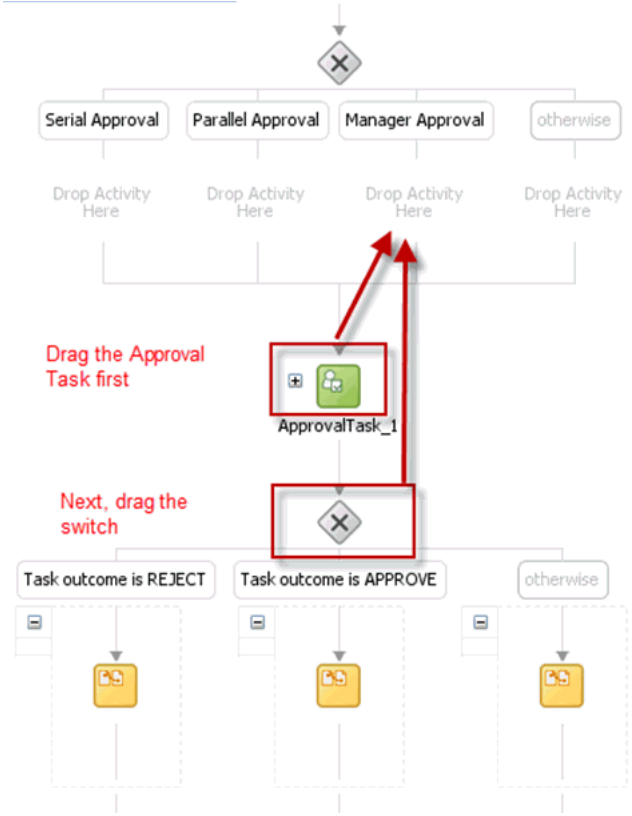
25. Rename the conditions as Serial Approval, Parallel Approval, and Manager Approval, as shown in Figure 21–21.

Figure 21–21 Renamed Conditions



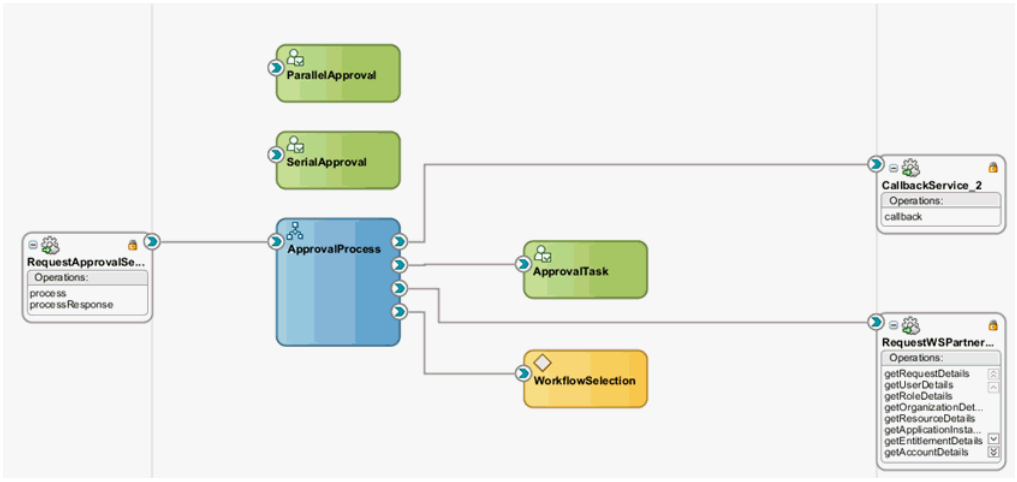
26. Drag the default Human Task into the Manager Switch Case, as shown in Figure 21–22.

Figure 21–22 Dragging Default Human Task



27. Switch to the SOA Composite view.
28. Add two Human Tasks, SerialApproval and ParallelApproval, as shown in Figure 21–23.

Figure 21–23 Adding Human Tasks



29. Switch to the BPEL Process.
30. Edit the Manager Approval Switch case, and add the following expression:

```
bpws:getVariableData('workflowtype','/ns18:StageOutput/ns18:stageType')='Manage
r'
```

You must first configure the newly added Tasks and then wire them to the BPEL Process.

Note: Oracle recommends using expression builder to add the expression..

21.4.5.4 Configuring Human Tasks

Configuring the Human Task consists of the following:

- Configuring the Parallel Human Task
- Configuring the Serial Approval Task
- Configuring the Default Approval Task

21.4.5.4.1 Configuring the Parallel Human Task

To configure the parallel Human Task:

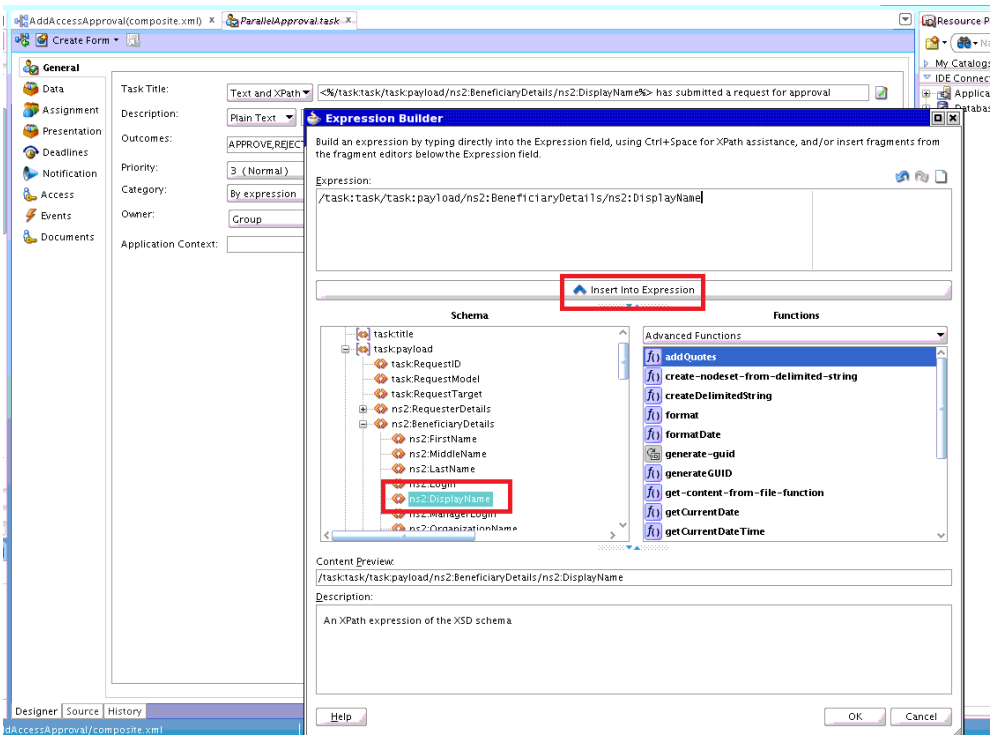
1. Switch to the SOA Composite view.
2. Edit the Parallel Approval Task.
3. Click the **Data** tab, and add the attributes listed in the following table:

Patameter	Data Type
RequestID	{http://www.w3.org/2001/XMLSchema}string
RequestModel	{http://www.w3.org/2001/XMLSchema}string
RequestTarget	{http://www.w3.org/2001/XMLSchema}string
RequesterDetails	{http://xmlns.oracle.com/request/RequestDetails}RequesterDetails
BeneficiaryDetails	{http://xmlns.oracle.com/request/RequestDetails}BeneficiaryDetails
ObjectDetails	{http://xmlns.oracle.com/request/RequestDetails}ObjectDetails
OtherDetails	{http://xmlns.oracle.com/request/RequestDetails}OtherDetails
url	{http://xmlns.oracle.com/request/RequestDetails}url
Catalogdata	{http://xmlns.oracle.com/RequestServiceApp/RequestDataService/CatalogData}CatalogData
RequesterDisplayName	{http://www.w3.org/2001/XMLSchema}string
BeneficiaryDisplayName	{http://www.w3.org/2001/XMLSchema}string
Requester	{http://www.w3.org/2001/XMLSchema}string

4. Verify the task parameters in the Data tab.
5. Click the **General** tab.
6. Set the Task Title to `<%/task:task/task:payload/ns2:BeneficiaryDetails/ns2:DisplayName%> has submitted a request for approval`. To do so:
   a. Click **Edit** next to Task Title, and select **task:payload**, **ns2:BeneficiaryDetails**, **ns2:DisplayName**.

b. Click **Insert Into Expression**. Task Title is displayed as shown in Figure 21–24:

Figure 21–24 The Task Title



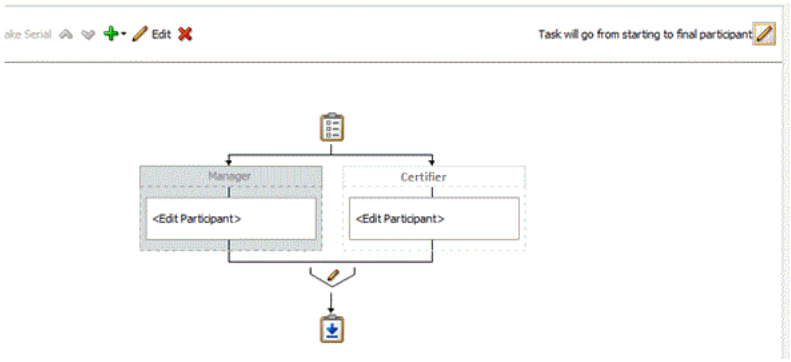

```
<%/task:task/task:payload/ns2:BeneficiaryDetails/ns2:DisplayName%>
```

This can be edited to configure meaningful title, such as:

```
<%/task:task/task:payload/ns2:BeneficiaryDetails/ns2:DisplayName%> has
submitted a request for approval.
```

7. Set the Task Owner to Group and SYSTEM ADMINISTRATORS.
8. Click the **Notification** tab, and then click **Advanced**.
9. Select the **Make notification actionable** option.
10. Click the **Assignment** tab.
11. Add a Parallel stage. To do so, select Stage1 in the flow chart, and click the plus (+) icon, and select **Parallel stage**.
12. Select a stage and click **Edit**. Provide the name as `Manager`.
13. Select the other stage, and provide the name as `Certifier`, as shown in Figure 21–25.

Figure 21–25 Manager and Certifier Stages



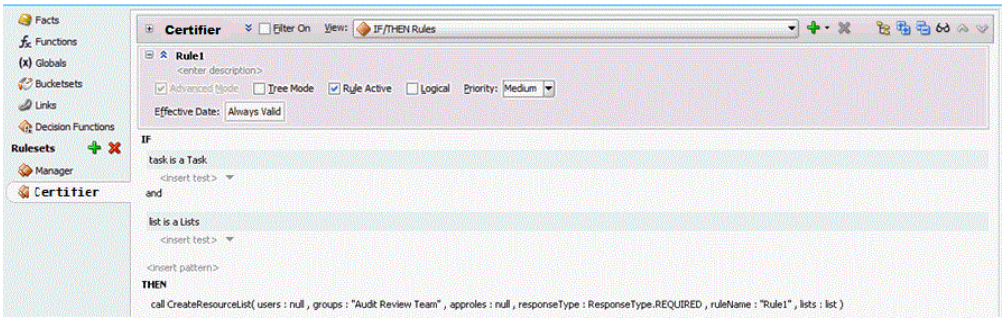
14. Click the pencil icon after the two stages.
15. In the Vote Outcome dialog box, set the voted outcome to APPROVE. Set the default outcome to REJECT. This is because when both the approver reject the task, the request moves to completed state based on the default outcome.
16. Select the **Share attachments and comments** option.
17. In the Vote Outcome dialog box, click **OK**.
18. Ensure that the **Immediately trigger voted outcome when minimum percentage is met** option is selected.
19. Select **<edit participant>** in Manager Stage, and click **Edit**.
20. In the Add Participant Type dialog box, set the Participant type to Single. From the Build a list of participants using list, select **Rule-based** to build the participant list using Rules.
21. In List RuleSet field, enter Manager. Also, edit the label as Manager.
22. In the Add Participant dialog box, click **OK**. The ParallelApprovalRules.rules tab is displayed.
23. Create a rule, as shown in Figure 21–26.

Figure 21–26 Manager Participant Rule

Note: In the Create Rule page, the <insert pattern > in the IF clause is not displayed by default. To display <insert pattern>, expand Rule1 by clicking the two down arrows before Rule1 label, and select the Advanced Mode option as shown in Figure 21–26.

24. Similarly, configure the Certifier stage by specifying the following values in the Add Participant Type dialog box:
    - Type: **Single**
    - Label: **Certifier**
    - Build a list of participants using: **Rule-based**
    - List Ruleset: **Certifier**
    - Select **Let participants manually claim the task**
25. Create a participant rule, as shown in Figure 21–27.

Figure 21–27 Certifier Rule



21.4.5.4.2 Configuring the Serial Approval Task

To configure the serial approval task:

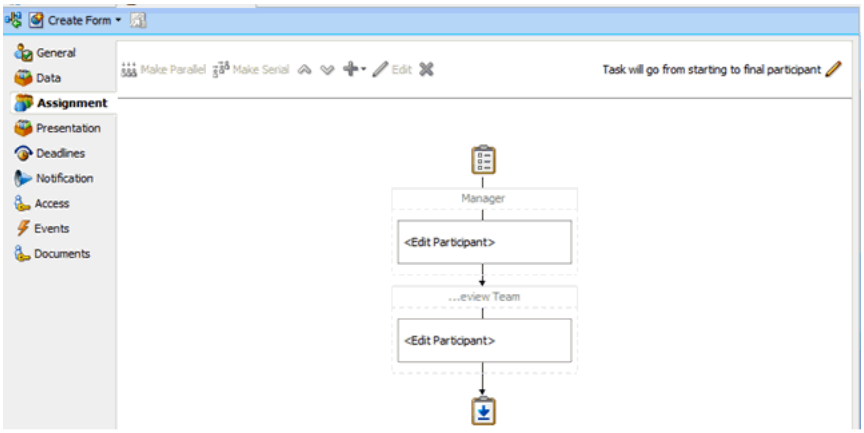
1. Switch to the SOA Composite view.
2. Edit the Serial Approval Task.
3. Click the **Data** tab.
4. Add the parameters listed in the following table:

Parameter	Data Type
RequestID	{http://www.w3.org/2001/XMLSchema}string
RequestModel	{http://www.w3.org/2001/XMLSchema}string
RequestTarget	{http://www.w3.org/2001/XMLSchema}string
RequesterDetails	{http://xmlns.oracle.com/request/RequestDetails}RequesterDetails
BeneficiaryDetails	{http://xmlns.oracle.com/request/RequestDetails}BeneficiaryDetails
ObjectDetails	{http://xmlns.oracle.com/request/RequestDetails}ObjectDetails
OtherDetails	{http://xmlns.oracle.com/request/RequestDetails}OtherDetails
url	{http://xmlns.oracle.com/request/RequestDetails}url
Catalogdata	{http://xmlns.oracle.com/RequestServiceApp/RequestDataService/CatalogData}CatalogData
RequesterDisplayName	{http://www.w3.org/2001/XMLSchema}string

Parameter	Data Type
BeneficiaryDisplayName	{http://www.w3.org/2001/XMLSchema}string
Requester	{http://www.w3.org/2001/XMLSchema}string

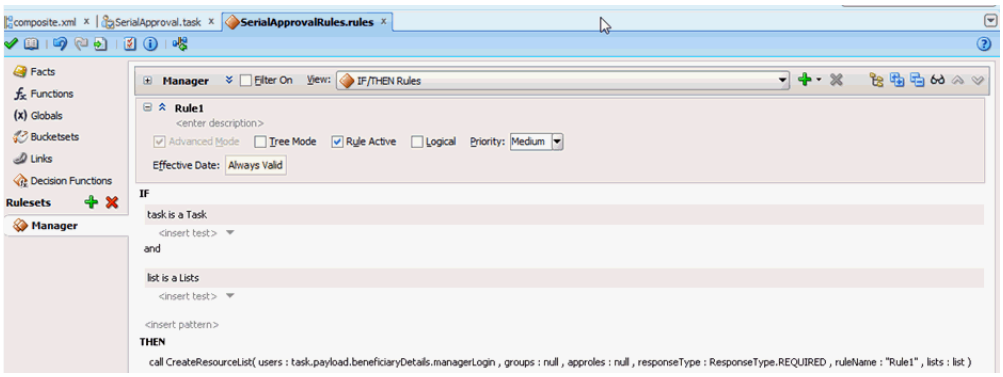
5. Verify the task parameters in the Data tab.
6. Click the **General** tab.
7. Set the Task Title to `<%/task:task/task:payload/ns2:BeneficiaryDetails/ns2:DisplayName%> has submitted a request for approval`.
8. Set the Task Owner to Group and SYSTEM ADMINISTRATORS.
9. Click the **Notification** tab, and then click **Advanced**.
10. Select the **Make notification actionable** option.
11. Click the **Assignment** tab.
12. Add a Sequential stage and rename the stages as Manager and Review Team, as shown in Figure 21–28.

Figure 21–28 Serial Stages



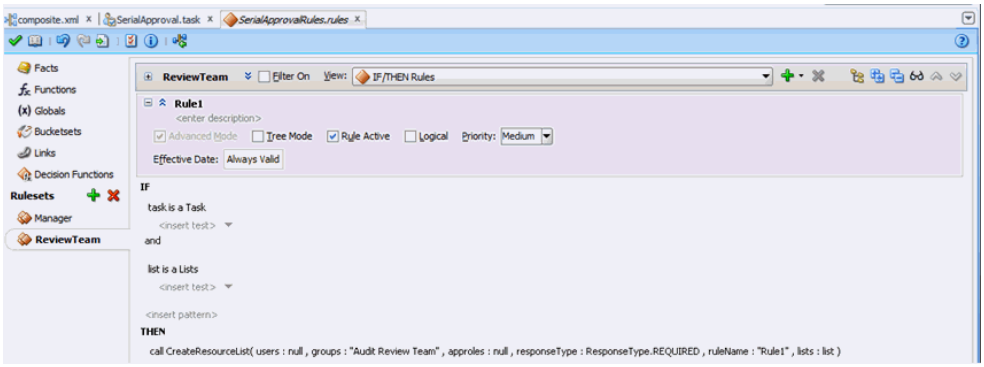
13. Edit the Manager stage.
14. In the Add Participant Type dialog box, set the Participant type to Single and build the participant list using Rules.
15. Create the participant list rule as shown in Figure 21–29.

Figure 21–29 Rule for Manager Stage



16. Similarly, configure the Review Team stage.
17. Create the participant list rule as shown in Figure 21–30.

Figure 21–30 Rule for Review Team Stage

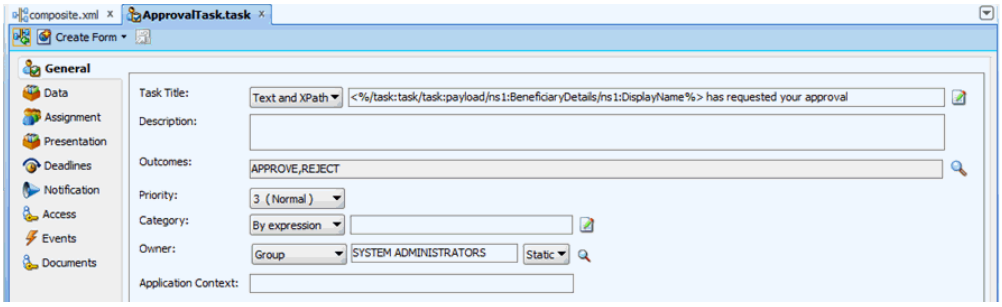


21.4.5.4.3 Configuring the Default Approval Task

To configure the default approval task:

1. Switch to SOA composite view.
2. Edit the Approval Task.
3. Click the **General** tab.
4. Set the task title, as shown in Figure 21–31.

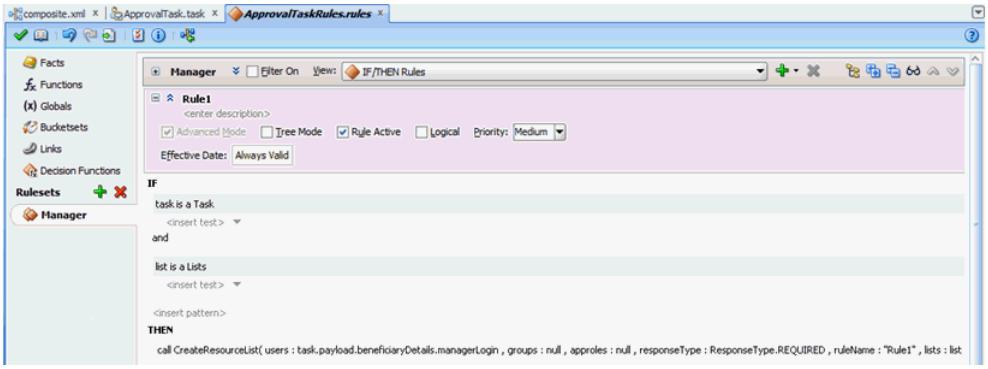
Figure 21–31 Default Approval Task



5. Click **Assignment**.

6. Select Stage1.Participant1, and click **Edit**.
7. In the Add Participant Type dialog box, set the Participant type to Single. From the Build a list of participants using list, select **Rule-based** to build the participant list using Rules.
8. Enter Manager in Label and RuleSet, and click **OK**, which opens the rules.
9. Create a Participant list rule, as shown in Figure 21–32.

Figure 21–32 Participant List Rule



21.4.5.5 Configuring the Human Task and BPEL Mappings

Configuring the Human Task and BPEL mappings involves:

- Configuring the Serial Approval Human Task
- Configuring the Parallel Human Task
- Configuring Auto Approval

21.4.5.5.1 Configuring the Serial Approval Human Task

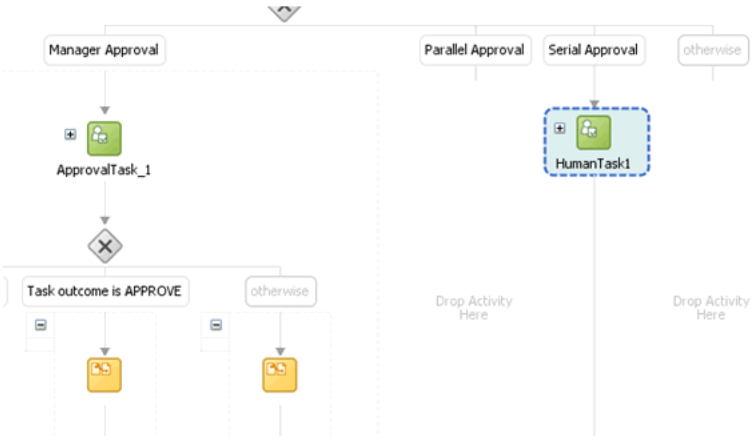
To configure the serial approval Human Task:

1. Switch to BPEL process.
2. Add the following condition to the Serial Approval switch:

```
bpws:getVariableData('workflowtype','/ns18:StageOutput/ns18:stageType') =
'Serial'
```

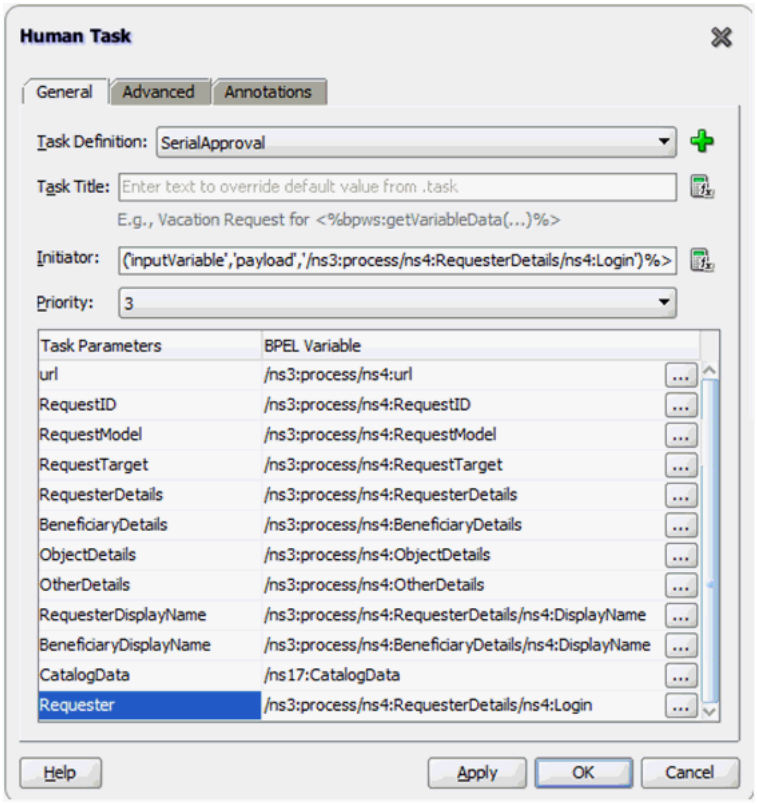
3. Drag and drop a Human Task activity from the SOA Components into the Serial Approval switch, as shown in Figure 21–33.

Figure 21–33 Human Task Activity



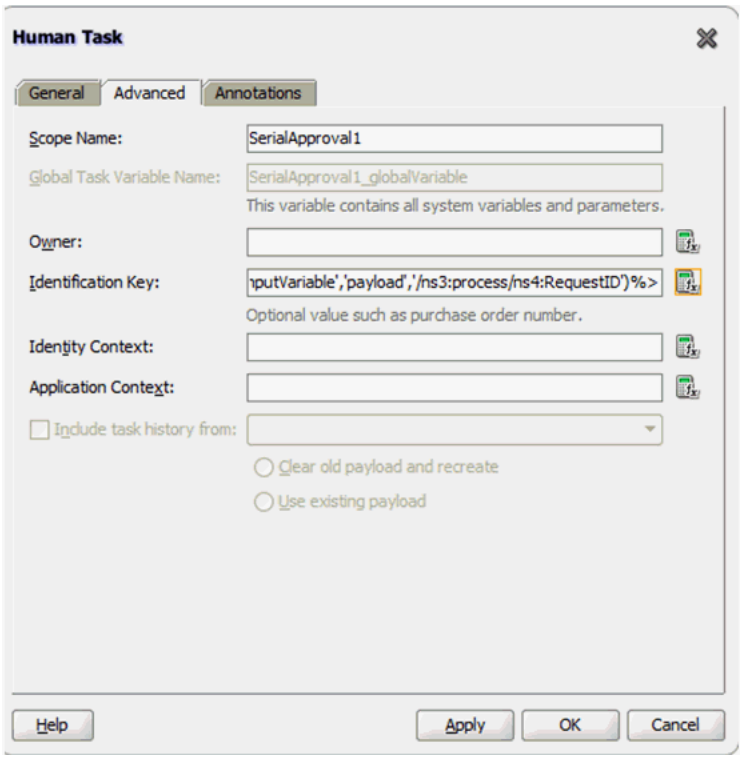
4. Edit the Human Task, and in the Human Task dialog box, select the Serial Approval Human Task definition.
5. Map Initiator to requester login, and map the task parameters to the BPEL variable as shown in Figure 21–34.

Figure 21–34 Task Parameters and BPEL Variable Mapping



6. Click the **Advanced** tab.
7. Map the Identification Key to the Request ID as shown in Figure 21–35.

Figure 21–35 Identification Key and Requester ID Mapping



8. Map Iniatator to requester login, and then click **Apply** and **OK**.
9. Select the Switch case for Task outcome is REJECT.
10. Replace the existing condition script with the following:

```
bpws:getVariableData('SerialApproval1_
globalVariable','payload','/task:task/task:systemAttributes/task:outcome') =
'REJECT'
```

11. Select and edit the Assign activity under Task outcome is REJECT.
12. Delete all the copy rules except one. The copy rule that you retain can be any one so that you can replace it in the Source view.
13. Save and click the **Source** tab.
14. Select the copy activity.
15. Replace the activity with the following:

```
<sequence>
   <assign>
      <copy>
                <from expression="string('rejected')"/>
                <to variable="outputVariable"
                    part="payload"
                    query="/ns3:processResponse/ns3:result"/>
      </copy>
      <copy>
                <from expression="ora:getConversationId()"/>
                <to variable="Invoke_1_callback_InputVariable_1"
                    part="parameters"
                    query="/ns1:callback/arg0"/>
      </copy>
```

```
      <copy>
               <from expression="string('rejected')"/>
               <to variable="Invoke_1_callback_InputVariable_1"
                   part="parameters"
                   query="/ns1:callback/arg1"/>
      </copy>
   </assign>
</sequence>
```

16. Repeat the steps for the Task outcome is APPROVE. Select the Switch Case and copy the following in the Condition field:

```
bpws:getVariableData('SerialApproval1_
globalVariable','payload','/task:task/task:systemAttributes/task:outcome') =
'APPROVE'
```

17. Select the Assign activity under the Approve outcome, and replace the copy rules with the following:

```
<sequence>
          <assign>
            <copy>
                  <from expression="string('approved')"/>
                  <to variable="outputVariable"
                      part="payload"
                      query="/ns3:processResponse/ns3:result"/>
            </copy>
            <copy>
                <from expression="ora:getConversationId()"/>
                <to variable="Invoke_1_callback_InputVariable_1"
                    part="parameters"
                    query="/ns1:callback/arg0"/>
            </copy>
            <copy>
               <from expression="string('approved')"/>
               <to variable="Invoke_1_callback_InputVariable_1"
                   part="parameters"
                   query="/ns1:callback/arg1"/>
            </copy>
          </assign>
</sequence>
```

18. Select the Assign activity under the Otherwise outcome, and replace the copy rules with the following:

```
<sequence>
    <assign>
        <copy>
             <from expression="bpws:getVariableData('SerialApproval1_
globalVariable','payload','/task:task/task:systemAttributes/task:state')"/>
             <to variable="outputVariable" part="payload"
                        query="/ns3:processResponse/ns3:result"/>
         </copy>
         <copy>
            <from expression="ora:getConversationId()"/>
            <to variable="Invoke_1_callback_InputVariable_1"
                part="parameters"
                query="/ns1:callback/arg0"/>
         </copy>
         <copy>
            <from expression="bpws:getVariableData('SerialApproval1_
```

```
globalVariable','payload','/task:task/task:systemAttributes/task:state')"/>
            <to variable="Invoke_1_callback_InputVariable_1"
                part="parameters"
                query="/ns1:callback/arg1"/>
          </copy>
          </assign>
</sequence>
```

21.4.5.5.2 Configuring the Parallel Human Task

To configure the parallel Human Task:

1. Add the following condition to the Parallel Approval switch activity:

   ```
   bpws:getVariableData('workflowtype','/ns18:StageOutput/ns18:stageType') =
   'Parallel'
   ```

2. Drag and drop a Human Task activity from the SOA Components into the Parallel Approval switch.
3. Select the Parallel Approval Human Task.
4. Map the Human Task parameters in the same way as the Serial Human Task.
5. Map the Assign activity for the APPROVE outcome in the same way as the equivalent in the Serial Human Task.
6. Map the Assign activity for the REJECT outcome in the same way as the equivalent in the Serial Human Task.
7. Map the Assign activity for the Otherwise outcome in the same way as the equivalent in the Serial Human Task.

> **Note:** You must specify appropriate global variable (ParallelApproval1_globalVariable) in the copy activity.

21.4.5.5.3 Configuring Auto Approval

To configure auto approval:

1. Drag and drop an Assign activity in the Otherwise switch case.
2. Select the Assign activity, and switch to Source view.
3. In the Assign activity, replace the following:

   ```
   <assign name="Assign1"/>
   ```

   With:

   ```
   <sequence>
            <assign>
              <copy>
                    <from expression="string('approved')"/>
                    <to variable="outputVariable"
                        part="payload"
                        query="/ns3:processResponse/ns3:result"/>
              </copy>
              <copy>
                  <from expression="ora:getConversationId()"/>
                  <to variable="Invoke_1_callback_InputVariable_1"
                      part="parameters"
   ```

```
                    query="/ns1:callback/arg0"/>
                </copy>
                <copy>
                   <from expression="string('approved')"/>
                   <to variable="Invoke_1_callback_InputVariable_1"
                       part="parameters"
                       query="/ns1:callback/arg1"/>
                </copy>
            </assign>
</sequence>
```

21.4.5.6 Deploying the SOA Composite

To deploy the SOA composite:

1. Select **File**, **Save All** to save your work.
2. Right-click the project, and select **Deploy**, *COMPOSITE_NAME*, **Deploy to Application server**. Alternatively, you can deploy to SAR (SOA Archive), and then deploy it by using Oracle Enterprise Manager.

Note:

- The default version is 1.0. You can also change the version, if you have existing composite instances running.
- If you are redeploying the composite and you have added or removed one or more human tasks, then it is recommended to deploy with a different version.

21.4.5.7 Creating the Approval Policies

The SOA composite that you have created can be used for all requests other than user operations, such as self-registration, create, modify, disable, enable, and delete. To ensure that all requests other than user operations invoke this composite, you must create an approval policy in Oracle Identity Manager. To create the approval policies in Oracle Identity Manager:

1. Login to Oracle Identity System Administration.
2. Under Policies, select **Approval Policies**.
3. Create a request-level approval policy for Provision Application Instance, and set it to auto approve.
4. Create an operational-level approval policy for Provision Application Instance, and click **Approval Process**.
5. Select the SOA composite that you deployed.
6. Specify scope type to be All.
7. Create a default approval rule:

   Request.Request Type = Provision Application Instance
8. Save the approval policy.

Tip: You can access custom attribute's value of entities, such as catalog, user, role, or organization, supported by the request web service is SOA composite BPEL process. The custom attributes or UDFs are part of the CustomAttribute element. An instance of catalog entity containing UDF is:

```
<ns12:CustomAttribute Name="ApproverRolePhoneNumber">
    <ns5:Value>1234</ns5:Value>
</ns12:CustomAttribute>
<ns12:CustomAttribute Name="ApproverRoleEmailId">
   <ns5:Value>approver@mydomain.com</ns5:Value>
</ns12:CustomAttribute>
```

For example, to access the ApproverRolePhoneNumber catalog UDF value in BPEL process, specify the following:

```
bpws:getVariableData('catalogDetails','CatalogData','/ns22:CatalogD
ata/ns22:CustomAttribute[@Name =
string("ApproverRolePhoneNumber")]/ns24:Value')
```

21.5 Configuring Default Request-Level and Operation-Level Approval Composites

You can configure the default request-level and operation-level composites by setting the DefaultRequestLevelComposite and DefaultOperationLevelComposite properties in the oim-config.xml file. You can edit these properties by using System MBean Browser in Oracle Enterprise Manager. The default values for these properties are `default/DefaultRequestApproval!3.0` and `default/DefaultOperationalApproval!3.0` respectively.

The values for these properties are in the following format:

NAMESPACE`/`*COMPOSITE_NAME*`!`*VERSION*

For example:

```
default/AddAccessApproval!2.0
```

If you change the default values of the DefaultRequestLevelComposite and DefaultOperationLevelComposite properties, then you must restart Oracle Identity Manager.

Note: The composites configured as default request-level and operation-level composites would be applicable for all request types. Therefore, these composites must be designed to work with all the request types.

21.6 Creating and Deploying Custom Task Details Taskflow

By default, all tasks are configured to use the default task details page in pending approvals. This taskflow is not customizable. However, you might want to customize the UI or show some other information in the task details page. This section describes how to build your own taskflow, and configure the human task in the DefaultRequestApproval composite to invoke your custom taskflow.

This section contains the following topics:

21.6.1 Prerequisites for Developing Custom Task Details Taskflow

Before developing a custom task details taskflow, you must have the following software installed on your computer:

- Oracle Identity Manager 11*g* Release 2 (11.1.2.2.0)
- Oracle SOA 11*g* (11.1.1.7.0)
- JDeveloper 11*g* (11.1.1.7.0) with Oracle SOA Composite Editor extension

21.6.2 Developing Custom Task Details Taskflow

To build a custom taskflow for the human task in the DefaultRequestApproval composite:

1. Open Jdeveloper and create a new Generic Application. To do so:
   a. Enter Application Name as `RequestApprovalTaskDetailsApp`, and then click **Next**.
   b. Enter Project Name as `RequestApprovalTaskDetails`. Do not select any project technologies.
   c. Click **Finish**.
2. Add Oracle Identity Manager shared library. To do so:
   a. Right-click **RequestApprovalTaskDetails** project, and select **Project Properties**, **Libraries and Classpath**.
   b. Click **Add Library**.
   c. Click **Load Dir**.
   d. Navigate to the *IAM_HOME*/server/jdev.lib/ directory, and click **Select**.

   Note: *IAM_HOME* is the path to the Oracle Identity Manager home directory, for example, BEA_HOME/Oracle_IDM1/. Here, *BEA_HOME* is the path to the middleware directory in Oracle Identity Manager installation.

   e. Select **OIM View Shared library**, **OIM Model Shared library**, and then click **OK**.
   f. Click **OK**.
3. Create task details taskflow. To do so:
   a. Navigate to the following directory in shiphome:

      IAM_HOME/server/workflows/composites/

b. Unzip the DefaultRequestApproval.zip file.

c. Go back to Jdeveloper, right-click **RequestApprovalTaskDetails**, and select **New**.

d. Select **Web Tier**, **JSF**, ADF task flow based on human task.

e. In the file browser, navigate to the directory in which you unzipped DefaultRequestApproval.zip. Select the DefaultRequestApproval/ApprovalTask.task file

f. In the Create Task flow dialog box, provide the following values:

File Name: request-approval-details-tf.xml

Task Flow ID: request-approval-details-tf

g. Click **OK**.

4. Delete hwtaskflow.xml. To do so, go to Application Sources under RequestApprovalTaskDetails project, and then delete hwtaskflow.xml.

For more information on workflow for multiple human tasks, see "How To Reuse the Task Flow Application with Multiple Human Tasks" in *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite Guide*.

5. Create the task details page. To do so:

a. Open request-approval-details-tf.xml. Switch to diagram mode.

b. Rename **taskdetails1_jspx** view activity to **request-approval-details**.

c. Right-click the **request-approval-details** view activity, and select **Create Page**. Provide the following values:

File name: request-approval-details.jspx

Initial Page layout and content: Blank Page

d. Click **OK**.

6. Add managed bean for this page. To do so:

a. Right-click the **RequestApprovalTaskDetails** project, and select **New**, **Java Class**. Provide the following values:

Name: RequestApprovalDetailsStateBean

Package: oracle.iam.ui.custom.view.backing

b. Click **OK**.

c. Add the following code to the managed bean:

```
package oracle.iam.ui.custom.view.backing;

import javax.el.ELContext;
import javax.el.ExpressionFactory;
import javax.el.ValueExpression;

import javax.faces.application.Application;
import javax.faces.context.FacesContext;

import oracle.iam.ui.platform.model.config.ConstantsDefinition;


public class RequestApprovalDetailsStateBean implements
java.io.Serializable{
```

```
    public RequestApprovalDetailsStateBean() {
        super();
    }

    private String requestAction = ConstantsDefinition.REQUEST_ACTION_
APPROVAL_UPDATE;
    private String requestType = ConstantsDefinition.REQUEST_TYPE_VIEW_
DETAIL;

    public void setRequestAction(String requestAction) {
        this.requestAction = requestAction;
    }

    public String getRequestAction() {
        return requestAction;
    }

    public void setRequestType(String requestType) {
        this.requestType = requestType;
    }

    public String getRequestType() {
        return requestType;
    }

    public String getUserIds() {
        Object benefDisplayName =
getValueFromELExpression("#{bindings.DisplayName.inputValue}");
        //benefDisplayName would be "None" if beneficiary does not exist
        if (benefDisplayName != null &&
!ConstantsDefinition.NONE_BENEF_DISPLAY_
NAME.equalsIgnoreCase(benefDisplayName.toString()))
            return benefDisplayName.toString();

        Object requestTarget =
getValueFromELExpression("#{bindings.RequestTarget.inputValue}");
        if (requestTarget != null)
            return requestTarget.toString();

        return "";
    }

    private Object getValueFromELExpression(String expression) {
        FacesContext facesContext = FacesContext.getCurrentInstance();
        Application app = facesContext.getApplication();
        ExpressionFactory elFactory = app.getExpressionFactory();
        ELContext elContext = facesContext.getELContext();
        ValueExpression valueExp =
            elFactory.createValueExpression(elContext, expression,
                                            Object.class);
        return valueExp.getValue(elContext);
    }


}
```

d. Open request-approval-details-tf.xml in Overview mode. Select Managed Beans sections and register the managed bean with the following details:

Name: requestApprovalDetailsStateBean

Class: oracle.iam.ui.custom.view.backing.RequestApprovalDetailsStateBean

Scope: pageFlow

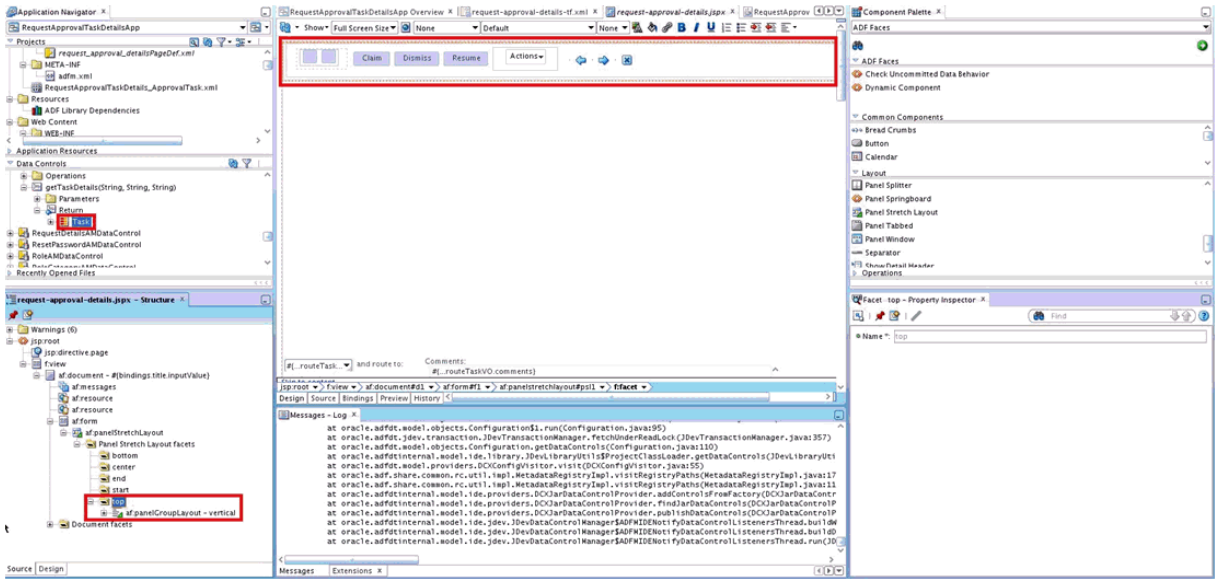
7. Create the details page structure. To do so:

   a. Open request-approval-details.jspx.

   b. From the Component Palette, add a **panelStretchLayout** to the page. In the Property Inspector, set `TopHeght==auto` for panelStretchLayout.

   c. Go to Data controls in Application Navigator. Expand **RequestApprovalTaskDetails_ApprovalTask**, **getTaskDetails**, **Return**. Drag and drop **Task** from Data Controls on to the Top Facet of panelStretchLayout, as shown in Figure 21–36. From the context menu, select **Human Task**, **Task Action**. The Human task actions are added to the Top Facet.

Figure 21–36 Dragging Task to the Top Facet



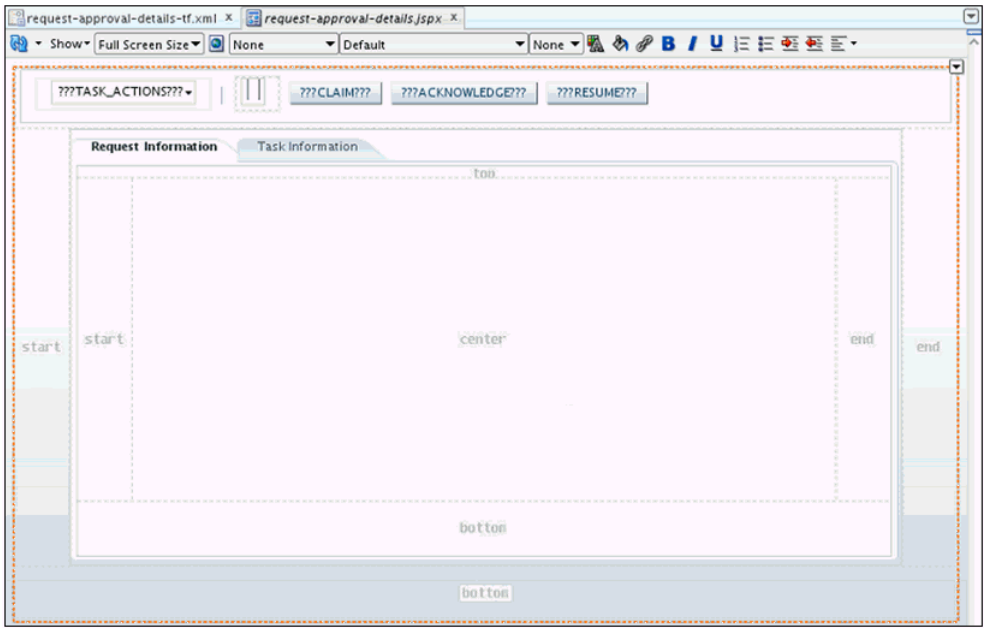
   d. From the Component Palette, add a **panelTabbed** layout to the Center Facet of panelStretchLayout.

   e. From the Component Palette, add two **showdetailItem** components to the panelTabbed layout. From property inspector, set the text name for these components as `Request Information` and `Task Information`.

   f. Click the **Request Information** tab. From the property inspector, set attribute `stretchChildren=first`.

   g. Add another **panelStretchLayout** in Request Information tab. Set attribute `topHeight=auto` for this panelStretchLayout. Figure 21–37 shows the Request Information and Task Information tabs.

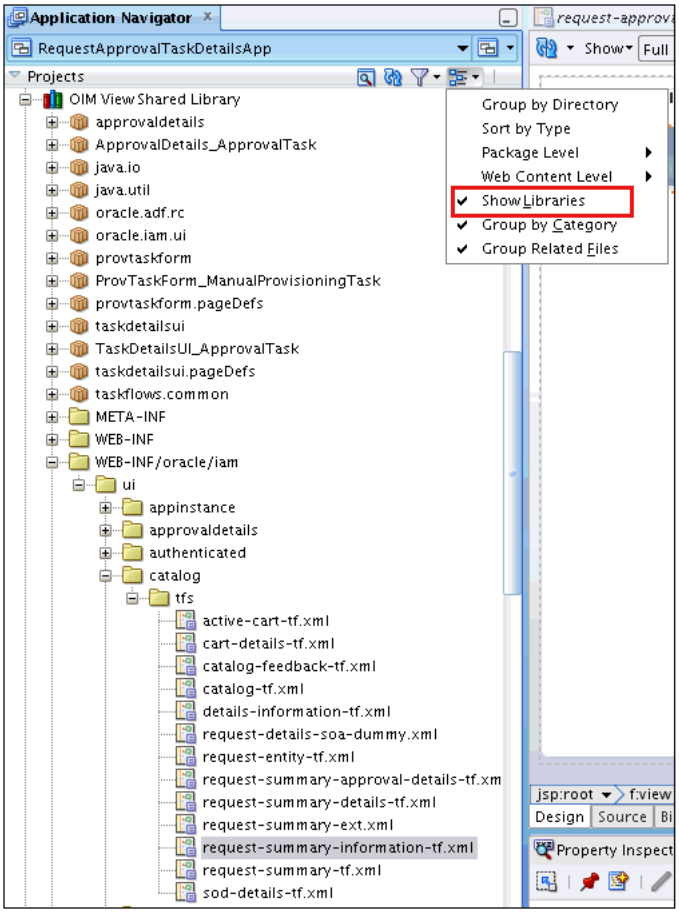
Figure 21–37 The panelTabbed Layout



8. Populate the **Request Information** tab. To do so:

   a. Go to Navigator Display Options, and select **Show Libraries**, shown in Figure 21–38. This will show **OIM View Shared Library** in the Application Navigator.

Figure 21–38 OIM View Shared Library



b. In the Application Navigator, expand **OIM View Shared Library**, **WEB-INF/oracle/iam/ui/catalog/tfs**. Drag and drop **request-summary-information-tf.xml** to the Top Facet of PanelStretchLayout added in step 7g. The Create context menu is displayed. Select **Region**.The Edit Task Flow Binding dialog box is displayed. You can provide parameters to the taskflow later. Therefore, click **OK**.

c. Similarly, drag and drop **catalog-tf.xml** to the Center Facet of PanelStretchLayout added in step 7g. The Create context menu is displayed. Select **Region**. The Edit Task Flow Binding dialog box is displayed. Click **OK**.

d. Click the **Bindings** tab at the bottom of the page to view the bindings. Click the plus (+) sign to add a binding in the following way:

i) Enter the following and click **OK**:

Category: Generic Bindings

Item to be created: attributeValues

ii) Click **Add Datasource**. Select **RequestApprovalTaskDetails_ ApprovalTask**, **getTaskDetails**, **Return**, **Task**, **Payload**. Click **OK**.

iii) Specify Attribute as `RequestID`, and click **OK**.

e. Under executables, select **taskflow-requestsummaryinformationtf1**. In the Property Inspector, add a taskflow parameter by clicking the plus (+) sign.

Edit the value field, and update it with #{bindings.RequestID.inputValue}, as shown:

ID=requestID, Value= #{bindings.RequestID.inputValue}

f. Click the plus (+) sign to add another binding. This binding will be referenced in RequestApprovalDetailsStateBean.

i) Enter the following and click **OK**:

Category: Generic Bindings

Item to be created: attributeValues

ii) Click **Add Datasource**. Select **RequestApprovalTaskDetails_ApprovalTask**, **getTaskDetails**, **Return**, **Task**, **Payload**, **BeneficiaryDetails**. Click **OK**.

iii) Specify Attribute as `DisplayName`, and click **OK**.

g. Click the plus (+) sign to add another binding. This binding will be referenced in RequestApprovalDetailsStateBean.

i) Enter the following and click **OK**:

Category: Generic Bindings

Item to be created: attributeValues

ii) From the list, select datasource **RequestApprovalTaskDetails_ApprovalTask**, **getTaskDetails**, **Return**, **Task**, **Payload**.

iii) Specifiy Attribute as `RequestTarget`, and click **OK**.

h. Select **taskflow-catalogtf1** in Executables, click **Edit** on the top-right corner of Executables, and edit the values of the following in the Edit Task flow Binding dialog box:

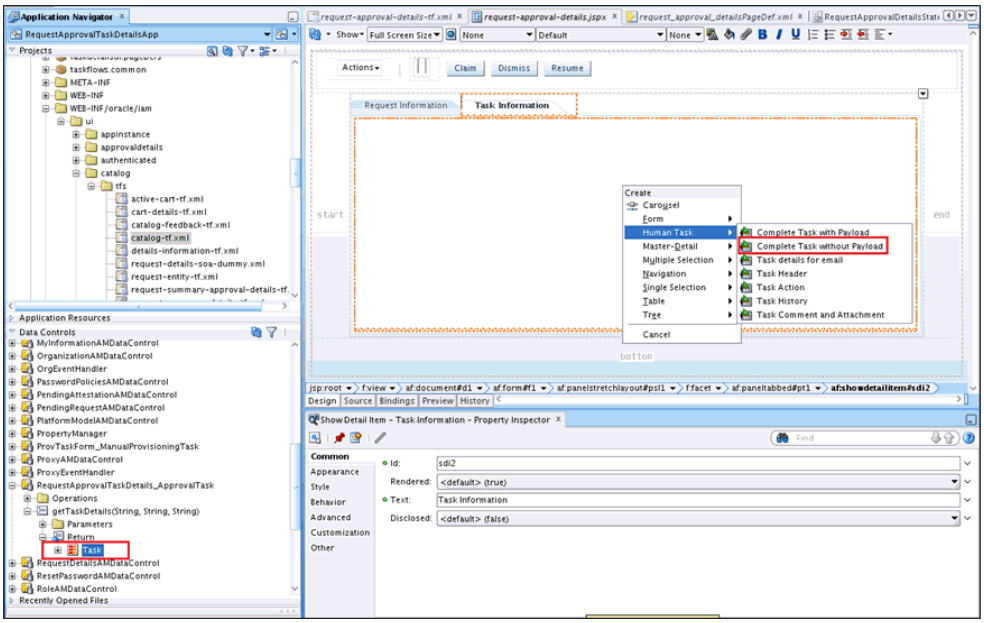
- Id=requestId, Value= #{bindings.RequestID.inputValue}
- Id=requestType, Value=#{pageFlowScope.requestApprovalDetailsStateBean.requestType}
- Id=requestAction, Value=#{pageFlowScope.requestApprovalDetailsStateBean.requestAction}
- Id=userIds, Value=#{pageFlowScope.requestApprovalDetailsStateBean.userIds}

9. Populate the **Task Information** tab. To do so:

a. Switch to Design mode. Click the **Task Information** tab.

b. In the Application Navigator, go to Data Controls. Expand **RequestApprovalTaskDetails_ApprovalTask**, **getTaskDetails**, **Return**. Drag and drop **Task** in the **Task Information** tab. The Create context menu is displayed. Select **Human Task**, **Complete Task without Payload**, as shown in Figure 21–39.

Figure 21–39 Task Details DataControl



c. A **panelHeader** wrapped inside **panelGroupLayout** is added to the **Task Information** tab. Navigate to the **panelHeader** and delete the **Toolbar Facet** of the **panelHeader**. The task actions have already been in step 7c). In addition, task details, task history, comments, and attachments are also added to the Task Information tab.

d. Save your work.

21.6.3 Developing Custom Task Details for Email Notification (Optional)

By default, for sending email notification, if there is no separate page for email, then the same task details page developed in "Developing Custom Task Details for Email Notification (Optional)" on page 21-47 is sent in email notification.

Sometimes, limited information needs to be sent in email notification. In such scenarios, separate page for email notification can be developed. The email page will also be part of the same task details taskflow.

For more information on building custom taskflow for email, refer to "Creating an Email Notification" in the *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

21.6.4 Deploying the Task Details Taskflow

To deploy the task details taskflow:

1. Deploy the Task Details as an ADF library jar. To do so:

   a. Right-click **RequestApprovalTaskDetails**, **Project Properties**, **Deployment**.

   b. Click **New**. The Create deployment profile dialog box is displayed.

   c. Provide following values, and click **OK**.

   Archive Type: ADF Library Jar File

   Name: adflibRequestApprovalTaskDetails

d. Right-click **RequestApprovalTaskDetails**, and select **Deploy**, **adflibRequestApprovalTaskDetails**.

e. In the deployment action popup, click **Finish**.

2. Package the adflibRequestApprovalTaskDetails.jar in custom shared library. To do so:

a. Navigate to the *IAM_HOME*/server/apps/ directory.

b. Create following directory structure:

WEB-INF/lib/

c. Copy adflibRequestApprovalTaskDetails.jar to the WEB-INF/lib/ directory.

d. Update *IAM_HOME*/server/apps/ oracle.iam.ui.custom-dev-starter-pack.war to add adflibRequestApprovalTaskDetails.jar. For example:

```
jar uvf oracle.iam.ui.custom-dev-starter-pack.war WEB-INF/*
```

3. Restart Oracle Identity Manager managed server for the changes to custom shared library to take effect.

21.6.5 Configuring Human Task and Taskflow Permissions

To configure Human Task and taskflow permissions:

1. Add view permission for custom taskflow by using Authorization Policy Manager (APM). To do so:

a. Login to APM application as WebLogic user.

b. Navigate to **Applications**, **OracleIdentityManager**, **Resource Types**. Click **Open**.

c. Click New to create a new resource type. Provide following details:

- **Display Name:** ADF Taskflows
- **Name:** ADFTaskFlows
- **Actions:** personalize, customize, grant, view. Click `New` to add each action.
- **Supports Resource Hierarchy:** No
- **Resource Delimiter:** Slash(/)
- **Evaluation Logic:** Permission Class
- **Permission Class:** oracle.adf.controller.security.TaskFlowPermission
- **Action Name Delimiter:** Comma(,)

d. Click **Save**.

e. Navigate to **Applications**, **OracleIdentityManager**, **Default Policy Domain**, **Resource Catalog**, **Resources**. Click **Open**.

f. Click **New** to create a new resource. Provide the following values, and then click **Save**.

- **Resource Type:** Select the resource type created in step 1(c).
- **Display Name:** Provide a display name for your custom taskflow.
- **Name:** Provide the name of the custom taskflow in the following format:

```
TASKFLOW_DOCUMENT#TASKFLOW_ID
```

For example:

```
/WEB-INF/request-approval-details-tf.xml#request-approval-details-tf
```

– **Description:** Provide a description for the custom taskflow.

Note: For each custom taskflow, you must create a resource as mentioned in step 1(f). You can use the same resource type that you created in step 1(c) for all your custom taskflows.

---

g. Navigate to **Applications**, **OracleIdentityManager**, **Default Policy Domain**, **Authorization Policies**. Click **Open**.

h. Select **Find By Principal**, and click **Search**. If you want to add to a displayed existing policy, then select it and click **Open**. Otherwise, click **New** to create a policy.

i. Add **name** and **principals** for the new policy.

j. Click **Add Targets** (+) sign. The Search Targets dialog box is displayed.

k. Click the **Resources** tab. Provide the resource type as defined in step 1(f), and then click Search.

l. Select the resource created in step 1(f). Click **Add Selected**.

m. Click **Add Targets**. The resource is added to the Targets table.

n. Expand the resource that you added to the table. Select the permissions you want to apply to the taskflow.

o. Click **Apply**.

2. Configure human task to use the custom taskflow. To do so:

a. Login to Oracle Enterprise Manager as WebLogic user.

b. Navigate to **Farm_*IAM_DOMAIN***, **SOA**, **soa_infra (*SOA_SERVER*)**, **default**, **DefaultRequestApproval [4.0]**.

c. Click **Component Metrics**, **Approval Task**.

d. Click the **Administration** tab.

e. Modify the URI in the existing entry to point to the custom taskflow, as shown:

– **Application Name:** *ANY_NAME*

– **Host Name:** *OIM_SERVER_HOSTNAME*

– **HTTP Port:** *OIM_HTTP_PORT*

– **HTTPS Port:** *OIM_HTTPS_PORT* (Optional)

– **URI:** /identity/faces/adf.task-flow?_id=request-approval-details-tf&_document=WEB-INF/request-approval-details-tf.xml

21.6.6 Testing the Custom Taskflow

To test custom taskflow:

1. Login to Oracle Identity Self Service as an end user.

2. Go to My Information, and modify the value of the Telephone attribute. A request is created and the task is assigned to the System Administrator.
3. Login to Oracle Identity Self Service as System Administrator.
4. Go to Pending Approvals.
5. Click the Task Details link of the corresponding request. The custom task details page is displayed.

21.7 Understanding Request Datasets

Request datasets are XML files that store the data that can be associated with requests of given types.

> **Note:** The information in this section is for reference. Oracle recommends that you do not modify, export, or import request datasets. To extend the user and application instance definition, use the Form Designer in Oracle Identity System Administration. See "Managing Forms" in *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager* for information about the Form Designer.

Table 21–2 lists the request types and the associated default request dataset file names that are shipped with Oracle Identity Manager.

Table 21–2 Default Request Datasets Shipped with Oracle Identity Manager

Request Type	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
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
Remove from Roles	RemoveRolesDataset.xml	Role

21.8 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.

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

21.8.1 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 27, "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>
</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 27-7 for information about registering plug-ins in Oracle Identity Manager.

21.8.2 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.

You can define the dataset validators and prepopulation adapters associated with the given plug-in. The request datasets associated with the plug-ins can be defined at the time of plug-in registration. The plugin.xml file is used to define the association between plug-ins and dataset validator or prepopulation adapters. The <metadata> node attached with the <plugins> element is used to define the association between data validators and prepopulation adapters.

> **Note:** DataSetValidator plug-in specified for a dataset cannot be overridden by the plug-in enhancement of specifying the validators metadata in the plugin.xml itself. For instance, the predefined dataset 'ModifyUserDataset' shipped with default validator does not get overridden by the custom implementation class. Therefore, the validator in dataset will be given precedence, currently there is no option to override it.

Example 21–1 shows how the plug-ins can be associated with data validators and prepopulation adapters.

Example 21–1 Associating plug-ins With Data Validators and Prepopulate Adapters

```
<?xml version="1.0" encoding="UTF-8"?>
<oimplugins xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <plugins pluginpoint="oracle.iam.request.plugins.RequestDataValidator">
     <plugin pluginclass=
"oracle.iam.plugin.appinst.ApplicationInstanceDataValidator" version="1.0"
name="AppInstDataValidator">
     <metadata name="DataValidator">
      <value>AppInstanceDataSet|ADAppDataSet|EBSDataSet</value>
   </metadata>
      </plugin>
  </plugins>
</oimplugins>
```

In this example, the following line of code indicates defining metadata xml element to indicate that the plug-in is associated with request data validator datasets:

```
<metadata name="DataValidator">
```

Note that `Attribute name="DataValidator"` in the metadata element indicates plug-in associated with request data validators.

Defining the names of the datasets to be associated with the current plug-in is indicated by the following line:

```
<value>AppInstanceDataSet|ADAppDataSet|EBSDataSet</value>
```

> **Note:** Request dataset names must be delimited by the single pipe character (|).

Consider the following scenarios:

- Scenario I: Provisioning Users to a Target System
- Scenario II: Provisioning or Modifying Entitlement Request

21.8.2.1 Scenario I: Provisioning Users to a Target System

Suppose Oracle Identity Manager is configured for provisioning users to the AD User APAC target. A RequestDataValidator specifies ADUserDataValidator is configured for the corresponding request dataset, as shown:

```
<plugin pluginclass= "oracle.iam.plugin.appinst.ADUserDataValidator" version="1.0"
name="ADUserDataValidator">
<metadata name="DataValidator">
<value>ADUserAPACDataSet</value>
</metadata>
</plugin>
```

Later, if the System Configurator wants to configure Oracle Identity Manager for provisioning users to the AD User EMEA target, then the System Configurator would create a new application instance, and associate a UI form with it. Request dataset would be auto-generated in the process. If the data-validator is to be re-used for this request dataset, then perform the following:

1. Edit plugin.xml of the ADUserDataValidator.
2. In the <metadata> <value> subtag, add the name of the new request dataset separated by a delimiter. For example:

   ```
   <value>ADUserAPACDataSet|ADUserEMEADataSet</value>
   ```

3. Re-register the data-validator plug-in.

21.8.2.2 Scenario II: Provisioning or Modifying Entitlement Request

Entitlement data provided as part of Provision Entitlement and Modify Entitlement request can be validated by creating a dataset validator plug-in and specifying the following in the plug-in metadata (plugin.xml):

```
<metadata name="DataValidator">
      <value>EBSForm.UD_EBS_RESP</value>
</metadata>
```

Here, EBSForm is the name of the form associated with the application instance on which the account is provisioned, and UD_EBS_RESP is the name of the child form corresponding to the entitlement. UD_EBS_RESP identifies the entitlement type.

21.8.3 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.

Defining metadata element to indicate that the plug-in is mapped to request data set attributes for filling up prepopulated data is indicated by the following line:

```
<metadata name="PrePopulationAdapater">
```

The association is defined by combining dataset name with attribute name in the following format:

```
<DATASET_NAME>::<ATTRIBUTE_NAME>
```

For example:

```
AppInstanceDataSet::First Name
```

Multiple attributes can be associated with the same prepopulation plug-in, where each association is separated by the single pipe character (|). For example:

```
<datasetname1>::<attribute1> |
<datasetname2>::<attributename2>|<datasetname3>::<attribute3>
```

The following is an example of prepopulation plug-in:

```
<plugins pluginpoint="oracle.iam.request.plugins.PrePopulationAdapter">
  <plugin pluginclass=
"oracle.iam.plugin.appinst.ApplicationInstancePrePopulateAdapter" version="1.0"
name="AppInstPrepopAdapter">
     <metadata name="PrePopulationAdapater">
   <value>
AppInstanceDataSet::First Name|ADAppDataSet::Last Name
  </value>
   </metadata>
   </plugin>
  </plugins>
```

Note: In addition to creating request datasets by using the catalog Form Designer, you can manually upload request datasets to MDS. You can also define DataSetValidator or PrepopulationAdapter elements within the request dataset. These dataset validators or prepoulation adapters configured in the dataset have the highest priority over other configuration.

For example, a plug-in EBSUserDataValidator is registered to associate it with a request dataset EBSUSerDataSet, but the dataset has not been created or uploaded. Another plug-in ADUserDataValidator is registered but not associated with any request dataset. When you later create the request dataset EBSUSerDataSet and use it for creating requests, the plug-in EBSUserDataValidator is called for validating the request data. Then, you add the DataSetValidator element to the request dataset EBSUSerDataSet that you manually created, and specify another plug-in ADUserDataValidator. When you use EBSUSerDataSet to create requests, the plug-in ADUserDataValidator is called. This is because ADUserDataValidator is configured as a part of the request dataset. If the DataSetValidator entry is removed from EBSUSerDataSet, then the plug-in EBSUserDataValidator is invoked to validate the request data.

21.9 Enabling Auto-Approval for Self Registration Requests

Approval policies can be configured at request level or operation level. They can use all the data available at the request and operational levels to construct a rule. The rule helps determine whether the request should be auto-approved or a SOA composite should be invoked. For information about enabling auto-approval for request and operation levels, see "Creating Approval Policies" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager*.

After the approval policies are updated, perform the following steps to enable auto-approval for self registration requests:

1. To assign the organization automatically, add a preprocess handler that sets the organization value in the orchestration. The order should be set such that it is executed before the approval-related handlers. Set this to be executed soon after the CreateUserPreProcessHandler. See Chapter 28, "Developing Event Handlers" for information about orchestration and event handlers.
2. By default, the Role/User Type set for the self registration requests is Part-Time Employee. If you want to overwrite this value, then change the plug-in configured in the SelfCreateUserDataset.xml dataset.
3. You can create a new plug-in implementation for the value/logic required and change the plug-in configured in the dataset to bring the new one in affect. The new plug-in must implement oracle.iam.request.plugins.PrePopulationAdapter.
4. Register the plug-in that you created by using the Plugin Registration Utility. For details, see "Registering and Unregistering Plug-ins By Using the Plugin Registration Utility" on page 27-8.
5. To update the request dataset:
   a. Export the /metadata/iam-features-requestactions/model-data/SelfCreateUserDataset.xml request dataset from the MDS, as described in "Exporting Metadata Files to MDS" on page 37-1.
   b. Update the name of the plug-in configured for Role attribute, as shown:

```
<AttributeReference name="Role" attr-ref="Role" available-in-bulk="false"
type="String" length="255" widget="dropdown"
lookup-code="Lookup.Users.Role"
required="true">
<PrePopulationAdapter
classname="oracle.iam.selfservice.uself.uselfmgmt.plugins.RolePrepopulateAd
apter"
name="RolePrepopulateAdapter"/>
</AttributeReference>
```

   c. Import the updated request dataset to MDS, as described in "Importing Metadata Files from MDS" on page 37-2.

22

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 describes SoD in the following sections:

22.1 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.

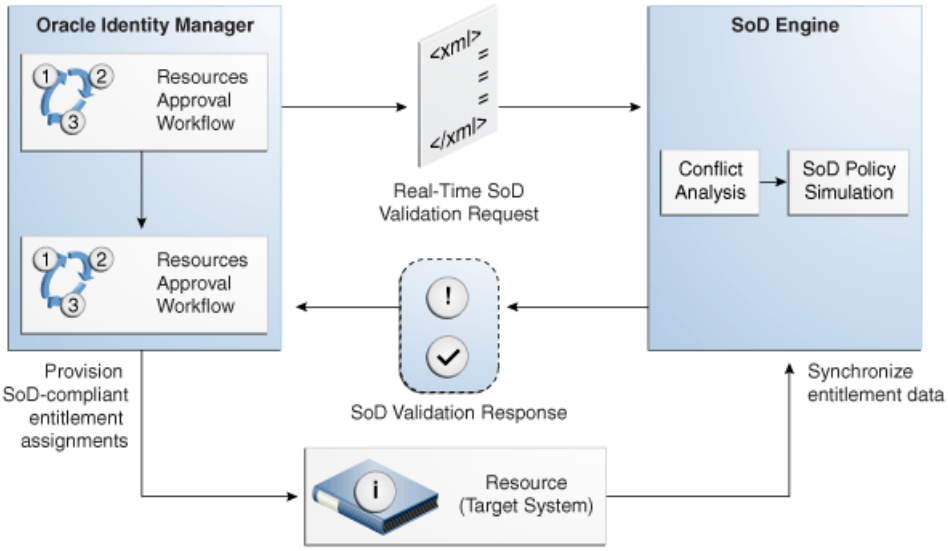
- The *resource approval workflow* is configured to validate requests in real time using an SoD engine. The SoD engine has predefined rules that are used to determine if the entitlement assignment would lead to SoD violations. The determination, once made, is returned to Oracle Identity Manager.
- The *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 22–1 shows the flow of data during the SoD validation process.

Figure 22–1 SoD Validation Process in Oracle Identity Manager



22.2 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 9.1.2.5 and later

> **Note:** With SAP UM 9.1.2.5, request entitlement does not work in Oracle Identity Manager 11*g* Release 2 (11.1.2.2.0).

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:

- SAP Governance, Risk and Compliance Access Control (GRC AC)

  If you want to configure and use the Access Risk Analysis or Access Request Management feature of this target system, then install one of the following:

  – SAP BusinessObjects Access Control 5.3 on SAP NetWeaver AS JAVA 7.00 Support Pack 14

    Install the VIRACLP 530_700_19, VIRAE 530_700_19, VIRCC 530_700_19, VIRFF 530_700_19, and VIRRE 530_700_19 components.

    Use ECC 6.0 with RTA components VIRSAHR SP 10 and VIRSANH SP 10.

    > **Note:** If you are using any other SAP application, then you must install the RTA component, which is compatible with that SAP application.

  – SAP BusinessObjects Access Control 10 on SAP NetWeaver AS ABAP 7.02 Support Pack 7

    Install the GRCFND_A SP 10 component.

    Use ECC 6.0 with RTA component GRCPIERP SP 9.

    > **Note:** If you are using any other SAP application, then you must install the RTA component which is compatible with that SAP application.

- SIL Provider for Oracle Application Access Controls Governor (OAACG) release 8.6.4

  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) 11*g* Release 1 Patch Set 1 Bundle Patch 2 (11.1.1.5.2)

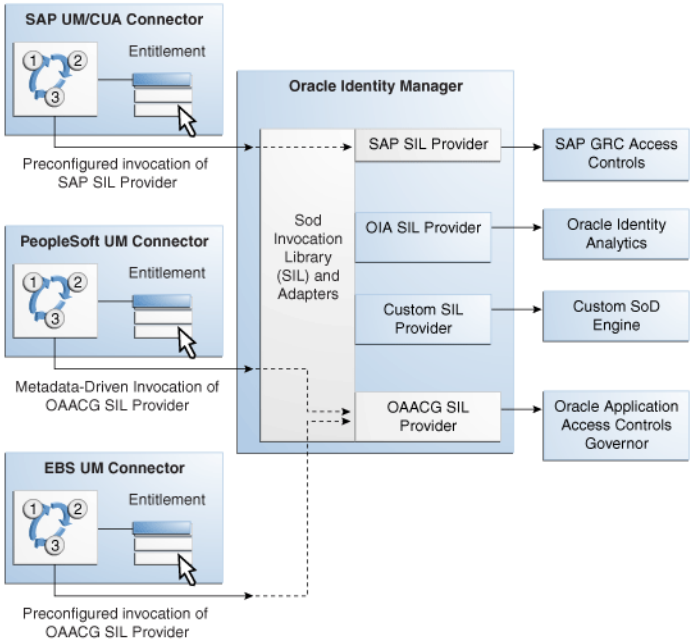
  This provider is also known as the OIA SIL Provider.

Note: You can download the current version of OIA from the My Oracle Support web site by navigating to the following URL and searching for Patch 13641335.

https://support.oracle.com

Figure 22–2 shows the architecture of SoD implementation in Oracle Identity Manager.

Figure 22–2 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.

22.3 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 URL:

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 9.1.2.5 and later

22.4 Deploying the SIL and SIL Providers

SIL registration is provided by default for the same 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 22.10, "Custom Combination of Target Systems and SoD Engines."

22.5 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

22.5.1 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 8.6.4.5259 is supported with Oracle Identity Manager 11*g* Release 2 (11.1.2.2.0). OAACG 8.6.4.5259 can be directly installed. If you have existing OAACG 8.6.3 GA installation, then you can upgrade it to 8.6.4.5259.

To upgrade from 8.6.3 GA to 8.6.4.5259:

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 **AACG 8.6.4**. Select the appropriate platform, and click **Search**. The following patches and other latest ones are displayed:
   - 14528065 - ORACLE APPLICATION ACCESS CONTROLS GOVERNOR 8.6.4.3347
   - 14786771 - ORACLE APPLICATION ACCESS CONTROLS GOVERNOR 8.6.4.4240
   - 16178807 - ORACLE APPLICATION ACCESS CONTROLS GOVERNOR 8.6.4.5259
5. Download the patches listed in step 4. Perform the upgrade in the following sequence:

   8.6.3 to 8.6.4.3347 to 8.6.4.4240 to 8.6.4.5259

   Perform the OAACG upgrade or install by referring to the OAACG install guide.
6. For each deployment, re-run Access ETL against the active data sources.

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 22-37, 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.

22.5.2 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.

22.5.3 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

---

Note: Oracle Identity Analytics 11.1.1.5.0 Patch Set 2 is certified for Oracle Identity Manager 11*g* Release 2 (11.1.2.2.0).

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 22-37, provide the user name and password of this account.

See the Oracle Identity Analytics documentation for information about creating the account.

Oracle Identity Analytics 11.1.1.5.2 provides information on the entitlements that caused a conflict. This information is visible under SoD Details. The entitlements under SoD Details will be displayed in an encoded form (for example, `4~695~24123`) because this is how policies are created on OIA.

Note: The Oracle Identity Analytics Admin account with username `rbacxadmin` can also used.

Synchronizing Oracle Identity Manager Metadata With Oracle Identity Analytics

Import the resource metadata and resources from Oracle Identity Manage 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 Manage.

See the Oracle Identity Analytics documentation for more information.

22.6 Enabling and Disabling SoD

The following sections contain information on enabling and disabling SoD.

- Enabling SoD
- Disabling SoD

22.6.1 Enabling SoD

To enable the SoD feature:

1. Set the 'Segregation of Duties (SOD) Check Required' system property to `true`. See "Managing 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 22-37.

   Note: To configure with OAACG 8.5 or higher, add a new field to this IT resource with name as `sodServerURL` and value http(s)://*HOST_NAME*:*PORT*/*URI*, where *URL* is `grcc/services/GrccService`. For OAACG8.2.1, the value of URL is `ags/services/AGService`.

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 22-13.

Note: The DefaultSODApproval workflow consists of two approval tasks. When the workflow is triggered, an approval task is generated that is assigned to System Administrator, and SoD check is performed. Again an approval task is generated that is again assigned to System Administrators role. Any user who is a member of the System 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, and on the left tab, expand **Weblogic Domain**.
   b. Open ***OIM_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.

22.6.2 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`.
- Set the value of the topologyName parameter for the connector IT Resource to `None` so that SoD check is not performed. Alternatively, you can remove this parameter to disable SoD check.

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 described in this section. If you want to perform the default SoD check in approval and only disable the SoD check in BPEL, then remove the approval policy for SoD, or remove the call to SoD Check web service from the approval workflow.

22.7 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

22.7.1 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`.

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
   ```

22.7.2 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.

22.7.3 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_SERVER***. 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_SERVER***.

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

---

22.8 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 and SAP User Management). This section discusses the following procedures:

- Modifying the Approval Workflow for SoD
- Modifying the Provisioning Workflow for SoD

22.8.1 Modifying the Approval Workflow for SoD

To modify the approval workflow for SoD validation:

Note: Forms are created from the UI based on the connector process form. Request dataset is no longer required in Oracle Identity Manager.

---

1. In the IT resource of the connector, create the TopologyName parameter if it does not already exist. Figure 22–3 shows a sample IT resource in which this parameter has been added:

Figure 22–3 The TopologyName Parameter

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 form 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.

---

2. 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. Currently, no patch is required for Oracle Identity Manager.

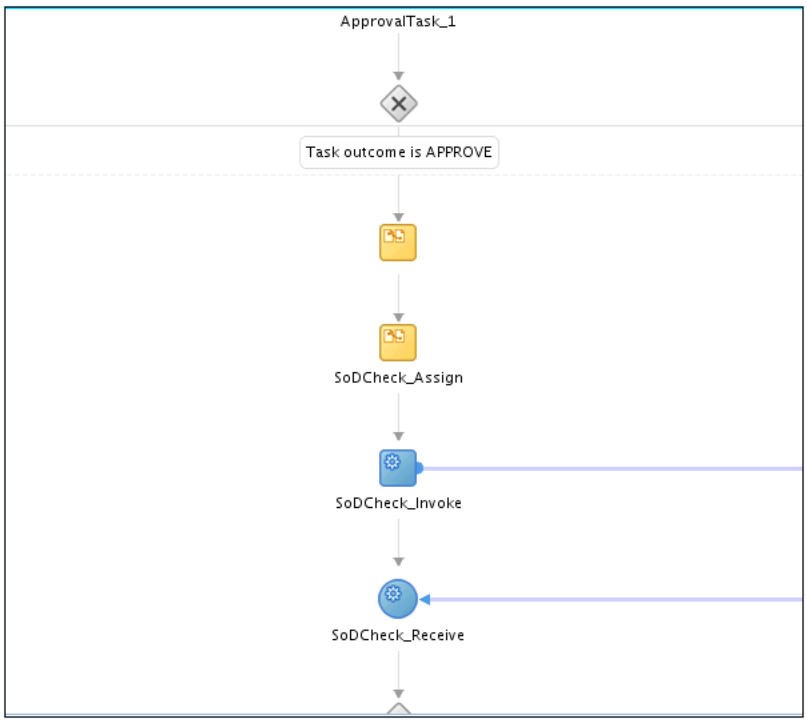
For using the default workflow, see "Applying the Workflow By Using Approval Policy" on page 22-25. This workflow contains an approval task that is assigned to the system administrator. After this approval task, a call is made to the SoDCheck Web service to return the SoD check result. The workflow with SoDCheck Web service call is shown in Figure 22–4.

Note: There are two levels of approval for any request, 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 application instances and users. After this level of approval, separate requests are created for each combination of application instance and user. The workflow performs SoD check separately for each application instance 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.

Approval-only field is not supported in Oracle Identity Manager 11*g* Release 2 (11.1.2.2.0).

Figure 22–4 Workflow with SoDCheck Web Service Call



After this, a switch case with approval tasks are assigned to the System 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 22–5:

Figure 22–5 Switch Case With Approval Tasks

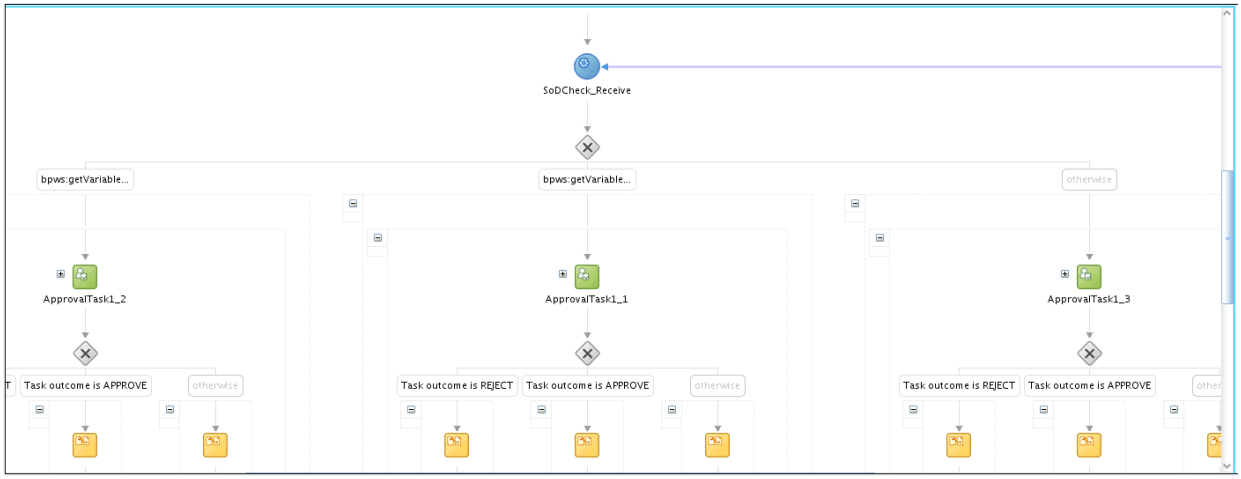
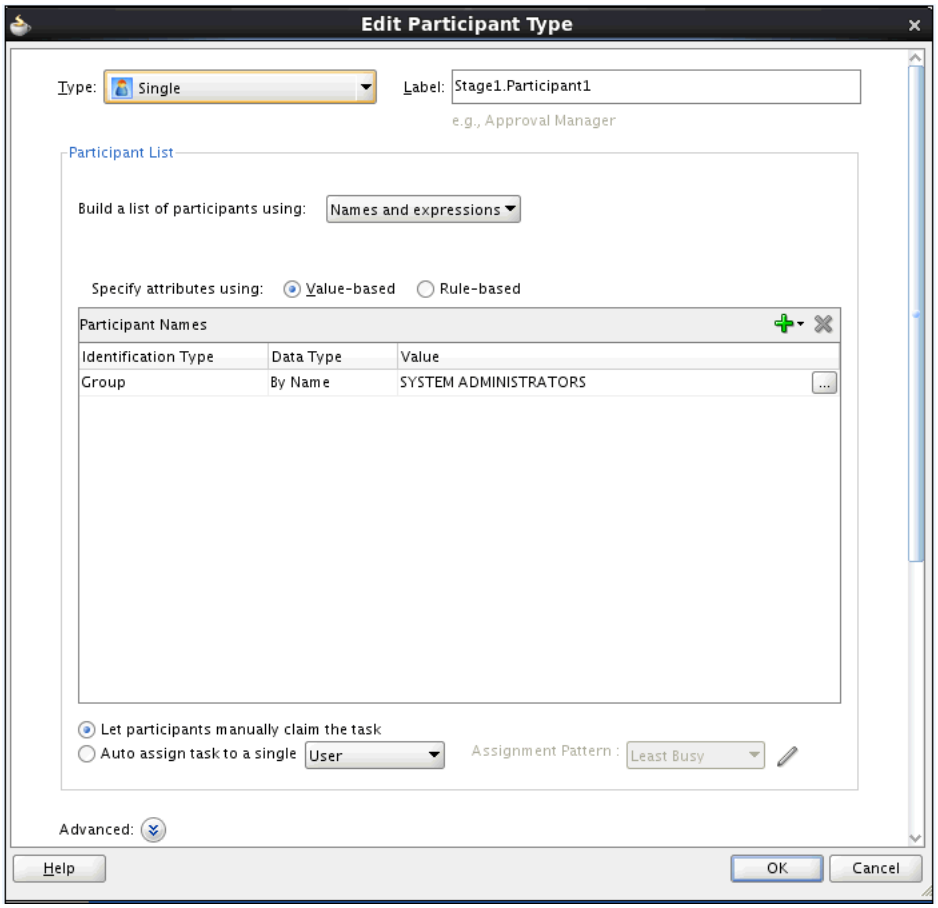


Figure 22–6 shows the assignment of the approval task.

Figure 22–6 Assignment of the Approval Task



Approval workflow has migrated to BPEL in Oracle Identity Manager 11*g* Release 2 (11.1.2.2.0), 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:

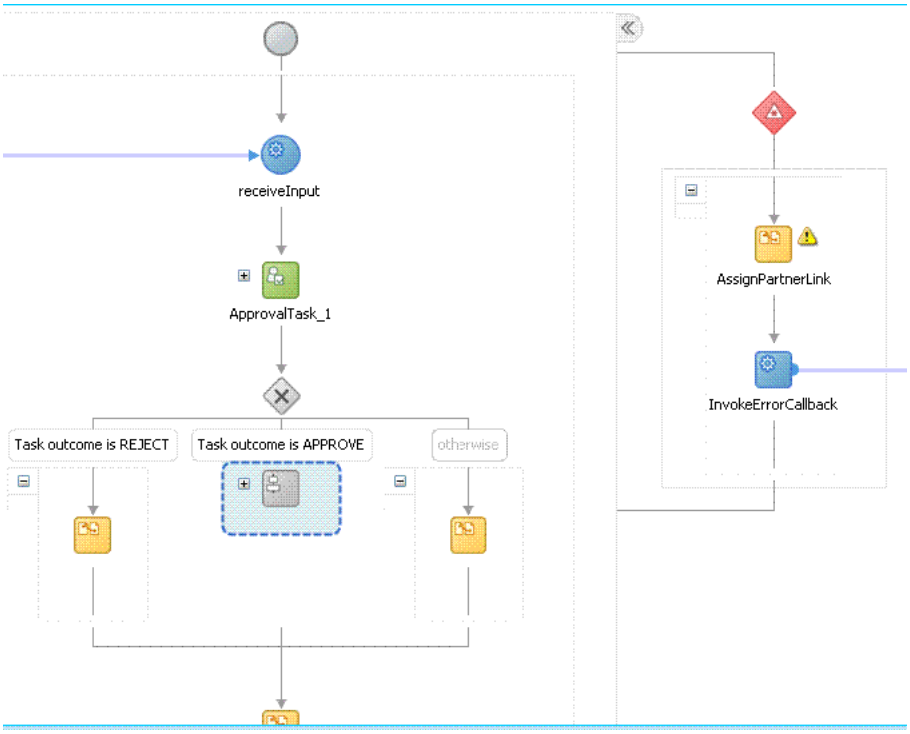
```
ant -f new_project.xml
```

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 22–7 shows the default workflow modified to perform SoD Check after human approval:

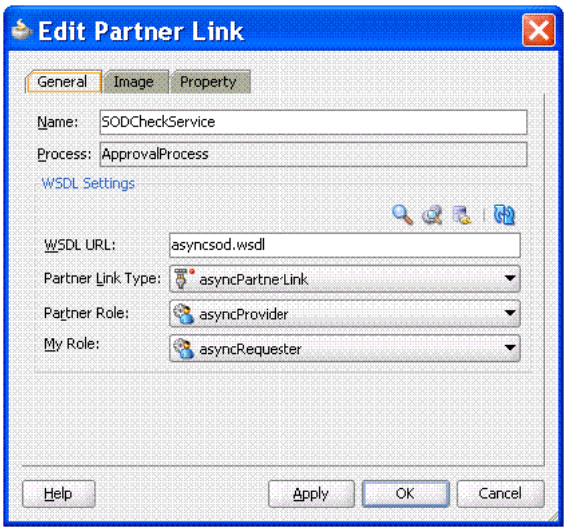
Figure 22–7 Modified Workflow To Perform SoD Check



c. Extract *OIM_HOME*/workflows/composites/DefaultSODApproval.zip and copy SodCheckServicePortImplService.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 asyncsod.wsdl as the WSDL file. The SoDCheck partner link is as shown in Figure 22–8:

Note: BPEL connects to all external entities through a partner link.

Figure 22–8 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 Rules** tab.

iii) Click **Add**, and then select **Copy Operation**. Provide the values for the variables, as shown in Table 22–1:

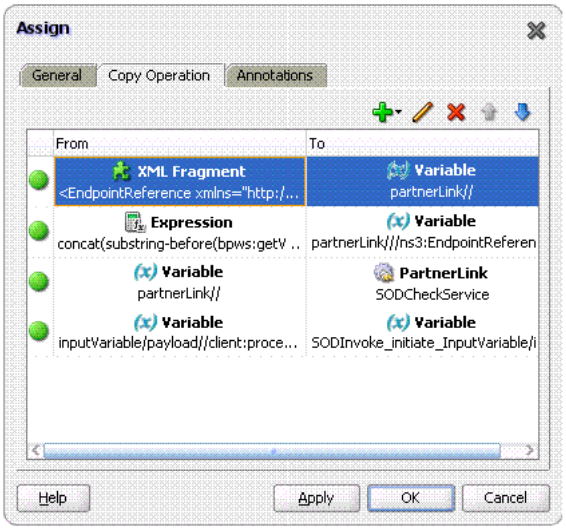
Table 22–1 Variables to Assign

Copy From	Copy To
XML Fragment	Variable
`<EndpointReference xmlns="http://www.w3.org/2005/08/addressing"> <Address/> </EndpointReference>`	partnerlink
Expression concat(substring-before(bpws:getVariableData('inputVariable','payload','/client:process/ns1:url'), "/workflowservice/CallbackService"), '/sodcheck/SoDCheckInitiateService')	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 22–9 shows the final assign activity:

Figure 22–9 Final Assign Activity



– **INVOKE:** The details for this activity are:

Interaction Type: Partnerlink

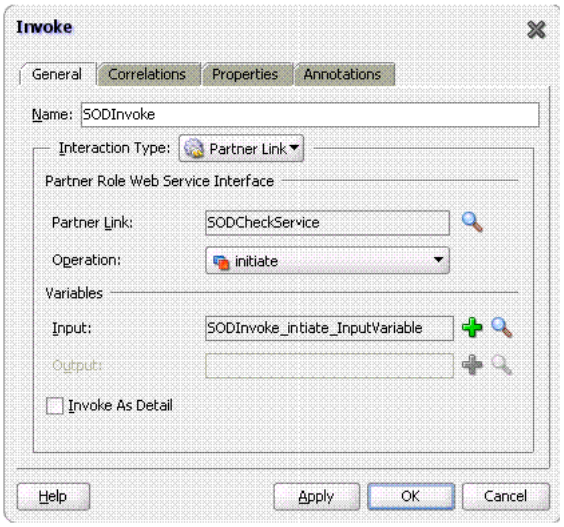
Partnerlink: SODCheckService

Operation: Initiate

Input Variable: SODInvoke_initiate_InputVariable

Figure 22–10 shows the Invoke dialog box with sample values in the fields:

Figure 22–10 The Invoke Dialog Box



– **RECEIVE:** The details for this activity are:

Interaction Type: Partnerlink

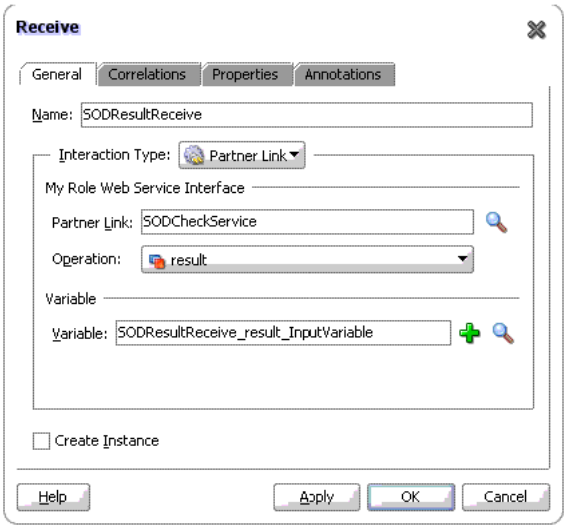
Partnerlink: SODCheckService1

Operation: Result

Variable: SODResultReceive_result_InputVariable

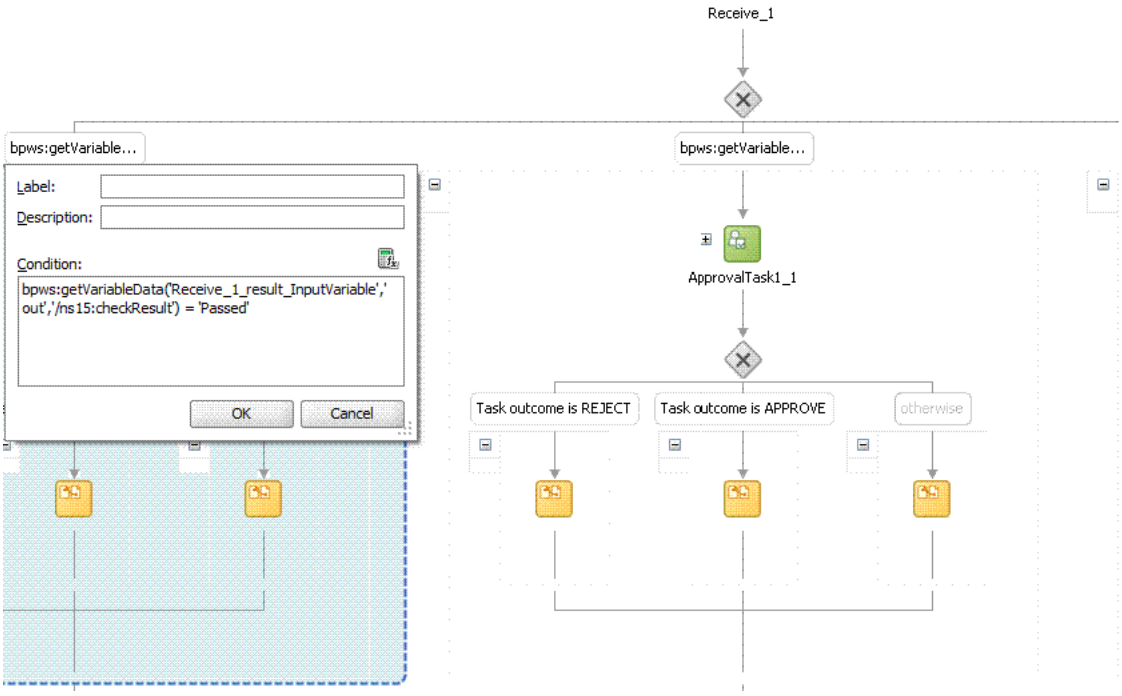
Figure 22–11 shows the Receive dialog box with sample values in the fields:

Figure 22–11 The Receive Dialog Box



– **SWITCH:** This activity is to switch between workflows based on SODCheck Result. The switch case is as shown in Figure 22–12:

Figure 22–12 Switch Case



– **New Human Tasks:** A new human task may be created and assigned to an approver other then 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 System Administrators role. DefaultSODApproval always assigns approval task to the System Administrators role.

Note: The SoDCheck Web service can be called multiple times.

---

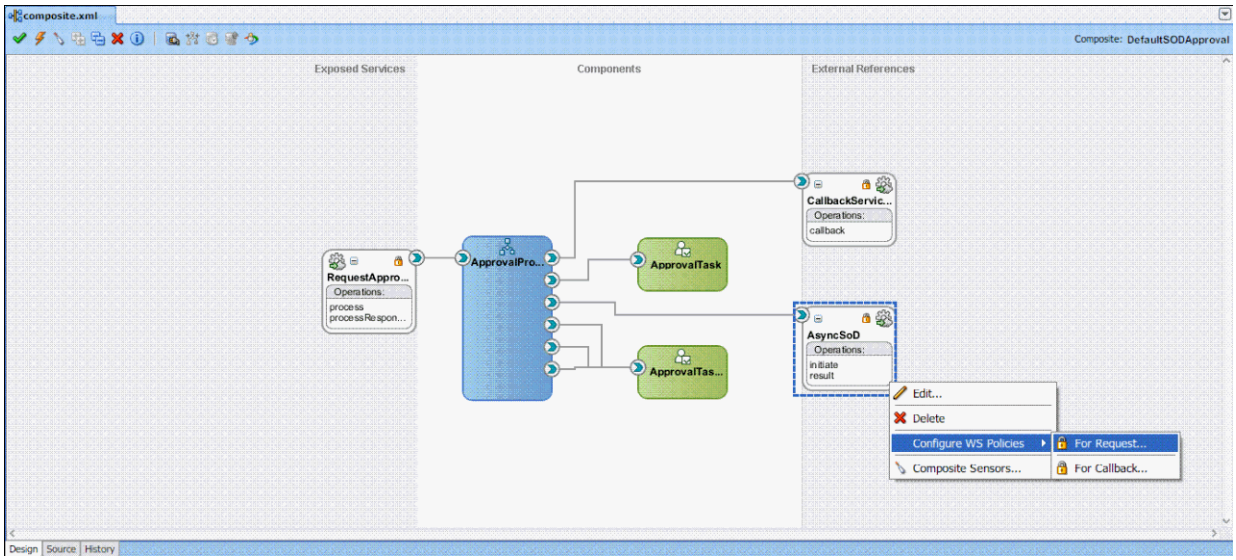
e. Applying SAML policy for request and callback for the AsyncSoD Web service:

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:

i) Right-click **AsynchSoD** Web service, and select Configure WS Policies, and then select For Request, as shown in Figure 22–13:

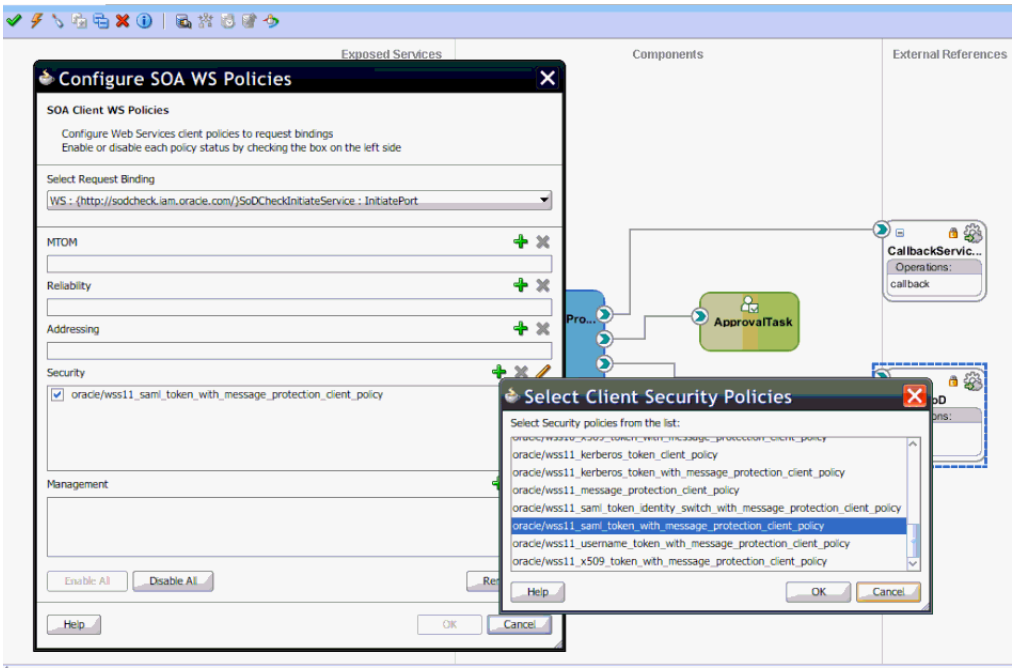
Figure 22–13 Configuring WS Policies for Request



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 22–14, and then click **OK**.

Figure 22–14 Select Client Security Policies



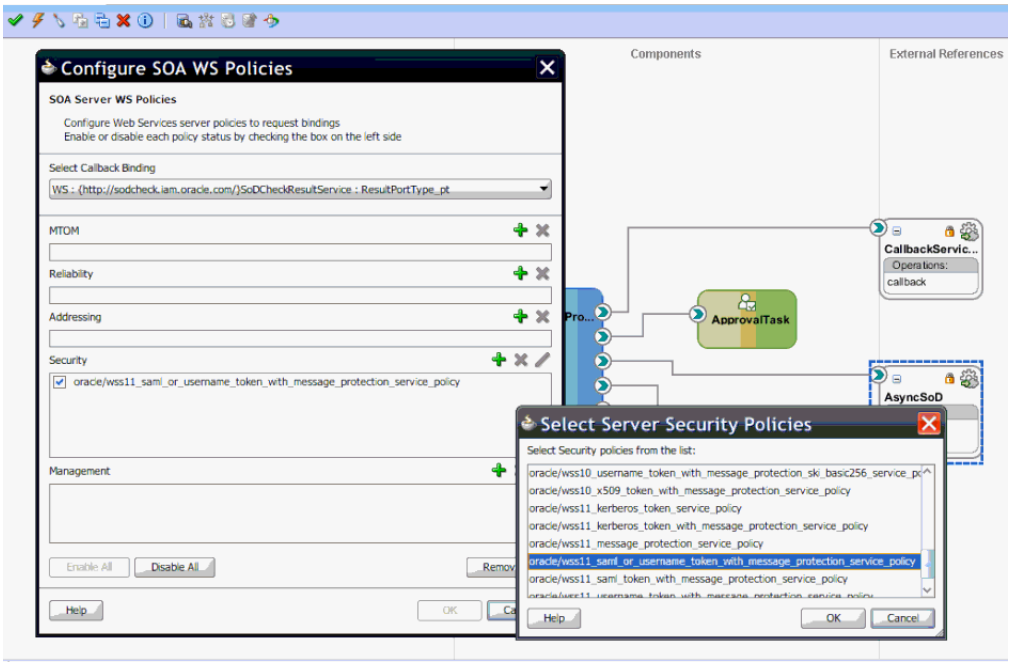
To apply SAML or Username token with Message Protection Service Policy for callback:

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 22–15, and then click **OK**.

Figure 22–15 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: "Developing Workflows for Approval and Manual Provisioning" on page 21-1 for general procedure for creating a new workflow

Applying 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 an application instance, then create a policy for the Provision Application Instance request model. See "Creating Approval Policies" in the *Oracle Fusion Middleware Administrator'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.

22.8.2 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:

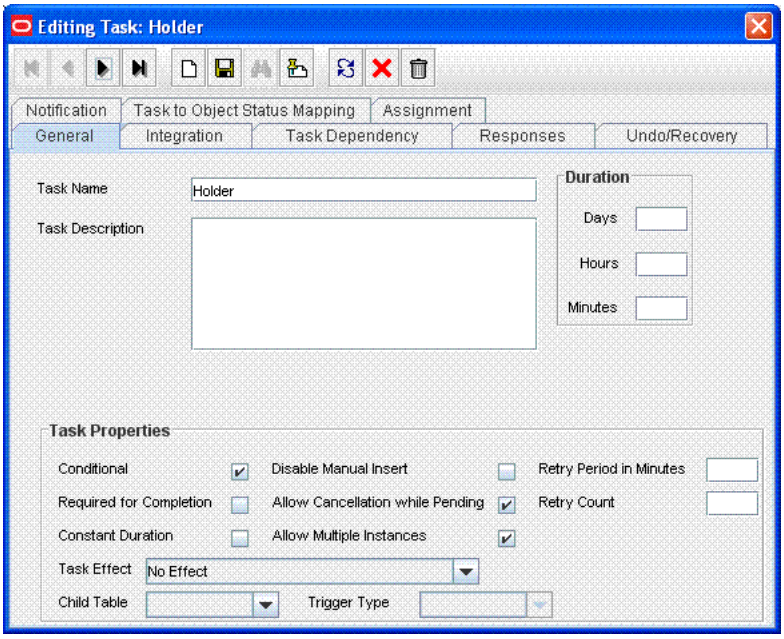
- Is the provisioning related to a request?
- If yes, is the SoDCheckStatus field set to `SoDCheckCompleted`?
- If yes, then do not perform the SoD validation process during entitlement provisioning.

> **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:

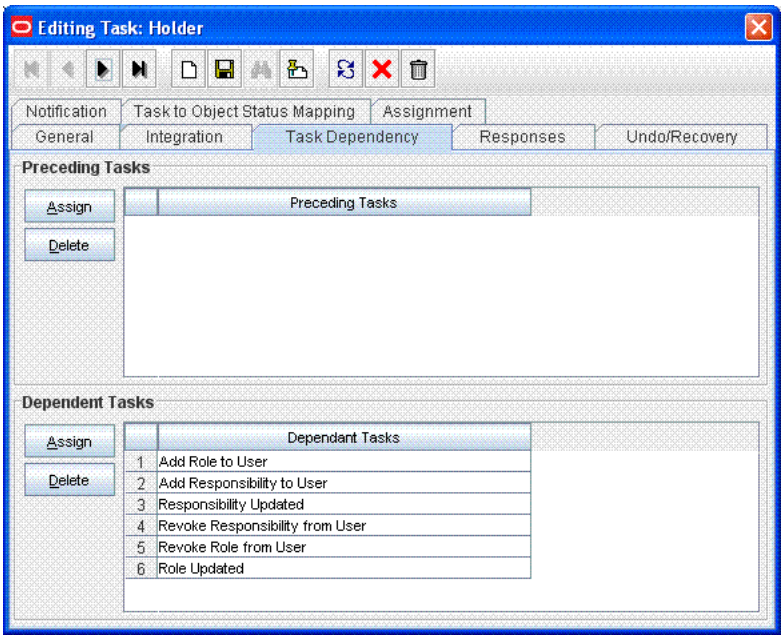
1. Add a Holder task to the provisioning workflow. This task must be made conditional and the Allow Multiple instances option must be selected.

   The following figure shows this Holder task:

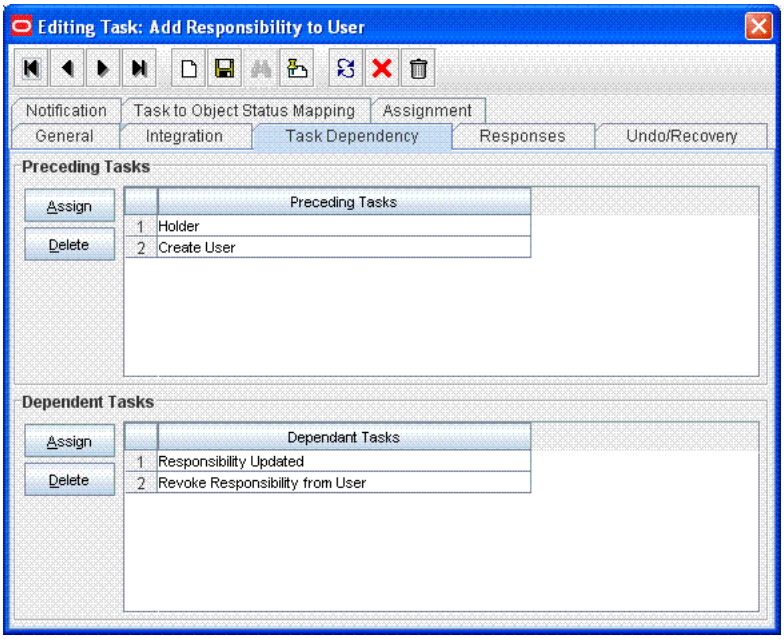


2. Make the connector insert, update, and revoke entitlement tasks dependent on the Holder task.

   The following figure shows all entitlement tasks of the Oracle e-Business User Management connector dependent on the Holder task:

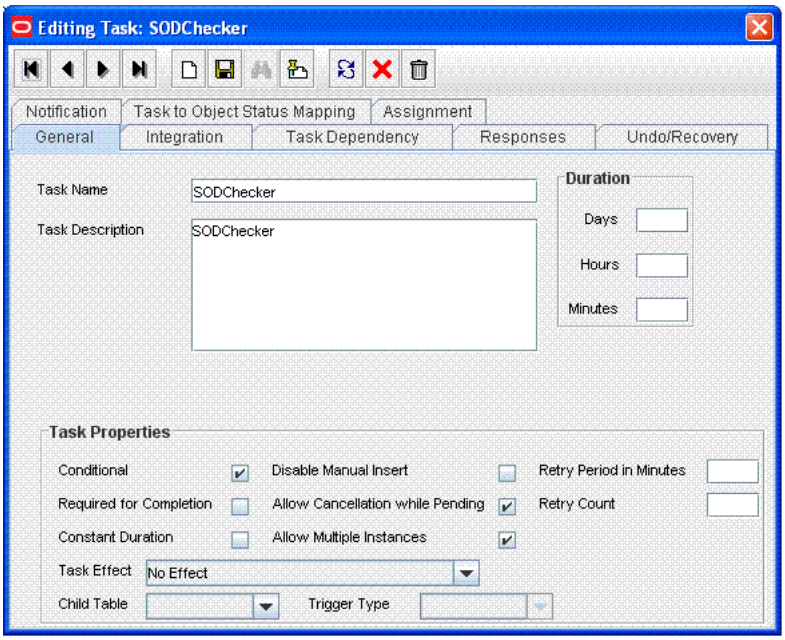


The following figure shows the Holder task as a preceding task of the Add Responsibility to User task:



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:

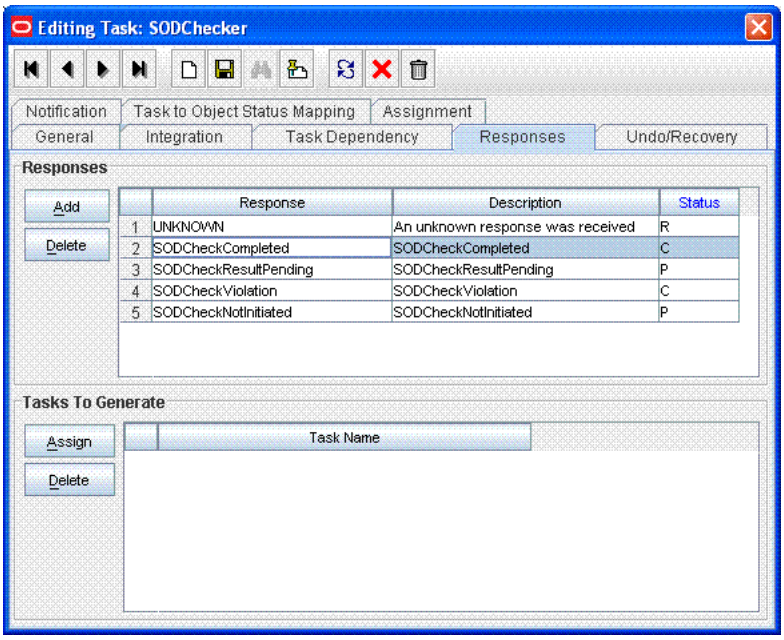


4. Attach the InitiateSODCheck process task adapter to the SoDChecker task.

   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. **Note:** For request provisioning of the EBS 9.1.0.7.0 resource with conflicting entitlements, the SodCheckViolation field in the process form is not updated. The entitlement violation is mapped to the field with the SoDCheckEntitlementViolation label, while the EBS resource has the field with the SoDCheckViolation label. Therefore, the mapping does not occur. Direct provisioning and provisioning through access policy successfully takes place with the SoDCheckViolation field label. To workaround this issue for request provisioning, change the SoDCheckViolation field label to SoDCheckEntitlementViolation in the EBS form by using the Design Console. **Note:** If the value is SoDCheckEntitlementViolation, then all types of provisioning, such as request, direct, and access policy, works fine. Therefore, you can keep the value as SoDCheckEntitlementViolation instead of changing the values.
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:



22.9 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" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager* for more information.

22.10 Custom Combination of Target Systems and SoD Engines

This section contains the following topics:

- Using a Custom Target System
- Adding Custom SoD Engine

22.10.1 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 and SAP R3. You must also perform the procedures given in "Adding Custom SoD Engine" on page 22-37 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 22-37.

2. Create Java class implementations of the IdMvsSoDDataTransformationOper interface for the connector. See "Creating the Transformation Layer" on page 22-30 for instructions.
3. Deploy the transformation service component. See "Deploying the Transformation Layer" on page 22-30.
4. Add entries in the registration XML file for the new target system. See "Modifying the Registration XML File" on page 22-31 for instructions.
5. Perform the procedure described in "Configuring Workflows on Non SoD-enabled Connectors" on page 22-13.
6. Mark child process forms that hold entitlement data. See "Marking Child Process Form Tables That Hold Entitlement Data" on page 22-29 for instructions.
7. Register the new target system. See "Registering the New Target System" on page 22-32 for instructions.

22.10.1.1 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.

22.10.1.2 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 11*g* Release 2 (11.1.2.2.0), 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.

22.10.1.3 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.

22.10.1.4 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 22-30. 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 22-37

- *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.

22.10.1.5 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

22.10.1.5.1 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:

- Before running this utility, set *APP_SERVER*, *OIM_ORACLE_HOME*, *JAVA_HOME*, *MW_HOME*, *WL_HOME*, and *DOMAIN_HOME*.
- 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 22–1 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
   [Enter the Context Factory:]CONTEXT_FACTORY
   ```

   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 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 must be followed for SAP, GRC, OIM SDS, and OIA 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 22-36.

    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 22-36 for information about the child elements of the Topologies element.

17. Import SILConfig.xml back to MDS.

Example 22–1 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 22–1 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
```

22.10.1.5.2 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                 oiminstance             1
EBS                 ebsinstacne             2
SAP                 sapinstance             3
OAACG               oaacginstance           4
```

3. Copy these instance names for your reference.

22.10.2 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 22-29 if you are using a target system other than Oracle e-Business Suite 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 22-37.
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 22-37.
3. Create Java class implementations of the interfaces for the SIL provider. See "Implementing the Service Components for the Provider" on page 22-38 for instructions.
4. Deploy the service components. See "Deploying the Service Components" on page 22-39.
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 22-39 for instructions.
6. Register the new SoD engine. See "Registering the New SIL Provider" on page 22-41 for instructions.

22.10.2.1 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.

22.10.2.2 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 4, "Developing Application Instances" 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 22-5 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 `true` if the SoD engine accepts only HTTPS communication requests. Otherwise, enter `false`.	`false`
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.

22.10.2.3 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.

22.10.2.4 Deploying the Service Components

Service components created in "Implementing the Service Components for the Provider" on page 22-38 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.

22.10.2.5 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="isNotNullOAACG"/>
</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 22-30. 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 22-37

- *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.

22.10.2.6 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 22-32 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 22-36 for information on entering data about the new target system in the SILConfig.xml file.

22.11 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.

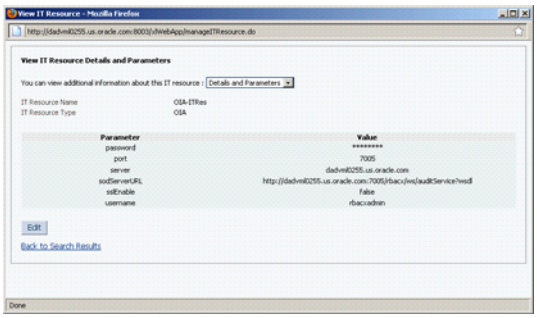
- Enabling Role SoD Check
- Using Role SoD Check

Note: Integration between Oracle Identity Manager and Oracle Identity Analytics is a pre-requisite for performing an SoD Check with Oracle Identity Analytics.

22.11.1 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 "Managing 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:



22.11.2 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

22.11.2.1 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 the Audit Reviewer role.

Perform the following steps:

1. Login to Oracle Identity Self Service as the user.
2. Under My Profile, click **My Access**. Click the **Roles** tab, and then click **Request Roles**. The Catalog page is displayed where you can search for the role to be requested.
3. Search for the specific role you want to assign to the user, for example Asset Management Team.
4. Click **Add to Cart**. This shows one item in the cart. Then, click **Checkout**.
5. Click **Submit** to submit the request for the role. The request is created.
6. Navigate to **Requests**, **Track Requests**.
7. Search for the request that you created. Open the request to see the SoD Check results. The SoD Status column should display SoD check complete.
8. Click on the SoD Status to see the SoD Check result, as shown in the following figure:

SoD Check result is Failed because the Asset Management Team role, for which request is raised, and Audit Reviewer role, which the user already had, are conflicting.

The SoD Check result is available before request-level approval. Here, SoD Check has failed, but the administrator can approve the request to assign the conflicting role to the user.

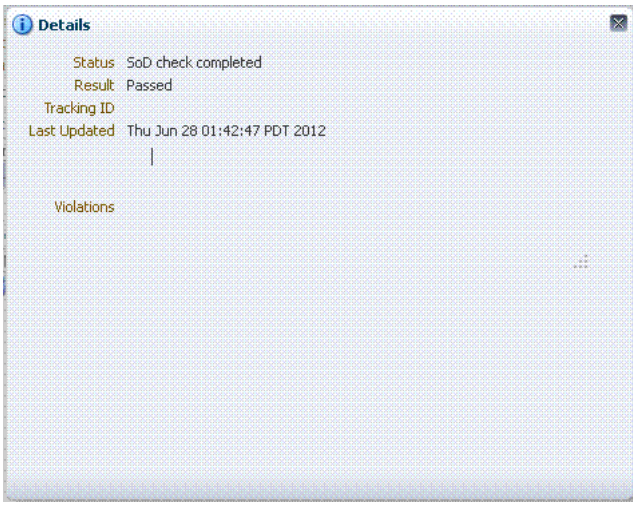
22.11.2.2 SoD Check When A User Revokes a Role

This procedure assumes that the user has already been assigned the role being revoked.

Perform the following steps:

1. Login to Oracle Identity Self Service as the user.
2. Navigate to **My Profile**, **My Access**, and then click the **Roles** tab. The roles assigned to the user are displayed.
3. Select the role to be removed, and then click **Remove Roles**. The Remove Roles page is displayed with the selected role as the cart item.
4. Click Submit. The request is created.

   You can open the request details to see the SoD Check result, as shown in the following figure:



The SoD Check result is now Passed because a conflicting role has been removed.

22.11.2.3 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 Oracle Identity Self Service as the System Administrator.
2. Navigate to **Administration**, **Users**, search for users, and then open the details of the selected user.
3. Click the **Roles** tab, and then click **Request Roles**. The Catalog page is displayed.
4. Search for the Asset Management Team and Audit Reviewer roles, and add them to the cart. Click **Checkout**. The cart details is displayed with the added roles.
5. Click **Submit**. A request is created. Because this is a bulk request, SoD Check is not initiated for it.
6. Login to Oracle Identity Self Service as the system administrator, and navigate to **Inbox**, **Pending Approvals**, and then approve the request. Two child requests are created and SoD Check is performed for each child request.
7. Navigate to **Track Requests**, and open each child request to see the SoD details. The following figure shows the SoD details for the request created for Audit Reviewer role:

The following figure shows the SoD details for the request created for the Asset Management Team role:

22.11.2.4 SoD Check When an Administrator Requests To Revoke Roles

This procedure allows the system administrator to revoke a role from the user:

1. Login to Oracle Identity Self Service as the system administrator.
2. Navigate to **Administration**, **Users**, search for users, and then open the details of the selected user.
3. Click the **Roles** tab, select the roles to be removed, and then click **Remove Role**.

> **Note:** A request is created to remove more then one role. For a single role, no request is created, and therefore, no SoD Check is performed.

4. On the Remove Roles page, click **Submit**. A request is created.
5. Approve the request. Two requests are created and SoD Check is performed for each request. The following figure shows the SoD Check result for the request created for the Audit Reviewer role:

The following figure shows the SoD Check result for the request created for the Asset Management Team role:

Because the request is raised by the System Administrator and there is no SoD Conflict, both the child requests are approved by default.

Note: By default, operational-level approval is triggered for requests that are raised by the System Administrator only if the SoD Check result is Failed because a conflict is detected.

22.12 Using SoD in Provisioning Workflow

This section describes various use cases related to SoD:

Note: The procedures in this section are for Synchronous SoD Engine, for example OAACG, for which you do not need to run the scheduled job to complete the SoD check.

- Provisioning Application Instance With Child Data
- Modifying Application Instance to Add or Delete Child Data
- Provisioning Entitlements to a User
- Revoking Entitlements From a User
- Requesting for Roles and Entitlements
- Requesting for Roles and Application Instances With Child Data
- Request Provisioning With the DefaultSODApproval Workflow
- Requesting for Role With an Access Policy Attached
- Provisioning Based on Access Policies Without Approval
- Provisioning Based on Access Policies With Approval
- Requesting for Entitlements From Two Application Instances

22.12.1 Provisioning Application Instance With Child Data

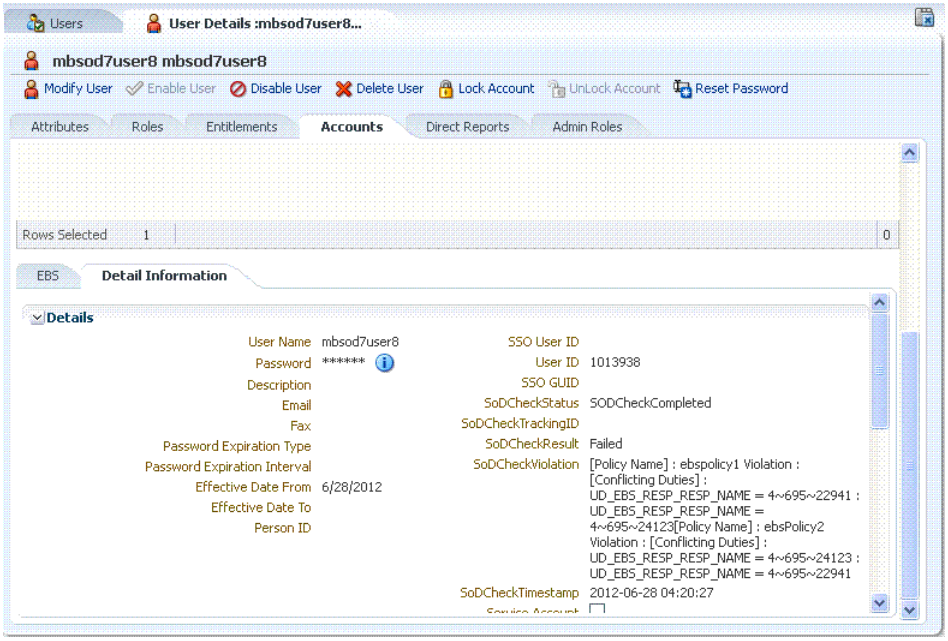
To provision an application instance as the system administrator:

1. Create a user whose account is to be created on the target system.
2. On the User Details page, click the **Accounts** tab, and click **Request Accounts**. The Catalog page is displayed.
3. Search for the application instance, for example EBS.
4. Add the application instance to cart, and checkout.

   The form that is added to the application instance is displayed on the Cart Details page. This form contains the default SoD check fields, such as SoDCheckStatus, SoDCheckTrackingID, SoDCheckResult, SoDCheckTimestamp, and SoDCheckEntitlementViolation. These fields are populated with values for SoD check.
5. Provide the required details in the form. 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. Click **Ready to Submit**, and then submit the request. Because this a request for a single user for a single application instance, no request is created and the application instance is directly assigned to the user.

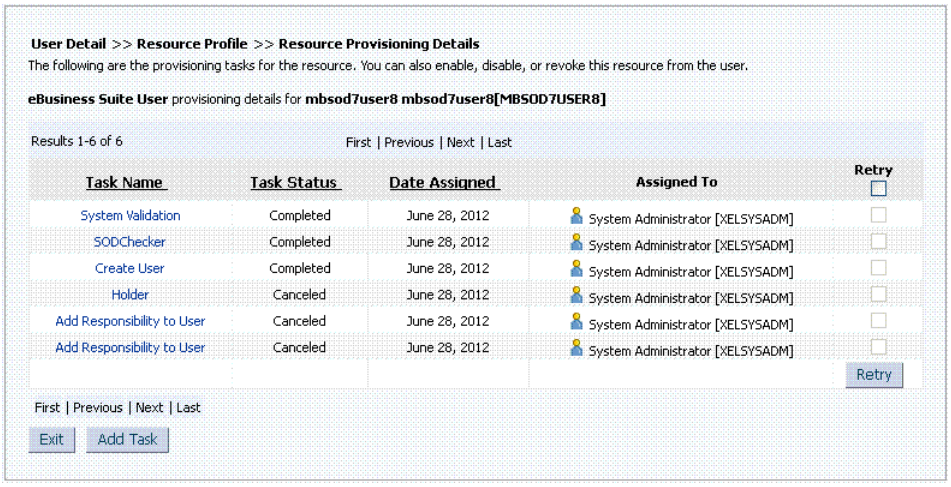
   Because synchronous SoD Check has happened, you can see the SoD Check result on the Account Details page. If you select conflicting entitlements, then the SoD Check will fail and the entitlements will not get provisioned on the target system. Figure 22–16 shows the SoD Check Result.

Figure 22–16 Conflicting Entitlements



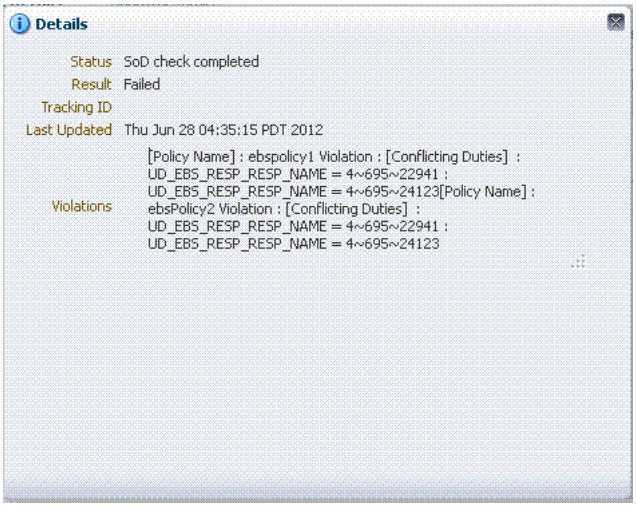
If you open the resource history, the Holder task is displayed in Canceled state because SoD Check resulted in a conflict. In addition, the SoDChecker task is in Completed state indicating that SoD Check has completed. Figure 22–17 shows the resource provisioning details.

Figure 22–17 Resource Provisioning Details



If the steps to provision an application instance are performed by a user with viewer admin role, a request is created. SoD Check result is visible in this request. Approver has the authority to approve or reject the request after seeing the SoD Check result. Figure 22–18 shows the SoD Check result in request details:

Figure 22–18 SoD Check Result in Request Details



22.12.2 Modifying Application Instance to Add or Delete Child Data

Whenever you open the user details, select an already provisioned account, and try to modify it by adding, updating, or deleting an entitlement in the child form, SoD check is triggered. If this operation is performed by the system administrator, new Holder and SoDChecker tasks are generated. If the new entitlement conflicts with the old ones, then the new entitlement is not provisioned. Otherwise, the new entitlement is provisioned on the target system.

If the operation is performed by the user with view admin role, a new request for modifying application instance is created. SoD Check result can be seen in this request.

22.12.3 Provisioning Entitlements to a User

Entitlements are first level entities in Oracle Identity Manager. Therefore, entitlements can be directly requested, as follows:

- **System Administrator requests for a single entitlement for a user:** The SoD Check is done and entitlement is granted to the user if it does not conflict with any of the existing entitlements. SoD Check results can be seen in the Account Details page. The Holder and SoDChecker tasks are created for performing the SoD Check.
- **System Administrator requests for multiple entitlements:** For bulk operation, requests are created. Therefore, if the system administrator requests for two entitlements, then a request r1 is created. SoD Check is always done at the child request level, and therefore, no SoD Check is done for r1. After the request-level approval is obtained for r1, two child requests r2 and r3 are created. SoD Check is done for both the child requests. Result can be seen in the Request Details.
- **User requests for single entitlement:** This results in a request being created because the user cannot directly get the entitlement. An approver must approve it. SoD Check is done for the request. No Holder or SODChecker tasks are created.

22.12.4 Revoking Entitlements From a User

Revoke entitlement use cases are similar to provisioning entitlements, as described in "Provisioning Entitlements to a User" on page 22-49. When the last entitlement is revoked from the user, no SoD Check is done because the user does not have any entitlements.

22.12.5 Requesting for Roles and Entitlements

Oracle Identity Manager supports heterogeneous requests that allow you to request for roles along with entitlements. To do so, open the Catalog page, search for the required entities, select the entities, and submit. This result in the creation of a request. When this request is approved, child requests are created for the requested entities. SoD Check is done for each of these child requests. Roles and entitlements are sent for separate SoD Checks.

Note: SoD Check conflict is not detected between roles and target system entitlements. If request is raised for the two, they go through separate SoD Checks.

22.12.6 Requesting for Roles and Application Instances With Child Data

This is similar to requesting for roles and entitlements, as described in "Requesting for Roles and Entitlements" on page 22-49. Separate SoD Check is done for application instance and role.

22.12.7 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 completed`.

3. An approval task is generated that is again assigned to the System Administrator.

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.

22.12.8 Requesting for Role With an Access Policy Attached

If the role is requested by an end-user and the request is for multiple roles, then SoD Check is first be done for the roles. After the request is approved and the role is assigned to the user, run the Evaluate User Policies schedule job to evaluate the access policy. Then, account provisioning is triggered. This again results in SoD Check for the account being provisioned if child data is requested for. SoD Check result can be seen in the Account Details page. If account is provisioned without child data, then no SoD Check is done.

Note: See "Predefined Scheduled Tasks" in *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager* the for information about the Evaluate User Policies schedule job.

22.12.9 Provisioning Based on Access Policies Without Approval

To perform provisioning based on access policies:

1. Create a new role.

2. Create an access policy (without approval) 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. Run the Evaluate User Policies scheduled job to trigger the access policy and provision the account on the target system. But entitlement provisioning will wait for SoD check.

4. Check the Account Details to verify the SoDCheckStatus field value. If the SoD check is successfully completed, then the value of the SoDCheckStatus field is `SoD Check Completed`, and the SoDChecker task will be in Completed state.

   The Holder task status depends on the SoD Check result. If the SoD check passes, then the Holder task is completed and entitlements are provisioned. Otherwise, the Holder task is canceled and no entitlement provisioning takes place.

22.12.10 Provisioning Based on Access Policies With Approval

When an access policy is created with approval, request is created for account provisioning. After the role is assigned to the user and the Evaluate User Policies schedule job is run, a request is created to provision account to the user. SoD Check is done for this request.

22.12.11 Requesting for Entitlements From Two Application Instances

A request can be raised for two entitlements from different application instances, one for which Sod Check is enabled and other for which it is not enabled. Here, SoD Check is performed for the entitlement for which SoD Check is enabled.

Note: SoD Check is not supported between entitlements from different application instances. For example, SoD Check is not supported between an EBS and a PSFT entitlement.

22.13 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_SERVER*/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.

22.14 Troubleshooting SoD Check

Table 22–2 lists the troubleshooting steps that you can perform if you encounter errors while performing SoD check.

Table 22–2 Troubleshooting SoD Check

Problem	Solution
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.	This means that SoD configuration is incorrect. Check if the Segregation of Duties (SOD) Check Required system property is set to `true`. If yes, then check the value of topologyName in connector IT resource field. If default registration is used, then the value of the topologyName parameter is `sodoaacg` for OAACG SoD engine and `sodgrc` for SAP GRC. If registration is done manually, then check if the corresponding topology is defined in the SILConfig.xml file and this file is seeded into MDS after the change.
The SoDCheckStatus field in the Request form displays `Sod check completed with error`. The SoDCheckResult field displays `Error from SoD Engine`.	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: ▪ 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. Check the SoD engine log for further errors. If no tracking ID is returned by the SoD engine, then simulation is not started successfully.
The SoDCheckStatus field in the process form is in the `SoD Result Pending` status, and does not move to the `SoD Check Completed` status even on running the scheduled job.	Make sure that you run the Get SoD Check Results Provisioning scheduled job and not the scheduled job for approval. Make sure that the scheduled job is triggered. You may enable logging at DEBUG level to confirm this. 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.
When requesting for SoD-enabled resource, no SoD fields are displayed in the Request form after creating the request, and the request directly moves to the request-level approval.	This error means that SoD configuration is incorrect. Check if the Segregation of Duties (SOD) Check Required system property is set to `true`. If yes, then check the value of topologyName in the connector IT resource field. If default registration is used, then the value of the topologyName parameter must be `sodoaacg` for the OAACG SoD engine and `sodgrc` for SAP GRC. If registration is performed manually, then check if the corresponding topology is defined in the SILConfig.xml file and this file is seeded into MDS after the change.
The SoDCheckStatus field in the Request form stays in the `SoD Result Pending` status and does not move to the `SoD Check Completed` status even on running the scheduled job.	Make sure that you run the Get SoD Check Results Approval scheduled job and not the scheduled job for approval. Make sure that the scheduled job is triggered. You may enable logging at DEBUG level to confirm this. 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.
The SoD check is successfully performed during request provisioning, but the resource state in the user profile does not display as `Provisioned`. Therefore, the request is in the Operation Initiated status.	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. 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. 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.

Table 22–2 (Cont.) Troubleshooting SoD Check

Problem	Solution
The SoD check is successfully performed during direct provisioning, but the resource state in the user profile is not Provisioned.	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. 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.
Request provisioning has been successfully done and appropriate values are displayed in the Request form, but the SoD status and result are not reflected to the Account form.	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 form to the Account form.
A particular SoD request is being tried several times and generating error.	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: 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.
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 "SYSTEM 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.	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.

Table 22–2 (Cont.) Troubleshooting SoD Check

Problem	Solution
When trying to perform SoD check by using the DefaultSODApproval workflow, the following error message is displayed: `Unknown Credential type to find the password for the given map : oim key : sodcheck.credentials`	Add `sodcheck.credentials` as described in step 5 of "Enabling SoD" on page 22-9.
The following error is displayed: `[exec] Caused By: Thor.API.Exceptions.tcITResourceNotFoundException [exec] at com.thortech.xl.ejb.beansimpl.tcITResourceInstanceOperationsBean.getITResourceInstanceParametersData`	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.

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.	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 registeration 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:
	`<IdmId>1</IdmId>`
	`<SodId>7</SodId>`
	`<SDSId>6</SDSId>`
	
	Then the IDs returned by the registration script are:
	`1 oimInstance`
	`6 oimSDSInstance`
	`7 oiaInstance`
	
	If these are different, then change the IDs in the SILConfig.xml file and reimport it by using the MDS utility.

Part V

Data Synchronization

This part contains chapters that describe customizing reconciliation and developing LDAP containers and scheduled tasks.

It contains the following chapters:

23

Customizing Reconciliation

This chapter describes reconciliation features and architecture and the various aspects of customizing reconciliation operations in the following sections:

- Reconciliation Features
- Reconciliation Architecture
- Defining Reconciliation Rules
- Developing Reconciliation Scheduled Tasks
- Updating Reconciliation Profiles Manually
- Understanding Reconciliation APIs
- Postprocessing for Trusted Reconciliation
- Troubleshooting Reconciliation
- Populating Data in the RECON_EXCEPTIONS Table
- Reconciliation Best Practices
- Monitoring Reconciliation Performance Using DMS

23.1 Reconciliation Features

Reconciliation features can be divided into the following categories:

- Performance Enhancement Features
- Web-Based Event Management Interface
- Other Features

23.1.1 Performance Enhancement Features

The following features help increase performance during reconciliation:

- New Metadata Model - Profiles
- Parameters to Control Flow and Processing of Events
- Grouping of Events by Reconciliation Runs
- Grouping of Events by Batches
- Implementing Reconciliation Engine Logic in the Database
- Improved Java Engine

- Improved Database Schema

23.1.1.1 New Metadata Model - Profiles

If metadata is associated with a reconciliation target, then it limits the ability to run multiple jobs performing different types of reconciliation against the same target. Therefore, all configurations in various components of Oracle Identity Manager are stored centrally in an XML store called MDS.

For backward compatibility, current deployments continue managing their configurations through Oracle Identity Manager Design Console and the configuration continues to be stored in the Oracle Identity Manager database. The configuration APIs automatically read the configurations from the tables in Oracle Identity Manager and convert them into XML profiles, called default profiles, and associate those profiles with the existing reconciliation runs.

You manage all the metadata by using Oracle Identity Manager Design Console. Using Oracle Identity Manager Design Console, you can generate the default reconciliation profile. This can be used to regenerate the profile when reconciliation configurations are changed from Oracle Identity Manager Design Console. When configurations are imported from the Deployment Manager, the profile is generated by default.

All nondefault profiles can be completely managed by using any XML editor.

> **See Also:** "Reconciliation Profile" on page 23-8 for information about reconciliation profiles

23.1.1.2 Parameters to Control Flow and Processing of Events

This section consists of the following topics:

- Parameters to Control Event Processing
- System Property to Control AutoRetry

Parameters to Control Event Processing

BatchSize is the parameter to control event processing. This dictates the size of the batch. A batch size of 1 is equivalent to processing of events one at a time. Batch size is available as a system property and can be managed from Oracle Identity Manager Design Console. The property name is OIM.ReconBatchSize. The default value of the system BatchSize parameter is 500. For information about system properties, see "Managing System Properties" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager*.

System Property to Control AutoRetry

The Retry Count for recon event system property controls auto retry by indicating how many times an item needs to be retried before the reconciliation engine marks it as an error or sends it to manual queue. The value 0 for this property means that the auto retry option is not configured.

> **See Also:** "Handling of Race Conditions" on page 23-5 for more information about auto retry

23.1.1.3 Grouping of Events by Reconciliation Runs

All the events created in the reconciliation database are grouped by reconciliation runs. All events in a reconciliation run are grouped with a common reconciliation run ID. Because each reconciliation run is associated with a profile, all events in a reconciliation run are processed by using the same profile. This helps in optimizing the

performance because the configurations have to be retrieved only once per reconciliation run.

Each profile can use a different batch size. This enhances system performance for each target reconciliation by tuning the appropriate batch for it.

23.1.1.4 Grouping of Events by Batches

Batches are introduced to increase system performance during reconciliation. A batch consists of a number of events. It is a unit of processing in the reconciliation engine. The size of the batch is configurable. Reconciliation runs are broken into fixed size batches. For example, if a reconciliation run consists of 9900 events and batch size is 1000, then that reconciliation run is divided into 10 batches each with size 1000, and last batch with size 900.

Processing a batch as a unit optimizes system performance by eliminating the overhead of processing one event at a time. This also allows performing bulk operations wherever possible. Batches can also run in parallel to balance the use of hardware resources.

23.1.1.5 Implementing Reconciliation Engine Logic in the Database

In earlier releases, all engine logic was implemented in Java and the processing happened one event at a time. In 11*g* Release 2 (11.1.2.2.0), most of the logic to process the events is implemented as stored procedures. A combination for processing at batch level and the logic being implemented in PLSQL makes it possible to perform bulk operations at the SQL layer. The following steps are performed in bulk (one batch at a time):

- Required data check
- Applying matching rules
- Applying action rules

23.1.1.6 Improved Java Engine

Processing that cannot be performed in stored procedures and must be performed in Java layer also provides better performance than earlier releases of the engine for the following reasons:

- Java engine performs bulk operations by default:
  - Submits events in batches to the database
  - Submits bulk postprocess orchestration depending on the action
- Performs bulk operations wherever possible.

23.1.1.7 Improved Database Schema

A notable performance enhancement from the new database schema in 11*g* Release 2 (11.1.2.2.0) is by using horizontal tables for storing event details for various targets instead of using a single vertical table for storing the event details from various targets. A horizontal table is used for each profile.

> **See Also:** "Staging Tables" on page 23-4 for more information about horizontal tables

23.1.2 Web-Based Event Management Interface

Oracle Identity Manager provides a Web-based event management interface that allows you to manage the events from the Web. Authorized users are able to search for events, users, and handle exceptions by linking events with users and accounts. You can also close events, force failed events to be re-evaluated, and perform ad-hoc linking.

Ad-hoc linking refers to the ability provided to authorized users of the Event Management section to link an event to any user in Oracle Identity Manager. Although the reconciliation engine finds user matches for events, the user through this ad-hoc link feature can ignore those matches and select a different user. This allows you to handle exceptions resulting from error matches.

> **See Also:** "Managing Reconciliation" in *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager* for information about the tasks performed to manage reconciliation events

23.1.3 Other Features

Other reconciliation features are described in the following sections:

- Staging Tables
- Handling of Race Conditions
- Ad Hoc Linking

23.1.3.1 Staging Tables

In earlier releases of Oracle Identity Manager, the reconciliation schema has one table to store all the event details from various targets. The list of attributes and their names and types that the various reconciliation events contain can vary from target to target. This means that events from one target can contain a different set of data compared to events from another target. The only way to store data from such events in a single table is by storing one attribute per row. Therefore, in earlier releases, each row in the event detail table represents a single attribute of reconciliation event data. For each attribute, it stores the event to which it belongs, the attribute name, type, and value. This is also referred to as vertical table in this document. Although vertical tables are beneficial from the point of view of flexibility and extensibility, it is not an efficient way to store event records from the performance prospective.

Storage in vertical tables is replaced by separate tables for each target, called horizontal tables or staging tables. They are called horizontal tables because instead of storing attributes of an event vertically in the table as rows (as many rows as there are number of attributes), the attributes of an event are stored as columns. This means that there are as many columns as there are number of attributes for a target. Each event is stored as a row. Because different targets can have different sets of attributes, each target has a separate table in the reconciliation schema to store event details. There can be multiple tables per target because of requirements to handle multi-valued attributes that are stored as rows in child tables.

Each row of the event detail table for a specific profile stores the list of reconciliation fields for a single event. For example, for trusted user reconciliation in which firstname, lastname, email attributes are being reconciled, there is the RA_ XELLERATE_USER staging table with the following columns:

RE_KEY, RECON_FIRSTNAME, RECON_LASTNAME, RECON_EMAI

Creating and Maintaining Staging Tables

Staging tables can be created only when a target is being deployed against Oracle Identity Manager. This is because, at the time of target deployment, the reconciliation system knows the list of attributes and their types for the target, which needs to be reconciled.

Staging tables are updated when configurations are imported from the Deployment Manager or changes are made by using Oracle Identity Manager Design Console. To generate a staging table from Oracle Identity Manager Design Console, in the Object Reconciliation form, click **Generate Reconciliation Profile**.

23.1.3.2 Handling of Race Conditions

In earlier releases of Oracle Identity Manager, when an event is being reconciled, the reconciliation engine may not be able to process it successfully because before this event can be reconciled, another event needs to be reconciled. For example, before the reconciliation engine can reconcile an event that is supposed to create an account, the engine needs to reconcile an event that is supposed to create a user. This is called a race condition.

In Oracle Identity Manager 11*g* Release 2 (11.1.2.2.0), the race conditions are handled by setting the value of the 'Retry Count for recon event' system property. To configure auto retry, specify a value greater than 0 for this property. If you do not want to configure auto retry, then specify 0 as the value of the Retry Count for recon event system property.

When auto retry is configured, the reconciliation engine checks for the race conditions. If a race condition is found, then the reconciliation engine puts the reconciliation event in a re-evaluate queue until the retry count is exhausted.

A Reconciliation Retry Scheduled Task periodically checks if there is any event waiting for retry and is ready to be re-evaluated and if yes, it queues them up for reconciliation engine processing. This scheduled task is configured by default.

> **Note:** If the auto retry count is exhausted, the reconciliation engine does not further process the event and sets the status per the matching rules. However, you can manually retry by requesting for re-evaluate from Event Management. For information about re-evaluating events, see "Re-evaluating Events" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager*.

Auto retry can handle the following race conditions:

- An account event for creating an account in Oracle Identity Manager is processed before the user is created for this event because the event for creating user is not processed yet.
- A user event for creating a Xellerate user in Oracle Identity Manager is processed before the organization is created to which this user belongs.

All auto retry parameters are stored as part of the reconciliation profiles. This means that while the events belonging to one reconciliation run may have auto retry configured, the events belonging to another reconciliation run may not have auto retry configured.

In Oracle Identity Manager, there is no UI to manage these parameters within a profile and you must use an XML editor to manage them by directly editing the XML profile. For information about editing an XML profile, see "Creating and Updating

Reconciliation Profiles" on page 23-24.

23.1.3.3 Ad Hoc Linking

If the reconciliation engine is not able to determine the owner based on the matching rules, then you can manually link an account to a user by using Oracle Identity Manager Advanced Administration. Subsequent modifications to the account is automatically linked to that account.

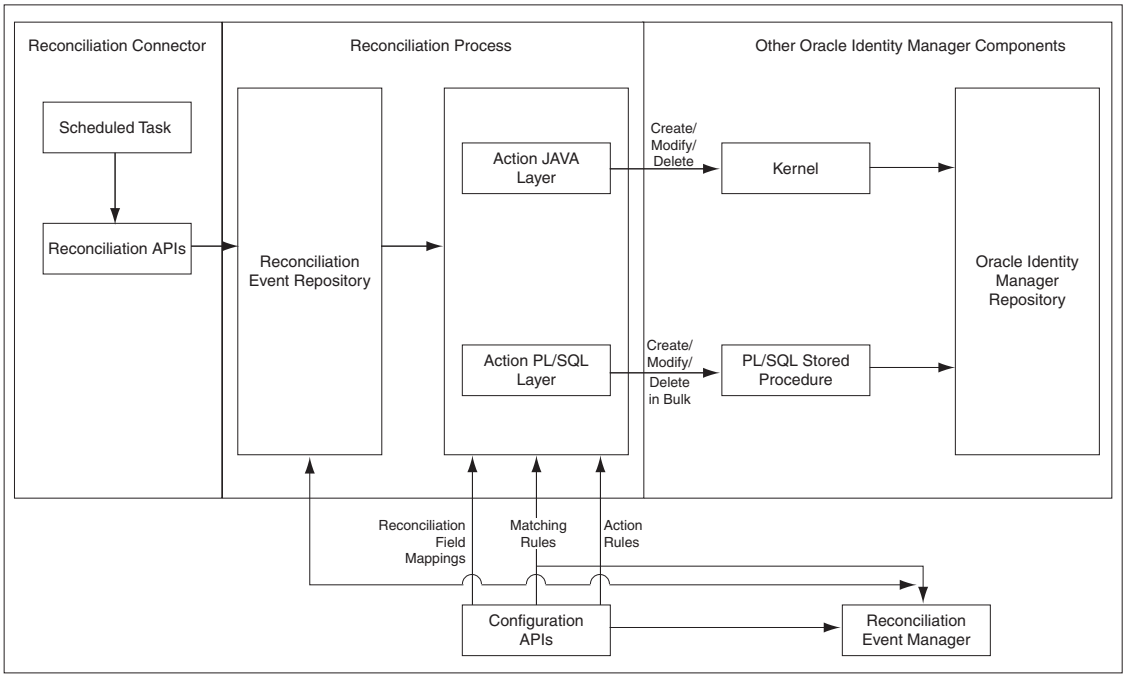
Ad hoc linking is supported for user and account events. If the reconciliation engine is not able to determine the owner based on the matching rules, then you can manually link a user or account event to a user.

> **See Also:** "Ad Hoc Linking" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager* for information about how to perform ad hoc linking

23.2 Reconciliation Architecture

Reconciliation is the process of pulling entity data from the target system into Oracle Identity Manager to keep the entity data in a consistent state between the two systems. The various components of Oracle Identity Manager involved in reconciliation and the interaction between these components are shown in the Figure 23–1:

Figure 23–1 Reconciliation Architecture



The reconciliation architecture is described in the following steps:

1. Each connector has scheduled tasks associated with it. The scheduler triggers the connector scheduled task, which invokes reconciliation APIs to generate events. The event can be of type Regular, Changelog, or Delete.

   For more information about the scheduler, see "Managing the Scheduler" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager*. For more information about scheduled tasks, see "Connector for Reconciliation" on

page 23-16.

2. The reconciliation events are stored in the reconciliation event repository, which is Oracle Identity Manager database.
3. When batch size is met, an asynchronous message is submitted which processes the batch of events in bulk. At the end of the schedule task another asynchronous message is submitted for processing the events of the last batch.

Note:

- In Figure 23–1, the reconciliation engine encapsulates the Action JAVA Layer as well as parts of the Reconciliation Event Repository, and orchestrates all the arrows in that diagram.
- In this release, trusted source reconciliation is supported for users only. It is not supported for roles, role membership, and role hierarchy reconciliation.
- In this release, Oracle Identity Manager supports trusted source reconciliation and account reconciliation for organizations.

4. The processing involves data validation, matching of the entities and action (create, update, delete and so on). This is followed by post processing via kernel orchestrations. For information about the action module, see "Action Module" on page 23-15. For information about the reconciliation profile, see "New Metadata Model - Profiles" on page 23-2.
5. By default the reconciliation event processing happens in bulk, and therefore all the steps till post processing are performed by PL/SQL stored procedures. Event can be processed one at a time in the following scenarios (in this case all the steps till matching are done in PL/SQL and the action is performed in java layer):
   - When events are processed from the Event Management UI
   - When failed events are retried by the retry scheduled task that runs periodically

   For reconciliation single event processing, actions and post processing take place through the kernel.
6. Reconciliation events are made available to the Event Management UI by another API call in the reconciliation management service.

The functionality of various components of the reconciliation service are explained in the following sections:

- Reconciliation Profile
- Reconciliation Metadata
- Reconciliation Target
- Reconciliation Run
- Reconciliation APIs
- Reconciliation Schema
- Reconciliation Engine
- Connector for Reconciliation
- Archival

- Backward Compatibility
- Reconciliation Event Management

23.2.1 Reconciliation Profile

A reconciliation profile is the configuration defined to govern how reconciliation is run for a particular resource. A particular resource can have multiple reconciliation profiles, each of which defines matching rules, action rules, and field mappings, which can differ in each profile corresponding to the resource. For example, while one reconciliation run can perform reconciliation of new and modified accounts, another reconciliation run can reconcile deletion of accounts because you might want to run the deletions only once a day. In this example, you define two reconciliation runs and two profiles. Each profile is associated with respective reconciliation run and each profile having its own rules of reconciliation.

The profile is an XML-based configuration file stored in Oracle Identity Manager MetaData Store (MDS). Example 23–1 shows a sample reconciliation profile:

Example 23–1 Sample Reconciliation Profile

```
<?xml version='1.0' encoding='UTF-8'?>
<profile xmlns="http://www.oracle.com/oracle/iam/reconciliation/config" ownerType="User"
changeType="CHANGELOG" auditEnabled="true" batchSize="500" resourceType="Account" name="Modified AD
User" configure="true" active="true">
   <matchingRule>((UPPER(USR.usr_udf_obguid)=UPPER(RA_ADUSERE469E5C8.RA_
OBJECTGUID)))</matchingRule>
   <form oimTableName="UD_ADUSER" stagingTableName="RA_ADUSERE469E5C8" name="Modified AD User"
mlsOimTable="mlsOIMTableIfAny" mlsStagingTable="mlsStagingTableIfmlsOIMTable">
      <matchingRule>(UD_ADUSER.UD_ADUSER_OBJECTGUID=RA_ADUSERE469E5C8.RA_OBJECTGUID)</matchingRule>
      <targetAttributes>
         <targetAttribute type="String" name="Status">
            <stagingField type="String" length="256" name="RA_STATUS"/>
         </targetAttribute>
 <targetAttribute type="String" name="copyStatus" ref="Status" mls="true">
            <stagingField type="String" length="256" name="COPY_STATUS"/>
            <oimAttribute type="String" fieldName="OIM_OBJECT_STATUS" fieldType="String" name="OIM_
OBJECT_STATUS"/>
         </targetAttribute>
         <targetAttribute type="String" name="password" encrypted="true" keyField="false"
required="false">
            <stagingField type="String" length="256" name="PASSWORD"/>
            <oimAttribute type="String" fieldName="UD_ADUSER_PASSWORD" fieldType="String" name="AD
Password"/>
         </targetAttribute>
         <targetAttribute type="Date" name="accountExpires">
            <stagingField type="Date" name="RA_ACCOUNTEXPIRES"/>
            <oimAttribute type="Date" fieldName="UD_ADUSER_DATE" fieldType="Date" name="Account
Expiration Date"/>
         </targetAttribute>
         <targetAttribute type="ITResource" name="IT Resource" keyField="false">
            <stagingField type="ITResource" length="19" name="RA_ITRESOURCE15641F83"/>
            <oimAttribute type="Number" fieldName="UD_ADUSER_AD" fieldType="Number" name="AD
Server"/>
         </targetAttribute>
 <targetAttribute type="String" keyField="true" name="objectGUID">
            <stagingField type="String" length="32" name="RA_OBJECTGUID"/>
            <oimAttribute type="String" fieldName="UD_ADUSER_OBJECTGUID" fieldType="String"
name="Object GUID"/>
         </targetAttribute>
```

```
      </targetAttributes>
      <form oimTableName="UD_ADUSRC" stagingTableName="RA_ADUSERGROUPDETA902DB909" name="memberOf">
         <matchingRule>(UD_ADUSRC.UD_ADUSRC_GROUPNAME=RA_ADUSERGROUPDETA902DB909.RA_
MEMBEROF)</matchingRule>
         <targetAttributes>
            <targetAttribute type="String" keyField="true" name="memberOf">
               <stagingField type="String" length="256" name="RA_MEMBEROF"/>
               <oimAttribute type="String" fieldName="UD_ADUSRC_GROUPNAME" fieldType="String"
name="UD_ADUSRC_GROUPNAME"/>
            </targetAttribute>
         </targetAttributes>
      </form>
   </form>
   <actionRules>
      <actionRule condition="One Entity Match Found" action="Establish Link"/>
   </actionRules>
</profile>
```

Table 23–1 describes the elements and the structure of the reconciliation profile XML file.

Table 23–1 Elements in the Reconciliation Profile XML

Element Level 1	Sub-element Level 2	Sub-element Level 3	Sub-element Level 4	Sub-element Level 5	Description
<profile>					The root element or object of the reconciliation configuration profile.
	<ownerType>				Populated only for role hierarchy, role membership, and account with values Role, Role, and User respectively.
	<changeType>				By default, or if the element is not present, then the value is CHANGELOG. Otherwise, the value can be REGULAR, CHANGELOG, or DELETE.
	<auditEnabled>				Used with account type profile only. By default or if the element does not exist, then value is false, and audit for the resource object is stopped.
	<batchSize>				Changes the size or number of reconciliation events per batch. By default, or if the element is not present, then batch size is 500.
	<resourceType>				Value can be any one of Account, User, Role, RoleRole, RoleUser, and Organization.
	<name>				This is the resource object name.
	<configure>				By default or if the element is not present, then the value is false. If reconciliation configuration is to be created or updated on a system, then this must be marked as true. After all manual corrections of a profile, this attribute must be marked as true. For test to production, mark this element as true before importing into target system.
	<active>				By default or if the element is not present, then the value is true. Value is false for corrupt or invalid profiles and marks profile unusable. Such profiles are never loaded into the system. After all manual corrections of a profile, this attribute must be removed or marked as true.

Table 23–1 (Cont.) Elements in the Reconciliation Profile XML

Element Level 1	Sub-element Level 2	Sub-element Level 3	Sub-element Level 4	Sub-element Level 5	Description
	<matchingRule>				Populated only for role hierarchy, role membership, and account with owner matching rule. Otherwise, the element is not present.
	<form>				This specifies one parent form per profile.
		<oimTableName>			Oracle Identity Manager table into which data will be reconciled.
		<stagingTableName>			Staging table into which data from the target system is stored before processing.
		<name>			Same as profile name for the parent form and same as multivalued attribute name for the child forms.
		<mlsOimTable>			Multilanguage supported (MLS) Oracle Identity Manager table into which data will be reconciled if resource object is MLA-enabled.
		<mlsStagingTable>			MLS staging table into which data from target system is stored before processing if resource object is MLS-enabled.
		<matchingRule>			Matching rule for the form (resource object associated with the profile), and is always required.
		<targetAttributes>			Groups all target attributes.
			<targetAttribute>		One for each attribute from the target system.
				<type>	Data type of the target attribute.
				<keyfield>	By default, the value is false. Used in matching rule for account resource type.
				<name>	Name of the attribute from the target system provided by the connector that starts reconciliation.
				<required>	If the attribute is required, then this element must be present.
				<encrypted>	If the value is true, then the attribute value will be encrypted and stored in staging and Oracle Identity Manager tables.
				<ref>	Name of the target attribute in the same form whose value will be copied and stored in this attribute.
				<stagingField>	Specifies the column of the staging table corresponding to the target attribute. This contains the following elements: **<type>:** data type of the staging table column. **<length>:** length/size of the staging table column/field. **<name>:** name of the staging table column.

Table 23–1 (Cont.) Elements in the Reconciliation Profile XML

Element Level 1	Sub-element Level 2	Sub-element Level 3	Sub-element Level 4	Sub-element Level 5	Description
				<oimAttribute>	Specifies the mapped Oracle Identity Manager domain attribute name. The element is present only if the target attribute is mapped. This contains the following elements: **<name>:** Oracle Identity Manager attribute name **<type>:** Oracle Identity Manager attribute type **<fieldName>:** Column name of the Oracle Identity Manager table corresponding to the Oracle Identity Manager mapped attribute **<fieldType>:** Column type of the Oracle Identity Manager table corresponding to the Oracle Identity Manager mapped attribute
		<form>			Specifies child form or forms for the parent or root form. It corresponds to a multivalued attribute.
			<matchingRule>		Matching rule for a child form.
			<targetAttributes>		This is the same element as the parent <targetAttributes> element. This element can be nested several times, for example, <form><targetAttributes><form><targetAttributes>.
	<actionRules>				Groups all action rules for the resource object.
		<actionRule>			An actionRule element for each action rule.
			<condition>		The value can be any one of No Matches Found, One Entity Match Found, Multiple Entity Matches Found, One Process Match Found, Multiple Process Matches Found.
			<action>		Can be anything based on the profile XSD.

There is always a default profile associated with reconciliation configurations for any resource object. The default profile can be explicitly generated from Oracle Identity Manager Design Console in the developer's environment or implicitly generated during import from the Deployment Manager. For details on how to create and update profiles, see "Updating Reconciliation Profiles Manually" on page 23-23.

23.2.2 Reconciliation Metadata

The reconciliation metadata consists of various configurations used in creating and processing the reconciliation events. The reconciliation metadata is stored in a logical container called a profile. For information about reconciliation profile, see "Reconciliation Profile" on page 23-8.

Examples of the reconciliation metadata are:

- **Mapping rules:** Used to map the data received from the target system to the data managed about that target system in Oracle Identity Manager.

- **Matching rules:** Used during the processing of each reconciliation event to correlate the event data to a particular account, user, or role in Oracle Identity Manager.
- **Action Rules:** Used to specify the actions taken by Oracle Identity Manager based on the result of the processing of a reconciliation event.
- **List of target attributes:** Used to define the data attributes received from the target system via reconciliation. It is used in the mapping rules, and is configured by using Oracle Identity Manager Design Console.

The various configurations used in creating and processing the reconciliation events are managed by using Oracle Identity Manager Design Console, and for backward compatibility, is stored in the same Oracle Identity Manager tables as in Oracle Identity Manager release 9.1.0. In addition, the configurations are also stored in the reconciliation profile.

Note: For reconciliation in Oracle Identity Manager, a metadata model is being used. See "Managing Reconciliation Events" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager*.

23.2.3 Reconciliation Target

Reconciliation target refers to an instance of an application that acts as a source of changes for Oracle Identity Manager. An example of reconciliation target is an HR system, which acts as a source of identities for Oracle Identity Manager. A reconciliation target can be a source of users or accounts.

23.2.4 Reconciliation Run

Reconciliation run refers to the combination of a reconciliation connector and associated configurations which when run by the scheduled task, performs the reconciliation based on the rules defined in the associated configurations. The scheduler runs reconciliation periodically at fixed intervals. Reconciliation runs are scheduled within Oracle Identity Manager scheduler to run at a specified frequency. All events created during a reconciliation run are grouped together by a unique reconciliation run ID.

23.2.5 Reconciliation APIs

These are a set of published APIs to provide reconciliation data to Oracle Identity Manager in the form of reconciliation events. Connectors can use the APIs to push data to the reconciliation event repository. Scheduled tasks can be setup to run the APIs when reconciliation is to be run on a scheduled basis. The existing connectors do not need to be changed because the existing APIs are supported.

23.2.6 Reconciliation Schema

The data that comes from the target system for reconciliation is stored in the reconciliation schema. The data contains the changes to be reconciled with Oracle Identity Manager.

Reconciliation schema refers to the set of schema tables to store the reconciliation data. The reconciliation schema is redesigned for performance reasons and future extensibility. See "Improved Database Schema" on page 23-3 for more information about the reconciliation schema.

23.2.7 Reconciliation Engine

The reconciliation engine uses all configurable components and includes the data processor and rule evaluator that use these components to convert input data into a list of action items. It also includes the components that determine whether or not the actions can be automated based on the rule context. When an action is performed, either automatically or manually, the engine performs the appropriate updates and provisioning actions.

The main task of the reconciliation engine is to perform the comparison, determine the action to be taken, and apply the action in Oracle Identity Manager. It contains two modules, which are described in the following sections:

- Matching Module
- Action Module

23.2.7.1 Matching Module

The matching rule specified in the profile is used to identify whether the record being searched, exists in Oracle Identity Manager or not. Matching rules are rules to identify whether the data is for an identity that Oracle Identity Manager already has a record of, or to identify the owner of the account in Oracle Identity Manager.

For account entities, when no record is found, an owner match is then performed to identify the owner of the account.

For role hierarchy events, matching is performed to identify the parent and child role.

Note: While performing role hierarchy and role membership reconciliation, the matching criteria must contain both Namespace and Role Name in the matching criteria. The following is an example of a matching rule:

((UGP.ugp_rolename=x) and (UGP.ugp_namespace=y))

Here, x is the name of the staging table name column that is mapped to Role Name, and y is the name of the staging column that is mapped to Namespace.

At the end of the evaluation, the match table contains all the possible matches found within Oracle Identity Manager that meet the criteria for the event, and the state of the event is updated to one of the statuses listed in Table 23–2:

Table 23–2 Reconciliation Status Events

Status Events	Description
Data Received	Event data has been created in the database and is ready for further processing.
Event Received	A reconciliation event has been created and is ready for further processing. The finishReconciliationEvent API has not yet been called.
Data Validation Failed	The reconciliation event record is invalid. For example, a role event with an invalid role category will fail to validate. This situation could indicate a race condition.
Data Validation Succeeded	The event data was successfully validated and the event can now safely be processed by the Engine.

Table 23–2 (Cont.) Reconciliation Status Events

Status Events	Description
Multiple Accounts Match Found	Given the current matching rules, multiple matching account records were found for the data.
No Account Match Found	Given the current matching rules, no matching account records were found for the data.
Single Account Match Found	Given the current matching rules, one matching account record was found for the data.
Multiple Org Matches Found	Given the current matching rules, multiple matching organization records were found for the data.
No Org Match Found	Given the current matching rules, no matching organization records were found for the data.
Single Org Match Found	Given the current matching rules, one matching organization record was found for the data.
Multiple Role Grants Match Found	Multiple matching records for user membership within a role were found.
No Role Grant Match Found	No matching records for user membership within a role were found.
Single Role Grant Match Found	One matching record for user membership within a role was found.
Multiple Roles Match Found	Given the current matching rules, multiple matching role records were found for the data.
No Role Match Found	Given the current matching rules, no matching role records were found for the data.
Single Role Match Found	Given the current matching rules, one matching role record was found for the data.
No Role Members Found	The Reconciliation Engine did not find role members matching the data, given the current matching rules.
No Role Parent Found	The Reconciliation Engine did not find a role matching the data, given the current matching rules.
Multiple Role Relationships Match Found	Given the current matching rules, reconciliation has found multiple role-to-role relationships that match data in the event.
No Role Relationship Match Found	Given the current matching rules, reconciliation did not find any role-to-role relationships that match data in the event.
Single Role Relationship Match Found	Given the current matching rules, reconciliation has found one role-to-role relationship that matches data in the event.
Multiple Users Match Found	Given the current matching rules, multiple matching user records were found for the data.
No User Match Found	Given the current matching rules, no matching user records were found for the data.
Single User Match Found	Given the current matching rules, one matching user record was found for the data.
Invalid Event Data Passed	The event contains invalid data.
Being Re-evaluated	The reconciliation event is being re-evaluated from the reconciliation event management UI.
Being Re-tried	The reconciliation event is being retried automatically. This status event has been deprecated.
Creation Failed	The user/account/role entity was not created successfully.

Table 23–2 (Cont.) Reconciliation Status Events

Status Events	Description
Creation Succeeded	The user/account/role entity was created successfully.
Delete Failed	The user/account/role entity was not successfully deleted.
Delete Succeeded	The user/account/role entity was deleted successfully.
Event Closed	The reconciliation event was closed from the reconciliation event management UI. The change is complete.
Update Failed	The user/account/role entity was not updated successfully.
Update Succeeded	The user/account/role entity was updated successfully.

23.2.7.2 Action Module

This module applies the action based on the event state, entity type, and the action rules, as listed in Table 23–3:

Table 23–3 Action Rules

Event State	Entity Type	Action	Description
No User Match Found	User	None	Does not perform any action
		Create User	Creates a user in Oracle Identity Manager
No Account Match Found	Account	None	Does not perform any action
User Matched	User or Account	None	Does not perform any action
	User	Establish Link	Modifies or deletes the matched user based on the change type
	Account	Establish Link	Owner identified - creates an account
Users Matched	User or Account	None	Does not perform any action
Account Matched	Account	None	Does not perform an action
		Establish Link	Modifies or revokes the account based on the change type
Accounts Matched		None	Does not perform any action
No Role Match Found	Role	None	Does not perform any action
Single Role Match Found	Role	None	Does not perform an action
		Establish Link	Modify or delete a role
	Role Membership	Create role membership	Grant a role member to Oracle Identity Manager
		Delete role membership	Delete a role member from Oracle Identity Manager
		None	Does not perform an action
	Role Hierarchy	Create role hierarchy	Creates a role hierarchy in Oracle Identity Manager
		Delete role hierarchy	Delete a role hierarchy in Oracle Identity Manager

Table 23–3 (Cont.) Action Rules

Event State	Entity Type	Action	Description
		None	Does not perform an action
Multiple Roles Matched	Role, Role membership and Role Hierarchy	None	Does not perform an action
No Role Grant Match Found	Role Membership	None	Does not perform an action
		Create Role Member	Creates a role member in Oracle Identity Manager
Single Role Grant Match Found	Role Membership	None	Does not perform an action
		Establish Link	Delete role member
Multiple Role Grant Match Found	Role Membership	None	Does not perform an action **Note:** This state does not occur because the role grant match is done by looking for the primary key, which is a combination of the usr key and the group key.
No Role Parent Match Found	Role Hierarchy	None	Does not perform an action
		Create role parent	Create a role parent in Oracle Identity Manager
Single Role Parent Match Found	Role Hierarchy	None	Does not perform an action
		Establish Link	Delete role parent
Multiple Role Parent Match Found	Role Hierarchy	None	Does not perform an action
Data Validation Failed	Role, Role Hierarchy, Role Member	Race condition	Does not perform an action. The event needs to be re-evaluated.
Parent role not found	Role Hierarchy	Race condition	Does not perform an action. The event needs to be re-evaluated.
Role member not found	Role membership	Race condition	Does not perform an action. The event needs to be re-evaluated.

23.2.8 Connector for Reconciliation

The connector refers to the software that extracts the changes from the target system and creates events in the reconciliation schema by calling the reconciliation APIs. If the connector that you want to use is shipped with a predefined reconciliation module, then a scheduled task definition is available. You use this component to control the frequency at which the target system is polled for changes to track data and other connector-specific parameters.

The connector for reconciliation is deployed by using the Deployment Manager. When the connector is deployed, the corresponding reconciliation profile for that connector is created in the metadata store (MDS), and horizontal tables that store the event data are also created.

Note: Do not manually update reconciliation profile or update any reconciliation configurations from the Deployment Manager or Oracle Identity Manager Design Console when a reconciliation run is still in progress. This is because, if a reconciliation field is deleted or updated when a reconciliation run is in progress, then the event data might not be valid any more.

For information about configuring connectors, see Oracle Identity Manager Connector documentation.

See Also:

- "Reconciliation Metadata" on page 23-11 for information about MDS
- "Staging Tables" on page 23-4 for information about the staging tables

23.2.9 Archival

The Reconciliation Archival utility allows you to move processed events from the active reconciliation tables to archive tables. The events to move can be selected based on a time range. Only linked and closed events, which means successfully processed or closed by an administrator, can be archived.

See Also: "Using the Reconciliation Archival Utility" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager* for information about how to use the Reconciliation Archival utility

23.2.10 Backward Compatibility

You do not need to change the existing reconciliation configurations or scheduled tasks to leverage the new reconciliation service.

The existing configurations for reconciliation setup in earlier Oracle Identity Manager releases continues to function after upgrading to 11*g* Release 2 (11.1.2.2.0). As part of the upgrade, corresponding reconciliation event tables are created for each of the existing object types being reconciled.

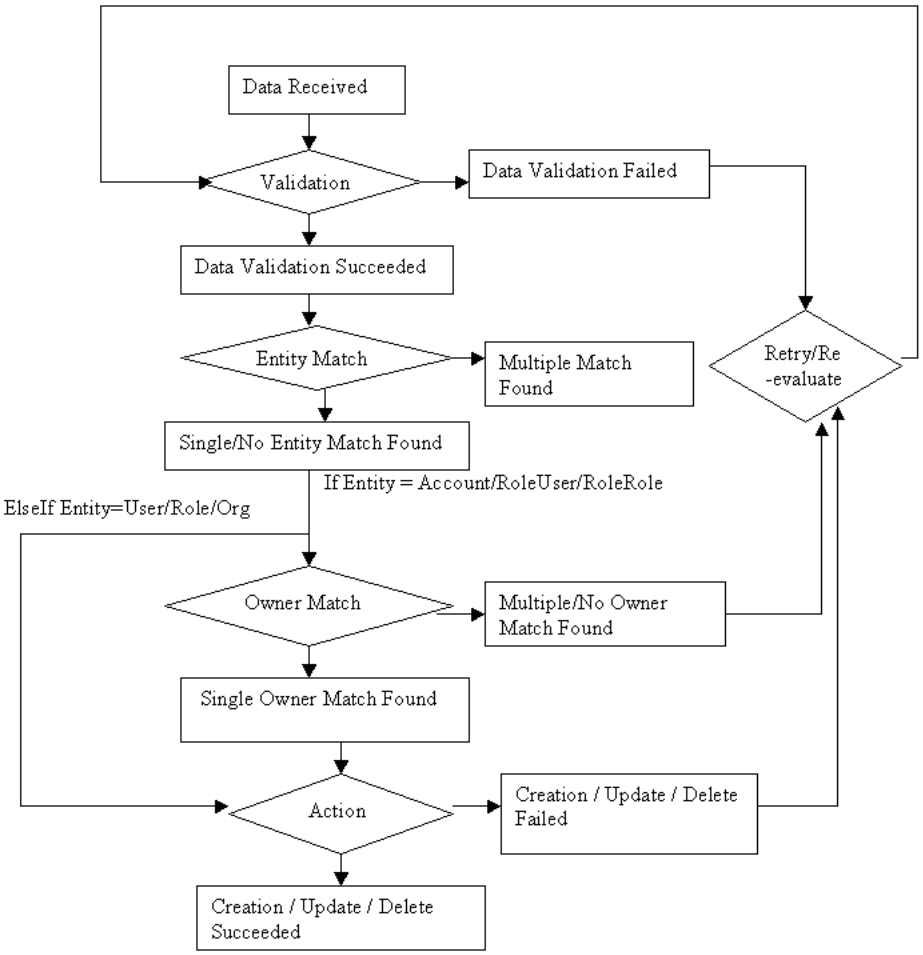
23.2.11 Reconciliation Event Management

The reconciliation events are managed by using the Event Management section of Oracle Identity System Administration. The Event Management section lets you view and manage reconciliation events generated by Oracle Identity Manager reconciliation engine. These events are generated through scheduled reconciliation runs. The Event Management section provides search capabilities on reconciliation runs as well as events. Users can use the Event Management section to perform reconciliation manually on generated events.

See Also: "Managing Reconciliation Events" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager* for more information about the managing reconciliation events in the Oracle Identity System Administration

Figure 23–2 shows the various stages in the lifecycle of a reconciliation event.

Figure 23–2 Reconciliation Event Lifecycle



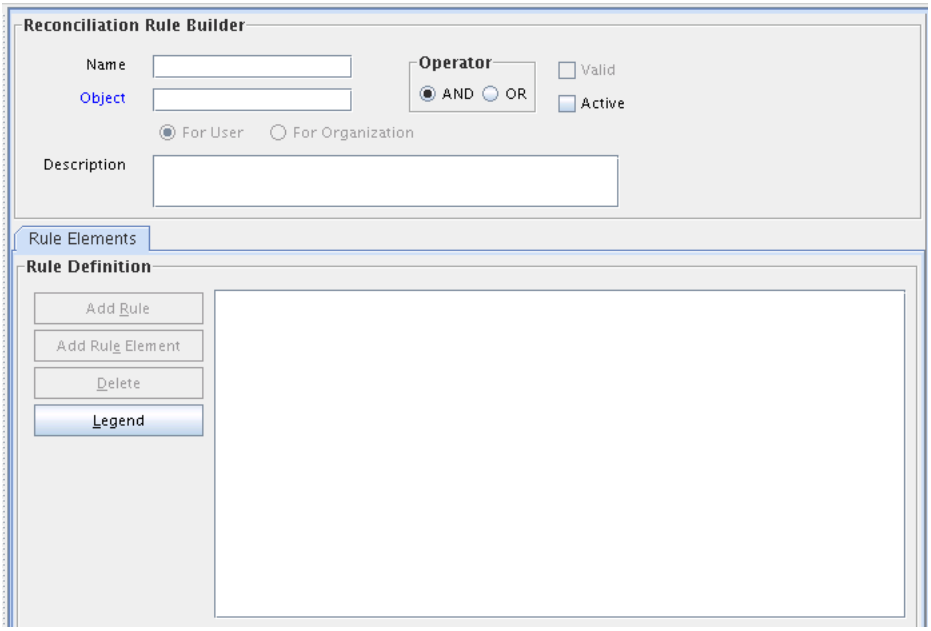
23.3 Defining Reconciliation Rules

You can define reconciliation rules that are invoked at the following instances:

- 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.

The Reconciliation Rules form in the Design Console is used to create and manage reconciliation rules in Oracle Identity Manager. This form is located in the Development Tools folder. Figure 23–3 shows the Reconciliation Rules form.

Figure 23–3 Reconciliation Rules 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

23.3.1 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.

23.3.2 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 23–4.

Table 23–4 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.

23.3.3 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.

23.3.4 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**.

23.4 Developing Reconciliation Scheduled Tasks

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 26, "Developing Scheduled Tasks" for information about developing a scheduled task

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 23-25 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 III, "Connectors" and Part IV, "Requests and Approval Processes" for details about creating conditional tasks, adapters, and dependencies

23.5 Updating Reconciliation Profiles Manually

This section describes updating reconciliation profiles manually in the following sections:

- Creating and Updating Reconciliation Profiles
- Changing the Profile Mode

23.5.1 Creating and Updating Reconciliation Profiles

For reconciliation based on resource objects, the profile name is the same as that of the resource object. For example, if resource object name is testresource, then the default profile name is also testresource. The corresponding reconciliation staging table name is available in the profile. If the resource has Multi-Language Support (MLS) data, then the MLS staging and Oracle Identity Manager table names are also available in the profile. See Table 23–1, " Elements in the Reconciliation Profile XML" for information about the structure and the elements in the reconciliation profile.

If the resource object has child forms, then for each child form, the Oracle Identity Manager table name and staging table name are available in the profile. Each staging table has a corresponding entity definition XML file, the name is same as staging table name with dot xml extension (.xml), which is stored in the MDS.

To change a anything in a reconciliation profile, for instance attribute batch size, either the profile can be updated manually or by using the Design Console. To update a reconciliation profile:

Note: If a reconciliation profile is changed by using the Design Console, the reconciliation profile must be regenerated by clicking the **Create Reconciliation Profile** button in the Object Reconciliation tab of the Design Console.

---

1. Export the /db/PROFILE_NAME profile document from MDS.
2. Make changes in the XML file, for example, change the batch size value.
3. Set the value of the configure attribute to true. For information about this attribute, Table 23–1, " Elements in the Reconciliation Profile XML".
4. Import the updated profile into MDS. See "Migrating User Modifiable Metadata Files" on page 37-1 for information about exporting and importing metadata to and from MDS.

   This automatically updates the staging tables and the corresponding staging table entity definitions.

23.5.2 Changing the Profile Mode

You can use one of the following methods to change the profile mode property from CHANGELOG to REGULAR:

> **See Also:** "Mode of Reconciliation" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager* for information about changelog and regular reconciliation modes

- Change the value of the changeType attribute in the profile, for example:

```
<profile xmlns="http://www.oracle.com/oracle/iam/reconciliation/config"
changeType="REGULAR" batchSize="500" resourceType="Organization"
name="Xellerate Organization">
```

- Change the attribute during event creation:

The event creation API, introduced in Oracle Identity Manager 11*g* Release 2 (11.1.2.2.0), contains three parameters. The first two parameters are same as those used in previous create event APIs. The third parameter can have attributes such as dateFormat, changeType, eventFinished, and actionDate. The following is the constructor for the third parameter:

```
EventAttributes(boolean eventFinished, java.lang.String dateFormat, ChangeType
changeType, java.util.Date actionDate)
```

See *Oracle Fusion Middleware Java API Reference for Oracle Identity Manager* for more information about the third parameter.

You can use this API to set the changeType as follows:

```
public long createReconciliationEvent(String objName, Map<String, Object>
inputData, EventAttributes eventAttribs);
```

Note: Using the API to set the changeType attribute overrides the value of the changeType attribute set in the profile.

23.6 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 Manage. However, in 11*g* Release 2 (11.1.2.2.0), 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 11*g* Release 2 (11.1.2.2.0).

Each run of a connector is known as a job. In 11*g* Release 2 (11.1.2.2.0), 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.

23.6.1 The ReconOperationsService API

The APIs in oracle.iam.reconciliation.api.ReconOperationsService are required for the following tasks:

- Ignore Event
- Create Event (single/bulk)
- Process Event
- Deletion Detection

> **See Also:** *Oracle Fusion Middleware Java API Reference for Oracle Identity Manager* for details about the APIs in oracle.iam.reconciliation.api.ReconOperationsService

The preferred API and the order of invocation of these APIs is as follows:

1. **Ignore Event:** This is a way to prevent event creation and processing of target system data that already exists in Oracle Identity Manager. The API invocation is as follows:

   ```
   boolean ignoreEvent(String resourceObjectName, Map inputData, String
   dateFormat) throws tcObjectNotFoundException, tcAPIException
   ```

   This API is used to validate whether or not the reconciliation create event needs to be raised for the specified object. If this API returns true, then you can skip the event creation, which saves extra event creation in the database.

   Similar to the ignoreEvent API, the ignoreEventAttributeData method can be used to validate whether or not the reconciliation create event flow needs to be raised for single and multivalued data coming from the target system. In this release, only the account entity type has such data. The API is as shown:

   ```
   boolean ignoreEventAttributeData(String resourceObjectName, Map inputData,
   String multiValueFieldName, Map[] childDataList, String dateFormat) throws
   tcAPIException, tcObjectNotFoundException
   ```

   > **Note:** Either ignoreEvent or ignoreEventAttributeData must be invoked; both the APISs are not required to be invoked.

2. **Create Event:** This can happen via single event creation or bulk event creation APIs. This flow simply stores target system data in staging tables. The processing of this data asynchronously takes place later on.

   Create Event (Single): This consists of the following APIs:

   - Use the createReconciliationEvent method to provide the data for creating reconciliation events. If there is child or multivalued data, then set the value of the eventAttribs.eventFinished flag to false. Otherwise, set this value to true. It returns the eventId of the created Event.

     ```
     long createReconciliationEvent(String resourceObjectName, Map<String,
     Object> inputData, EventAttributes eventAttribs)
     ```

   - The child data is provided using the addMultiAttributeData method. If there is no child data or the eventAttribs.eventFinished flag is set to true, then this API must not be invoked.

```
long addMultiAttributeData(long plReconciliationEventKey, java.lang.String
psFieldName, java.util.Map poData) throws tcAPIException,
tcEventNotFoundException,
tcEventDataReceivedException,tcAttributeNotFoundException
```

Note: For better performance with bulk multivalued attributes or the data for multiple child records instead of a single child record, use the following API:

```
void addDirectBulkMultiAttributeData(long reconciliationEventKey,
long reconciliationAttributeKey, String tableFieldName, List
dataList,String dateFormat) throws tcAPIException,
tcEventNotFoundException, tcAttributeNotFoundException,
tcEventDataReceivedException,tcInvalidAttributeException
```

- The providingAllMultiAttributeData method specifies whether the multivalued data being provided is the entire list of data, or only changeset that has been added/updated. By default, the value of the pbFlag is false. If there is no child data or the eventAttribs.eventFinished flag is set to true, then this API must not be invoked.

  ```
  public void providingAllMultiAttributeData(long plReconciliationEventKey,
  String psFieldName, boolean pbFlag) throws tcAPIException;
  ```

- The finishReconciliationEvent method is used to mark the end of event creation flow. Particular event status is updated to Data Received, which means that all the data for the particular event, including the child data if any, has been provided. If the eventAttribs.eventFinished flag is set to true, then this API must not be invoked.

  ```
  void finishReconciliationEvent(long eventId) throws tcAPIException,
  tcEventNotFoundException, tcEventDataReceivedException
  ```

- The callingEndOfJobAPI method processes all the reconciliation batches in the job. For a scheduled job, this API is automatically called when the job ends. This API must be explicitly called for a nonscheduled job API invocation.

  ```
  void callingEndOfJobAPI() throws tcAPIException
  ```

Create Event (Bulk): This consists of the following API:

```
ReconciliationResult createReconciliationEvents(BatchAttributes batchAttribs,
InputData... input)
```

This is the bulk create API. It creates bulk reconciliation events for the data passed in input data. It accepts all the data including multivalued attributes, and submits it for processing as one batch if the size of data is less then or equals to the batch size. Otherwise, it submits the data in multiple batches. There is no need to call any other API after this.

3. **Process Event:** This is a way to force the backend processing of an already created event. The processReconciliationEvent(eventId) API is invoked after create event flow has finished and an already created event needs to be processed as well. This API processes only a particular event, it does not update the batch or job status. If batch status needs to be updated as well, then invoke the callingEndOfJobAPI API after this. Using this API is not recommended because it is synchronous and processes data one at a time, rather than in batch.

Note: If the processReconciliationEvent API is used for processing, then set the reconciliation batch size (batchSize parameter) to 0 in the reconciliation profile of the resource object. See Table 23–5 for more information about this step.

4. **Deletion Detection:** This is a way to delete extra data in Oracle Identity Manager that does not exist in the target system. This consists of the following APIs:

   - The provideDeletionDetectionData method takes the list of all the existing target system data for a resource object as input, and then returns a list of matching data found in Oracle Identity Manager.

     ```
     Set provideDeletionDetectionData(String resourceObjectName, Map[]
     inputData) throws tcAPIException, tcIDNotFoundException,
     tcMultipleMatchesFoundException
     ```

   - The getMissingAccounts method takes the list keys of already found data in Oracle Identity Manager, and returns a list of extra data that is in Oracle Identity Manager but not in the target system. It retrieves all keys from Oracle Identity Manager and compares them with the keys present in the set returned by the provideDeletionDetectionData method.

     ```
     Thor.API.tcResultSet getMissingAccounts(String objectName, Set
     accountsFound) throws tcAPIException, tcIDNotFoundException,
     tcDataNotProvidedException
     ```

   - The deleteDetectedAccounts method takes a list of data found only in Oracle Identity Manager as input, and invokes a delete type create reconciliation event API call, one at a time. The tcResultSet returned by the getMissingAccounts method is passed as parameter to this API.

     ```
     long[] deleteDetectedAccounts(Thor.API.tcResultSet poDetectedAccounts)
     throws tcAPIException, tcAPIException
     ```

23.6.2 Invoking Non-scheduled Task-Based Reconciliation in a Multithreaded Environment

Example 23–2 shows the sample code to invoke non-scheduled task-based reconciliation in a multithreaded environment:

Example 23–2 Invoking Non-scheduled Task-based Reconciliation in a Multithreaded Environment

```
public class UserNonSTBasedRecon{

     private AtomicInteger threadCount =new AtomicInteger(0);

public Long getRandomLong(int maxValue) {
        Random random = new Random();
        long token = random.nextInt(maxValue);
        return token;
    }
     @Test
     public void testCreateUsersUsingNonScheduleTaskConnectorWithThreads() throws
Exception {

         Thread t = new CreateEvent();
```

```
            t.start();
            Thread t2 = new CreateEvent();
            t2.start();

        while (true) {
                Thread.currentThread().sleep(5000);
                if (threadCount.get() == 2){
                OIMClient oimClient = null;
                ReconOperationsService reconServ =
oimClient.getService(ReconOperationsService.class);
                    reconServ.callingEndOfJobAPI();
                    break;
                }
            }
        }

        public class CreateEvent extends Thread {

            @Override
            public void run() {

            String ctxFactory = "weblogic.jndi.WLInitialContextFactory";
                OIMProfileReader reader = new OIMProfileReader();
                String appServerType = reader.getString("appserver.type");
                String hostName = reader.getString("weblogic.host");
                String port = reader.getString("weblogic.port");
                String serverURL = "t3://" + hostName + ":" + port;
                System.out.println("Server URL is : " + serverURL);
                System.out.println("Context Factory is : " + ctxFactory);
                Hashtable<String, String> env = new Hashtable<String, String>();
                env.put(OIMClient.JAVA_NAMING_PROVIDER_URL, serverURL);
                env.put(OIMClient.JAVA_NAMING_FACTORY_INITIAL, ctxFactory);

                OIMClient client = new OIMClient(env);
                String username = "xelsysadm";
                String password = "Welcome1";
                try {
                    client.login(username , password.toCharArray());
                } catch (LoginException e1) {
                    throw new SuperRuntimeException(e1.getMessage(), e1);
                }

                String uniq2 = getRandomLong(10000).toString();
                long jobId = getRandomLong(10000);
                ContextManager.setValue(Constants.JOB_HISTORY_ID, new
ContextAwareNumber(jobId));
                ContextManager.setValue(Constants.JOB_NAME_CONTEXT, new
ContextAwareString(jobId +""));
                ReconOperationsService recon;
                try {
                    recon = client.getService(ReconOperationsService.class);
                    int count = 50;
                    HashMap<String, String> hm = new HashMap<String, String>();
                    ArrayList<Long> eventKeys = new ArrayList<Long>();
                    for (int i = 0; i < count; i++) {
                        hm.put("UserLogin", uniq2 + "ThreadTest" + i);
                        hm.put("FirstName", uniq2 + "Thread" + i);
                        hm.put("lastname", "Test");
                        hm.put("Type", "End-User");
                        hm.put("OrganizationName", "Xellerate Users");
```

```
                    hm.put("EmpType", "Full-Time");
                    hm.put("Middlename", "MID");
                    System.out.println("Creating Recon event i ="+ i);
                    long rceKey = recon.createReconciliationEvent("Xellerate
User", hm, true);
                    eventKeys.add(rceKey);
                }
                assertEquals(count, eventKeys.size());
            } catch (Exception e) {
                throw new SuperRuntimeException(e.getMessage(), e);
            } finally {
                threadCount.set(threadCount.get()+1);
                ContextManager.popContext();
            }
        }
    }
 }
```

23.7 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.

23.8 Troubleshooting Reconciliation

Before troubleshooting issues related to reconciliation, change the reconciliation logging level to INFO. To do so, add the following logger by using Oracle Enterprise Manager:

- **Name:** oracle.iam.reconciliation
- **Oracle Diagnostic Logging Level (Java Level):** NOTIFICATION:1(INFO)

For detailed steps of adding a logger, see "Configuring Logging" in *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager*.

Note: To change the logging level, you can also modify the /domains/*DOMAIN_NAME*/config/fmwconfig/servers/*OIM_SERVER*/logging.xml file. To do so:

1. In the logging.xml file, add a new logger, as shown:

```
<LOGGER NAME="oracle.iam.reconciliation" LEVEL="INFO"/>
```

2. Change the logging level of the 'console-handler' log_handler to INFO.
3. Restart Oracle Identity Manager.

This section describes troubleshooting reconciliation issues in the following sections:

- Troubleshooting General Reconciliation Issues
- Troubleshooting Database-Related Reconciliation Issues
- Troubleshooting Reconciliation Profile Configuration Failures
- Troubleshooting LDAP Reconciliation Issues

23.8.1 Troubleshooting General Reconciliation Issues

Table 23–5 lists the troubleshooting steps that you can perform if you encounter reconciliation errors:

Table 23–5 Troubleshooting Reconciliation

Problem	Solution
Failure in processing events	The error details can be obtained from the reconciliation tables, such as: ■ 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
Various data corruption issues resulting due to duplicate processing (both single and bulk processing) in case of push-based connectors when they are processing reconciliation using the processReconciliationEvent API For example, duplicate account creation, status unexpectedly getting changed, and so on.	Set the reconciliation batch size (batchSize parameter) to 0 in the reconciliation profile of the affected resource object.

Table 23–5 (Cont.) Troubleshooting Reconciliation

Problem	Solution
Failure occurring in kernel orchestration handler	The orchestration ID can be tracked from the reconciliation table, which can further be used to check the status of related handlers, such as: ■ 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 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.
After a job run, a lot of events are in the Data Received status.	Check if related batches are in Ready For Processing status by using the following statement: `select rb_batch_status, rb_note from recon_batches where rb_batch_status = 'Ready For Processing' and rj_key = JOB_ID_ON_UI` 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: ■ 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.
Race Condition - Events are in failed status because some dependent attribute is still not reconciled, for example, user's manager/organization needs to be reconciled before user.	■ If the size of the data is small, then retry reconciliation automatically handles the race condition, but it is slow. ■ If the size of the data is large, then perform the reconciliation two times. Remove the mapping for the dependent field for the first full reconciliation, and then add the dependent field mapping and perform the full reconciliation second time.
The following error is generated when performing user update for trusted source reconciliation: `ORA Error Code =>ORA-00001: unique constraint (.) violated`	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.

Note: The oracle.iam:type=Reconciliation,name=EventDiagnostic MBean with method 'diagnose' can be used to diagnose end-to-end reconciliation event flow. This MBean can be accessed by using Oracle Enterprise Manager. The diagnose method takes reconciliation EventID as input, and shows the following information about the event:

- Event, batch, job, and history details. This includes the RE_NOTE and the RB_NOTE values, and therefore, indicates the reason for failure, if any, or else the associated orchestration IDs.
- Old state table data, which is relevant for audit.
- Staging table data, which is data coming to reconciliation from the target system.
- Orchestration details, which includes all the event handlers that executed along with their status and reason for failure, if any.

23.8.2 Troubleshooting Database-Related Reconciliation Issues

This section the describes the following database-related issues for reconciliation:

Missing Critical Oracle Database 11*g* 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 11*g* 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 11*g* Release 2 Interim Patches

Running some SQL scripts might generate incorrect or inconsistent results on Oracle Database 11*g* 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_RECONBLKROLEMLSWRAPPER"
ORA-06512:

at weblogic.jms.client.JMSSession$UseForRunnable.run(JMSSession.java)
at weblogic.work.SelfTuningWorkManagerImpl$WorkAdapterImpl.run(SelfTuningWorkMana
gerImpl.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_RECONBLKROLEMLSWRAPPER"
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:** "Reconciliation Best Practices" on page 23-37 for information about creating indexes for reconciliation and collecting database statistics

23.8.3 Troubleshooting Reconciliation Profile Configuration Failures

For any issues related to profile configuration failures, validate the profile by using the ProfileValidator Mbean available in Oracle Enterprise Manager. If the profile is invalid, then the profile is displayed along with the cause of the invalid profile.

Table 23–6 lists the troubleshooting steps that you can perform if you encounter reconciliation errors.

Table 23–6 Troubleshooting Reconciliation Profile Configuration Failures

Problem	Solution
The profile is invalid, and it fails to load with the following exceptions: oracle.iam.reconciliation.exception.ConfigNotFoundException OR oracle.iam.reconciliation.exception.Config with internal exception org.xml.sax.SAXParseException	Perform any one of the following: ■ The exact problem can be diagnosed and fixed by checking the schema validation message. ■ Validate the reconciliation profile XML by using the MBean in Oracle Enterprise Manager or the Validate Recon Profile test in the Diagnostic Dashboard. ■ Validate the reconciliation profile by importing the profile and the XSD into an XML schema-aware editor and validate against that schema in that editor, which can point to the exact cause of the failure.
Importing a valid reconciliation profile XML into a system fails to create the necessary configurations.	Check the profile.configure attribute. The value of this attribute must be true. Check the profile.active attribute. The value of this must be true. Or if the attribute is not present, then it means that profile.active is true.
The following error is generated: `oracle.iam.reconciliation.exception.ReconciliationException: Exception occurred while inserting data into table` *`STAGING_TABLE_NAME`* `due to STAGING_TABLE_NAME` This means that a valid reconciliation profile is loaded, but it has not created any configuration in Oracle Identity Manager.	Check the profile.configure and profile.active attributes.

Note: The ProfileValidator Mbean is available for validating the reconciliation profile. This MBean can be accessed by using Oracle Enterprise Manager.

23.8.4 Troubleshooting LDAP Reconciliation Issues

This section describes the following issue related to LDAP reconciliation:

LDAP User Create and Update Reconciliation Scheduled Job Fails With Error

In an integrated environment of Oracle Identity Manager with Oracle Unified Directory (OUD) and libOVD, the LDAP User Create and Update Reconciliation scheduled job fails with the following error:

```
"Invalid syntax of the provided cookie"
```

This error occurs when a numeric change number value, for example 0, is being passed to libOVD when OUD is used as LDAP, and External Change Log (ECL) is enabled in the libOVD adapter. In the scheduled job page, the numeric value, for example 0, is set in the Last Change Number field. This attribute is used to specify the last changelog identifier processed by this scheduled job.

For OUD, when ECL mode is on, the value of the Last Change Number field must not be a numeric value but a string value, such as:

```
"dc=us,dc=oracle,dc=com:00000142b752f5cf473900000ce1;"
```

The following is an example command to get the expected value:

```
ldapsearch -h localhost -p 1389 -D "cn=Directory Manager" -w welcome1 -b "" -s
base "objectclass=*" lastExternalChangelogCookie
dn: lastExternalChangelogCookie:
dc=us,dc=oracle,dc=com:00000142b752f5cf473900000ce1;
```

23.9 Populating Data in the RECON_EXCEPTIONS Table

The RECON_EXCEPTIONS table in Oracle Identity Manager database is used to capture error messages generated during account reconciliation. This data is collected for the purpose of generating reports.

> **See Also:** "Account Reconciliation" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager* for information about account reconciliation

If a reconciliation match is found to a deleted user, then you must insert USER_DELETED in the REX_EXCEPTION column and the key of the deleted user in the USR_KEY column of the RECON_EXCEPTIONS table.

If no match is found, then insert USER_NOT_FOUND in the REX_EXCEPTION column.

If account match is found, then check if the account is already deprovisioned. Then insert into RECON_EXCEPTIONS table with the value `RESOURCE_DEPROVISIONED` in the REX_EXCEPTION column for the user who is to be provisioned.

To populate the RECON_EXEPTIONS table with exception data:

1. Fetch all the events with the change type != ('Modify' , 'Delete') and event status as ('Single User Match Found', 'Single Org Match Found').
2. Provision the resource object for the entities by performing the following:
   a. Collect the exception data from RECON_EXCEPTION DB table. To do so, perform any one of the following:
      – Check if the value of the XL.EnableExceptionReports property is TRUE. If it is set to TRUE, then continue to the next step. Otherwise, do not collect the exception data.
      – Select the obj_initial_recon_date in the obj table for the resource object being provisioned, and check if it is earlier than today's date. If an earlier date is displayed, then continue to the next step. Otherwise, do not collect the exception data.
   b. While provisioning the resource object to the user, check if the resource object has already been deprovisioned in Oracle Identity Manager:
      – If the resource object is already deprovisioned, then insert into RECON_EXCEPTIONS table the value `RESOURCE_DEPROVISIONED` in the REX_EXCEPTION column for the user who is to be provisioned.
      – If the resource object is not deprovisioned, then insert into RECON_EXCEPTIONS table the value `RESOURCE_NEVER_PROVISIONED` in the REX_EXCEPTION column for the user who is to be provisioned.

23.10 Reconciliation Best Practices

This section describes how to improve performance by identifying indexes that are required for connector tables and reconciliation tables. It contains the following topics:

- Additional Indexes Requirement for Matching Module
- Collecting Database Schema Statistics for Reconciliation Performance

Note: Oracle recommends configuring both the entitlement attribute and the key attribute for the child data in reconciliation field mappings to enable effective duplicate entitlement or child data validation. See "Duplicate Validation for Entitlement or Child Data" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager* for information about duplicate validation for entitlements or child data.

23.10.1 Additional Indexes Requirement for Matching Module

When Oracle Identity Manager is installed, the necessary indexes are created in the Oracle Identity Manager database schema. However, there can be additional indexes required because of dynamic nature of some of the features in Oracle Identity Manager. This is especially true for reconciliation.

Reconciliation uses matching algorithm to find if the user/account/role/organization for which the change is requested is already existing in Oracle Identity Manager or not. The matching algorithm compares the data in set of columns in Oracle Identity Manager with the data in target horizontal table columns. The columns that contains the matching rules are defined in the reconciliation profile. To improve the performance of the matching operation quickly, there must be correct indexes created on the matching rule columns.

To illustrate the recommended method of identifying the appropriate indexes, a sample Active Directory (ADUser) profile present in the Meta Data Store (MDS) repository is taken as an example.

Selecting Indexes for Reconciliation

To select indexes based on the matching rule criteria:

1. Open the AD profile file in a text editor.

   Note: The AD user profile must be imported from the MDS by using the Oracle Enterprise Manager, as described in "Migrating User Modifiable Metadata Files" on page 37-1.

2. Search for all occurrences of <matchingRule> tag element in the AD profile, as shown in Figure 23–4:

Figure 23–4 The <matchingRule> Tag Element

```
<profile ownerType="User" changeType="CHANGELOG" auditEnabled="true" batchSize="500" resourceType="Account" name="AD User">
  <matchingRule>
    ((UPPER(USR.usr_udf_obguid)=UPPER(RA_ADUSERE469E5C8.RA_OBJECTGUID)) OR
    (UPPER(USR.usr_login)=UPPER(RA_ADUSERE469E5C8.RA_USERID5A729570)))
  </matchingRule>
  <form oimTableName="UD_ADUSER" stagingTableName="RA_ADUSERE469E5C8" name="AD User">
    <matchingRule>
      (UD_ADUSER.UD_ADUSER_OBJECTGUID=RA_ADUSERE469E5C8.RA_OBJECTGUID)
    </matchingRule>
    <targetAttributes>
      <targetAttribute type="String" keyField="true" name="objectGUID">
        <stagingField type="String" length="32" name="RA_OBJECTGUID"/>
        <oimAttribute type="String" fieldName="UD_ADUSER_OBJECTGUID" fieldType="String" name="Object GUID"/>
      </targetAttribute>
      <targetAttribute type="String" required="true" name="User ID">
        <stagingField type="String" length="150" name="RA_USERID5A729570"/>
        <oimAttribute type="String" fieldName="UD_ADUSER_UID" fieldType="String" name="User ID"/>
      </targetAttribute>
      <targetAttribute type="ITResource" name="IT Resource">
        <stagingField type="ITResource" length="19" name="RA_ITRESOURCE15641F83"/>
        <oimAttribute type="Number" fieldName="UD_ADUSER_AD" fieldType="Number" name="AD Server"/>
      </targetAttribute>
    </targetAttributes>
    <form oimTableName="UD_ADUSRC" stagingTableName="RA_ADUSERGROUPDETA902DB909" name="memberOf">
      <matchingRule>
        (UD_ADUSRC.UD_ADUSRC_GROUPNAME=RA_ADUSERGROUPDETA902DB909.RA_MEMBEROF)
      </matchingRule>
      <targetAttributes>
        <targetAttribute type="String" keyField="true" name="memberOf">
          <stagingField type="String" length="256" name="RA_MEMBEROF"/>
          <oimAttribute type="String" fieldName="UD_ADUSRC_GROUPNAME" fieldType="String" name="UD_ADUSRC_GROUPNAME"/>
        </targetAttribute>
      </targetAttributes>
    </form>
  </form>
  <actionRules>
    <actionRule condition="No Matches Found" action="Assign To Administrator With Least Load"/>
    <actionRule condition="One Entity Match Found" action="Establish Link"/>
    <actionRule condition="One Process Match Found" action="Establish Link"/>
  </actionRules>
</profile>
```

3. After identifying the columns constituting each matching rule, create the indexes accordingly.

Note:

- If any key field is defined in Oracle Identity Manager as case-insensitive, then a function-based index on that key field must be created. For example, if the connector code internally performs a search for the first name, assuming that FIRST_NAME is a key, then appropriate indexing must be done.
- If multiple or composite keys are used for looking up a user, then choose between individual or composite indexes.
- Pointers for required indexes can also be taken by monitoring the real-time running of reconciliation process from the database side by using a performance-monitoring tool, such as Oracle Enterprise Manager, or through the Automatic Workload Repository (AWR) Reports available in Oracle Database 11*g*.
- To some extent, index creation is automated for profiles created or updated via the Design Console or Deployment Manager import. Validate the automatically created indexes per the rules defined in this section. You must rectify the indexes manually if there are any issues. For profiles created or updated manually, the indexes are not automated and must be created manually. In addition, there is no automation for dropping the indexes if the matching rule field has changed. Dropping indexes must be done manually.
- The list of existing indexes can be viewed on Oracle Enterprise Manager by using the ProfileValidator Mbean.

23.10.2 Collecting Database Schema Statistics for Reconciliation Performance

Database statistics is essential for the Oracle optimizer to select an optimal plan in running the SQL queries. Oracle recommends that the statistics are collected regularly for Oracle Identity Manager schema. Because Oracle Identity Manager 11*g* Release 2 (11.1.2.2.0) uses lot of database SQL features for reconciliation process, make sure that the schema statistics are updated before running the reconciliation.

Note:

- Other options with DBMS_STATS.GATHER_SCHEMA_STATS API can be used as required, such as DEGREE,ESTIMATE_PERCENT based on the environment, data profile, Oracle DB Edition and underlying hardware capabilities.
- See "Database Performance Monitoring" in the *Oracle Fusion Middleware Performance and Tuning Guide* for more information about collecting database schema statistics.

Because Oracle Identity Manager reconciliation process is a data-intensive process and quickly brings in large volume of data, database statistics must also be able to represent the underlying data correctly. To achieve this, refer to the following guidelines:

- Make sure that statistics is collected for reconciliation on a fresh setup or with a low volume with no or negligible existing data in the Oracle Identity Manager schema. Maximum row count in relevant Oracle Identity Manager tables must be between 100 and 1000 rows. Examples of tables are USR table for trusted source reconciliation and parent account table for target resource reconciliation.
- For the statistics to be a proper representative of data distribution after reconciliation has started and is expected to bring in a large volume of data, such as more than 20000 users or accounts, collect Oracle Identity Manager schema statistics in the following manner:
  a. Plan to gather statistics after the initial collection only after reconciliation has started successfully and has been running for a while. To verify this, check the counts of a few key tables from the Oracle Identity Manager schema, such as USR table for trusted source reconciliation and parent account (UD_*) tables for target resource reconciliation.
  b. After reconciliation has brought in almost 20000 to 25000 rows in the USR table or in the parent account tables, statistics can be collected.

  Note:

  - Statistics can be gathered concurrently with reconciliation running.
  - The row counts specified in the guidelines are examples and you can determine any other row count for collecting statistics.

- After the statistics is collected, the performance might not improve immediately. However, as older SQL Plans are cleared from the shared pool of the Oracle Database, new and more efficient plans are created and performance improves.

23.11 Monitoring Reconciliation Performance Using DMS

Dynamic Monitoring Service (DMS) commands are used to view performance metrics and configure event tracing. The following DMS matrices are relevant for monitoring reconciliation performance:

- OIM_ScheduledJob: The time taken by a particular scheduled job run.
- Reconciliation Service (ReconOperationsService Or tcReconciliationOpIntf): The time taken by each API on the reconciliation service for creating an event. Connector throughput can be calculated as 'Total Scheduled Job time – Total time for creating the events.
- OIM_JMS: ActionTask provides information about actual reconciliation processing (stored procedure) time. OrchestrationAsyncTask provides information about overall orchestration postprocessing. XLAuditMessage provides information about actual audit processing.
- OIM_EventHandlers: Time taken by each eventhandler within an orchestration.
- DMS Dump: If unable to resolve performance and functional issues, then DMS dumps should be provided for analysis. The command is:

```
${mwhome}/oracle_common/common/bin/wlst>>dms.log
Connect('adminusername','adminpassword');
dumpMetrics(format='xml');
Exit();
```

  For detailed information about the command, see the "dumpMetrics" section in "DMS Custom WLST Commands" of the *Oracle Fusion Middleware WebLogic Scripting Tool Command Reference*.

24

Using the Bulk Load Utility

Oracle Identity Manager may be one among many repositories of entity 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. The Bulk Load utility can load users, accounts, roles, role hierarchy, role membership, and role category data.

This document is divided into the following sections:

24.1 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 for users.
- 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.

24.2 Prerequisites for Running the Bulk Load Utility

Running the Bulk Load utility has the following prerequisites:

- Installing the Bulk Load Utility
- Preparing Your Database for a Bulk Load Operation

24.2.1 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.

   The utility can also be run directly from the *MIDDLEWARE_HOME*/Oracle_IDM1/server/db/oim/oracle/Utilities/oimbulkload/ directory.

   ---

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

24.2.1.1 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 and provide additional grants to the Oracle Identity Manager database user to perform required operations during Bulk Load. See "Creating a Datafile in the Oracle Identity Manager Tablespace" on page 24-5 for more information.

24.2.1.2 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.

24.2.1.3 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 users that you have loaded.

24.2.2 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

24.2.2.1 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.

24.2.2.2 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

After providing input to prompted Oracle Identity Manager database user name, appropriate grants to perform required operations during Bulk Load are provided to the database user.

24.3 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 characterset, in the following format:

NLS_LANG = *LANGUAGE_TERRITORY*.UTF8

For example:

NLS_LANG = American_America.UTF8

   - On UNIX computers:

     OIMBulkload/scripts/oim_blkld.sh
   - On Microsoft Windows computers:

     OIMBulkload\scripts\oim_blkld.bat

Note: OIMBlukload is the directory in which the scripts/sqls/csv_files/lib/sample_data directories are present.

---

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 24-4.
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 24-12.

> **Note:** See "Determining Values for the Input Parameters of the Utility" on page 24-12 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".

24.4 Performance Best Practices for Bulk Load

To enhance the performance of the Account Bulk Load operation:

1. Split the data load in phases for a high-volume entity data load for Users/Accounts/Role Membership, for example, when data load is greater than 1 million for Users and greater than 250 thousand for Accounts.
2. The phase-wise load can be in the initial size of 500 thousand, and thereafter, in the size of 2 to 3 million Entity data.
3. Perform Stats gathering operation essentially after the first and second batch of data load.

   For information about Stats gathering operation, see "Monitoring Oracle Identity Manager Performance" in the *Oracle Fusion Middleware Performance and Tuning Guide*.
4. For Account data load, when the source is database table, then make sure that relevant indexes are present on the columns as per the reconciliation matching rules.

   For further details please refer to the documentation on "Additional Indexes Requirement for Matching Module" in the *Oracle Fusion Middleware Developer's Guide for Oracle Identity Manager*.

24.5 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 24-5 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 24-6 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 diagnostic 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
- Generating an Audit Snapshot

24.5.1 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 Self Service as a user with Create User privileges.
2. On the left navigation pane, under Administration, click **Users**. The Search Users page is displayed.

3. From the Actions menu, select **Create**. The Create User page is displayed with input fields for user profile attributes.
4. Specify values for the following fields:
   - User Login
   - First Name (optional)
   - Last Name
   - Organization: Select **Xellerate Users**.
   - Password
   - Confirm Password
5. Click **Submit**.

24.5.2 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

24.5.2.1 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 11*g* Release 2 (11.1.2.2.0).
- 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.

24.5.2.2 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`
- 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.
- 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.

Table 24–1 shows the structure of a sample database table.

Table 24–1 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)

24.5.3 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 24-5 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 24-32 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 24-8.

24.5.4 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;
```

24.5.5 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 24–1 shows sample statistics recorded in the log file at the end of a bulk load operation to store OIM User data.

Example 24–1 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.

24.5.6 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.

Note: Being a database-intensive operation by design, Bulk Load disables the constraints and indexes on the relevant Oracle Identity Manager entity tables during the start of the operation. Bulk Load operation failure towards the end of the load might at times render the indexes and constraints in disabled state. To identify and fix this issue, manually restore the indexes and constraints as follows:

1. Identify the unusable indexes and disabled constraints. To do so, the following SQL queries or similar mechanism can be used:

```
SELECT TABLE_NAME, CONSTRAINT_NAME FROM user_constraints WHERE
status = 'DISABLED';
SELECT index_name FROM user_indexes WHERE status = 'UNUSABLE';
```

2. Enable the constraints and rebuild the indexes manually, as shown:

```
ALTER TABLE TABLE_NAME ENABLE CONSTRAINT CONSTRAINT_NAME;
ALTER INDEX INDEX_NAME REBUILD;
```

24.5.7 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.
- If Oracle Identity Manager is synchronized with LDAP, then after running the user data upload, run the Bulk Load Post Process scheduled job with the LdapSync option set to `Yes`. If you want to generate the random password and send an email to the user, then you must configure the email notification and set the Generate Password and Notification parameters to Yes in the Scheduler. See "Predefined Scheduled Tasks" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager* for information about the Bulk Load Post Process scheduled job.

- 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.
- Revoked 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.

24.5.8 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.

Before you generate an audit snapshot, for running the GenerateSnapshot script, you must set the following environment variables:

- APP_SERVER: weblogic
- OIM_ORACLE_HOME: c:\work1\Oracle_IDM1
- JAVA_HOME: C:\jdk160
- MW_HOME: c:\work1
- WL_HOME: c:\work1\wlserver_10.3
- DOMAIN_HOME: C:\work1\user_projects\domains\base_domain

Note: C:\work1\ is a sample directory path of *MW_HOME*.

See "Configuring Auditing" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager* for information about the procedure to generate audit snapshots.

24.6 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 24-5 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 diagnostic data from the operation.
12. Remove temporary tables and files created during the operation.

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.
- Bulk Load utility takes the corresponding application instance name as input to load account data. If the application instance name is not known to the user, then Bulk Load utility prompts for the resource object name and IT resource name, based on which account data is loaded.
- Loading accounts where the target system is Active Directory, make sure that the input source (CSV file or database table) have the following attributes as mandatory in the attribute list along with its values:
  - UD_ADUSER_COMMONNAME
  - UD_ADUSER_USERPRINCIPALNAME

  Failing to load values for these attributes at the time of Bulkload can result into failures in the provisioning-related operations at a later stage for this target.

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

24.6.1 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

24.6.1.1 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 5, "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_FNAME,UD_ADUSER_LNAME,UD_ADUSER_MNAME,UD_ADUSER_
FULLNAME,UD_ADUSER_OBJECTGUID
ADTEST1,"7~CN=ForeignSecurityPrincipals,DC=vivek01,DC=com",adtest1,adtest1,,adt
est1,102
```

  Note: The UD_ADUSER_OBJECTGUID column is mandatory in the parent CSV file for loading accounts by using the bulk load operation. This column must be added to the parent CSV file in spite of nullable column in the database.

The following are sample contents of a child CSV file:

```
UD_ADUSER_UID,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.

24.6.1.2 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 24–2 shows the structure of a sample parent table.

Table 24–2 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 24–3 shows the structure of a sample child table.

Table 24–3 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)

24.6.2 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. This must be entered twice when prompted.

- Application instance name (APP_INSTANCE)

  Name of the application instance corresponding to the account data to be loaded. If the user is not aware of the application instance name, then Account Bulkload utility prompts for the resource object name and IT resource name. The prompt is as shown:

  ```
  Do you know the Application Instance name? (Y,y,N,n)
  ```

  If you enter Y or y, then you are prompted for the application instance name. If you enter N or n, then you are prompted for the following:

  – Resource Object Name (OBJ_NAME)

    If the user is not aware of the application instance name, then Bulk Load utility prompts for the resource object name corresponding to the account data to be loaded.

  – IT Resource Name

    Name of the IT resource created for the target system. This is required only when the user is not aware of the application instance name. The account bulkload utility first prompts for resource object name, and then prompts for IT resource name.

- 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 24-19 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. In addition, enter the column that links the parent and child data.

- 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 24-20 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 24-5 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 24-32 for more information.

- Application Instance (APP_INSTANCE)

  Name of the application instance corresponding to the account data to be loaded.

  If the user is not aware of the application instance name, then account bulkload utility prompts for the Object name (OBJ_NAME)

- User ID (USR_LOGIN)

  The user login ID that is used to determine the user that provisioned Accounts using Bulk Load utility.

24.6.3 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;
```

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;
```

24.6.4 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 24–2 shows sample statistics recorded in the log file at the end of a bulk load operation to store account data.

Example 24–2 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.

24.6.5 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.

Note: Being a database-intensive operation by design, Bulk Load disables the constraints and indexes on the relevant Oracle Identity Manager entity tables during the start of the operation. Bulk Load operation failure towards the end of the load might at times render the indexes and constraints in disabled state. To identify and fix this issue, manually restore the indexes and constraints as follows:

1. Identify the unusable indexes and disabled constraints. To do so, the following SQL queries or similar mechanism can be used:

```
SELECT TABLE_NAME, CONSTRAINT_NAME FROM user_constraints WHERE
status = 'DISABLED';
SELECT index_name FROM user_indexes WHERE status = 'UNUSABLE';
```

2. Enable the constraints and rebuild the indexes manually, as shown:

```
ALTER TABLE TABLE_NAME ENABLE CONSTRAINT CONSTRAINT_NAME;
ALTER INDEX INDEX_NAME REBUILD;
```

24.6.6 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 Accounts tab of the User details page or in My Accounts tab of My Access page for the user.
- Log in to the target system by using the credentials of the newly created account.

24.7 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 24-5 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 diagnostic data from the operation.
12. Remove temporary tables and files created during the operation.

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

24.7.1 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
- Determining the UGP_NAME Generated After Role Load

24.7.1.1 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:

```
UGP_ROLENAME,UGP_NAMESPACE,USR_LOGIN,ORG_NAME,INCLUDE_HIERARCHY
"Finance Controllers",Default,XELSYSADM,Finance,YES
"Finance Controllers",Default,XELSYSADM,Requests,YES
```

- Role load is capable of publishing the roles to organizations to follow the security model in Oracle Identity Manager, with an option to include hierarchy.

  As a value of the ORG_NAME parameter, specify the organization name, such as Finance or Requests, to which you want to publish the roles. Specify YES for INCLUDE_HIERARCHY if you want to publish the roles to the specified organization and its suborganizations. Specify NULL or NO for INCLUDE_HIERARCHY if you want to publish the roles only to the specified organization and not its suborganizations. If you do not specify values for the ORG_NAME and INCLUDE_HIERARCHY parameters, then by default, the roles are published to the Top organization with hierarchy.

- 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,ORG_NAME,INCLUDE_HIERARCHY (UGP_NAMESPACE,ORG_NAME)

    INCLUDE_HIERARCHY 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 combinationation 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)

    ROLE_CATEGORY_NAME: Default

    UGP_DISPLAY_NAME: Defaults to UGP_NAME

    ORG_NAME: TOP

    INCLUDE_HIERARCHY: YES
  - For Role Hierarchy (GPG)

    None
  - 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.

24.7.1.2 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 24–4 shows the structure of a sample database role table.

Table 24–4 Structure of a Sample Database Table

Role	NULL	Type
UGP_ROLENAME	NOT NULL	VARCHAR2(2000)
UGP_NAMESPACE		VARCHAR2(512)
ORG_NAME	NOT NULL	VARCHAR2(256)
INCLUDE_HIERARCHY	NOT NULL	VARCHAR2(256)
...	...	...
OIM_BLKLD_USRSEQ	NOT NULL	NUMBER(19)

Note: ORG_NAME and INCLUDE_HIERARCHY are required for loading roles only, and not for role hierarchy, role membership, and role category.

24.7.1.3 Determining the UGP_NAME Generated After Role Load

Bulkload utility generates UGP_NAME during role load in the following format:

```
UGP_NAMESPACE.UGP_ROLENAME
```

By default, the value of UGP_NAMESPACE is Default, when you do not provide any specific value for UGP_NAMESPACE in the CSV file. To determine the generated UGP_NAME:

1. If UGP_NAMESPACE is null in the CSV file, then the namespace value is Default, and the generated UGP_NAME is equal to the value of UGP_ROLENAME.
2. If UGP_NAMESPACE is not null and has a defined value in the CSV file, then the generated UGP_NAME is equal to the value of UGP_NAMESPACE.UGP_ROLENAME.

On the basis of the UGP_NAME generation methodology, you can determine the UGP_NAME values for the next loading of role hierarchy, role membership, and role category, even if you do not have direct access to the database. Otherwise, you can check the generated value of UGP_NAME in the UGP table.

24.7.2 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. Enter the password twice when prompted.

- 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 24-26 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 24-28 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. Note that the OIM_BLKLD_USRSEQ column created in the source table must not be listed as a control line column.

- 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 24-5 for more information.

- Date format

  Date format used by date columns in the CSV files. This is prompted only for role load, and not for role hierarchy, role membership, and role category.

  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 24-32 for more information.

24.7.3 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;
```

24.7.4 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 24–3 shows sample statistics recorded in the log file at the end of a bulk load operation to store OIM Role data.

Example 24–3 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 24-31 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.

24.7.5 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 24-6 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.

   Note: Being a database-intensive operation by design, Bulk Load disables the constraints and indexes on the relevant Oracle Identity Manager entity tables during the start of the operation. Bulk Load operation failure towards the end of the load might at times render the indexes and constraints in disabled state. To identify and fix this issue, manually restore the indexes and constraints as follows:

   1. Identify the unusable indexes and disabled constraints. To do so, the following SQL queries or similar mechanism can be used:

      ```
      SELECT TABLE_NAME, CONSTRAINT_NAME FROM user_constraints WHERE
      status = 'DISABLED';
      SELECT index_name FROM user_indexes WHERE status = 'UNUSABLE';
      ```

   2. Enable the constraints and rebuild the indexes manually, as shown:

      ```
      ALTER TABLE TABLE_NAME ENABLE CONSTRAINT CONSTRAINT_NAME;
      ALTER INDEX INDEX_NAME REBUILD;
      ```

24.7.6 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 Self Service, 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.

24.8 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 24–5 describes the structure of the OIM_BLKLD_LOG table.

Table 24–5 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.
MODULE	NOT NULL	VARCHAR2(20)	This column stores one of the following values: ROLE: This value indicates that the message has been recorded while loading OIM Role data. ROLE HIERARCHY: This value indicates that the message has been recorded while loading role hierarchy data. ROLE MEMBERSHIP: This value indicates that the message has been recorded while loading OIM role membership data. ROLE CATEGORY: This value indicates that the message has been recorded while loading OIM role category data.
LOG_LEVEL	NOT NULL	VARCHAR2(20)	This column stores one of the following values: **ERROR:** Designates fine-grained informational events that are useful to debug. **DEBUG:** Designates error events that might allow the application to continue running. Error is used to log all unhandled exceptions. **PROGRESS_MSG:** Designates intermediate progress messages.
LOAD_SOURCE	NOT NULL	VARCHAR2(40)	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: CSV File: *FILE_NAME* DB Table

Table 24–5 (Cont.) Structure of the OIM_BLKLD_LOG Table

Column	NULL	Type	Description
MSG	NOT NULL	VARCHAR2(4000)	This column stores a message corresponding to the value stored in the LOG_LEVEL column.
CREATE_DATE		DATE	This column holds the time stamp at which the record was created. The format for entries in this column is as follows: yyyy/mm/dd hh24:mi:ss For example: 2008/06/23 21:49:16:32

24.9 Gathering Diagnostic 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 24–5, " Structure of the OIM_BLKLD_LOG Table" for information about the log levels that stores error events

24.10 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 Diagnostic 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.

24.11 Bulk Load High Volume Strategy and Case Studies

For information about general best practices and few case studies about high-volume data load, see the Tech note titled *OIM 11G BulkLoad Utility Strategies & Case Studies* (**Doc ID 1959363.1**) in the My Oracle Support web site at:

https://support.oracle.com

25

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 11*g* Release 2 (11.1.2.2.0), 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 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.

26

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:

26.1 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.

26.1.1 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 26.2, "Defining the Metadata for the Scheduled Task".

3. Add the task metadata to the scheduled task XML file.

   For details, see Section 26.3, "Configuring the Scheduled Task XML File".

4. Develop the scheduled task Java class.

   For details, see Section 26.4, "Developing the Scheduled Task Class".

5. Declare the new scheduled task as a plug-in.

   For details, see Section 26.5, "Configuring 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 26.6, "Creating the Directory Structure for the Scheduled Task".

26.1.2 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.

26.2 Defining 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

26.3 Configuring 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 27, "Developing Plug-ins" for examples of plug-ins.

Note: The scheduled task XML file can be imported into MDS using the Oracle Enterprise Manager. 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 "Migrating User Modifiable Metadata Files" on page 37-1.

The elements in the XML file reflect the task parameters that you described in Section 26.2, "Defining the Metadata for the Scheduled Task".

Example 26–1 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 26–1 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" 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:** "Scheduled Task Configuration File" on page 26-6 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.

> **Note:** For a task defined in a plugin, the metadata XML is not required to be seeded to MDS. This can be included in the META-INF folder in the plugin ZIP file. For details, see "Creating the Directory Structure for the Scheduled Task" on page 26-5.

You must export the task.xml file from MDS, add the required tags to the file, and then import it back to MDS. See "Migrating User Modifiable Metadata Files" on page 37-1 for information about exporting and importing MDS files.

26.4 Developing 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. The Java class must also override the other abstract methods:

   ```
   public HashMap getAttributes();
   public void setAttributes();
   ```

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.

26.5 Configuring 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 27, "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.

26.6 Creating 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 26–2. In this example, a single plug-in is being added, but there can be multiple plugins 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 System Administration while creating jobs.

Example 26–2 Directory Structure for the Scheduled Task

```
plugin/
      lib/
      PLUGIN.JAR
      plugin.xml
      META-INF (optional)
            METADATA.xml
```

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 27-6 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.
- The META-INF folder is an optional folder, in which metadata (task definition) file can be stored. If this file is placed in the META-INF folder, then it is not required to be seeded in MDS.
- If META-INF folder does not exist or if the metadata file is not placed in the META-INF folder, then seed the file to MDS.

In the preceding example, *CLASS_NAME*.JAR is the JAR file that you create in Section 26.4, "Developing 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 27-7 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.

26.7 Scheduled Task Configuration File

This appendix describes the structure and details of the XML file containing scheduler task definitions.

26.7.1 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>
```

26.7.2 The scheduledTasks Element

The scheduledTasks element is the root element in XML used to define scheduled tasks.

Table 26–1 summarizes the properties of the scheduledTasks element.

Table 26–1 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: <scheduledTasks xmlns="http://xmlns.oracle.com/oim/scheduler"> **Note:** The xmlns parameter is mandatory.
Child Elements	task
Number of Occurrences	One for each scheduled task XML file to be created.
Element Value	NA
Mandatory or Optional?	Mandatory

26.7.3 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 26–2 summarizes the properties of the task element.

Table 26–2 Properties of the task Element

Property	Value
Parent Element	scheduledTasks
Attributes	None
Child Elements	name, class, description, retry, and parameters

Table 26–2 (Cont.) Properties of the task Element

Property	Value
Number of Occurrences	One for each task to be created. **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.
Element Value	NA
Mandatory or Optional?	Mandatory

26.7.4 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 26–3 summarizes the properties of the name element.

Table 26–3 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

26.7.5 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 26–4 summarizes the properties of the class element.

Table 26–4 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 "Developing the Scheduled Task Class" on page 26-4 for information on developing a class for the scheduled task.
Mandatory or Optional?	Mandatory

26.7.6 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 26–5 summarizes the properties of the description element.

Table 26–5 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

26.7.7 The retry Element

Table 26–6 summarizes the properties of the retry element.

Table 26–6 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

26.7.8 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 26–7 summarizes the properties of the parameters element.

Table 26–7 Properties of the parameters Element

Property	Value
Parent Element	task
Attributes	None
Child Elements	string-param, number-param, boolean-param
Number of Occurrences	One

Table 26–7 (Cont.) Properties of the parameters Element

Property	Value
Element Value	NA
Mandatory or Optional?	Optional

26.7.9 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 26–8 summarizes the properties of the string-param element.

Table 26–8 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
Element Value	Name of the string parameter
Mandatory or Optional?	Optional

As listed in Table 26–8, 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.

26.7.10 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 26–9 summarizes the properties of the number-param element.

Table 26–9 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 require and helpText attributes for the number-param and string-param elements is the same. See "The string-param Element" on page 26-10 for information about the require and helpText attributes.

26.7.11 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 26–10 summarizes the properties of the boolean-param element.

Table 26–10 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 "The string-param Element" on page 26-10 for information about the require and helpText attributes.

26.8 Best Practices for Creating Custom Scheduled Tasks

Table 26–11 provides the guidelines for using variables/constants for creating custom scheduled tasks:

Table 26–11 Variables and Constants for Creating Custom Scheduled Tasks

Type	Example	Stor/Retrieve Value From
Target system connection details	Hostname, port number, SSL	IT Resource/application instance
Target system configurations	Attribute mappings, Unique Attribute, User Object Class	Lookup
Scheduled job-specific variables/constants	Application Instance Name, IT Resource Name, File Path, Search Filter, Batch Size, Retries	Scheduled job

Table 26–11 (Cont.) Variables and Constants for Creating Custom Scheduled Tasks

Type	Example	Stor/Retrieve Value From
Scheduled job advanced configuration variables/constants	Attribute Mappings, Target system Date Format, Constants, Attribute Transformation Classes	Lookup
Oracle Identity Manager-specific system wide highly static configuration properties/constants/variables	Default Date Format, Default policy for username generation	System properties
Email notifications	Subject, Body, To, From	Email templates

26.9 Using the isStop() Method

When a job is stopped from the Scheduler section in Oracle Identity System Administration, the job does not stop and keeps running. To stop the scheduled task, you can perform the following:

If you have developed a custom scheduled task, then you can call the isStop() or isStopped() method at various stages inside the execute method. If this method returns true, then return from the execute method. If you have loops inside the execute method, then make sure that the isStop() or isStopped() method is called for each loop iteration.

In the execute method, add a check for getting the job status. This can be obtained by calling the isStopped() method of the com.thortech.xl.scheduler.tasks.SchedulerBaseTask class. If the isStopped() method returns TRUE, then return from the execute method without performing any execution for the scheduled task. The following is the code snippet for this:

```
if(isStopped())
     return;
```

If you develop a custom scheduled task by extending the TaskSupport class in Oracle Identity Manager 11*g* Release 1 (11.1.1) or 11*g* Release 2 (11.1.2), then call the isStop() method in the execute method.

If the custom scheduled task code is extending legacy com.thortech.xl.scheduler.tasks.SchedulerBaseTask class of Oracle Identity Manager Release 9.x, then call the isStopped() method in the execute method.

Part VI

Custom Operations

This part contains chapters that describe how to develop customized operations in the Oracle Identity Manager.

It contains the following chapters:

- Chapter 27, "Developing Plug-ins"
- Chapter 28, "Developing Event Handlers"
- Chapter 29, "Understanding Context"

27

Developing Plug-ins

This chapter describes the concepts related to plug-in and how to develop and use a plug-in the following sections:

- Plug-ins and Plug-in Points
- Using Plug-ins in Deployments
- Plug-in Points
- Configuring Plug-ins
- Developing Custom Plug-ins
- Registering Plug-ins
- Migrating Plug-ins

27.1 Plug-ins and Plug-in Points

A *plug-in* is a logical component 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. You can extend the plug-in interface based on the business requirements and register them as plug-ins. To do this, you develop a Plugin Java class and compile it before archiving in a JAR file, define plug-in metadata in an XML file, and ZIP these artifacts as a plug-in package that is ready to deploy.

For example, user creation is a business operation in Oracle Identity Manager. But this operation exposes a plug-in point for user name generation. If you want to model your custom logic of user name generation, then you must identify the plug-in point specifications and develop a plug-in accordingly.

The concepts related to plug-ins are described in the following sections:

- Plug-ins and Event Handlers
- Plug-in Stores

27.1.1 Plug-ins and Event Handlers

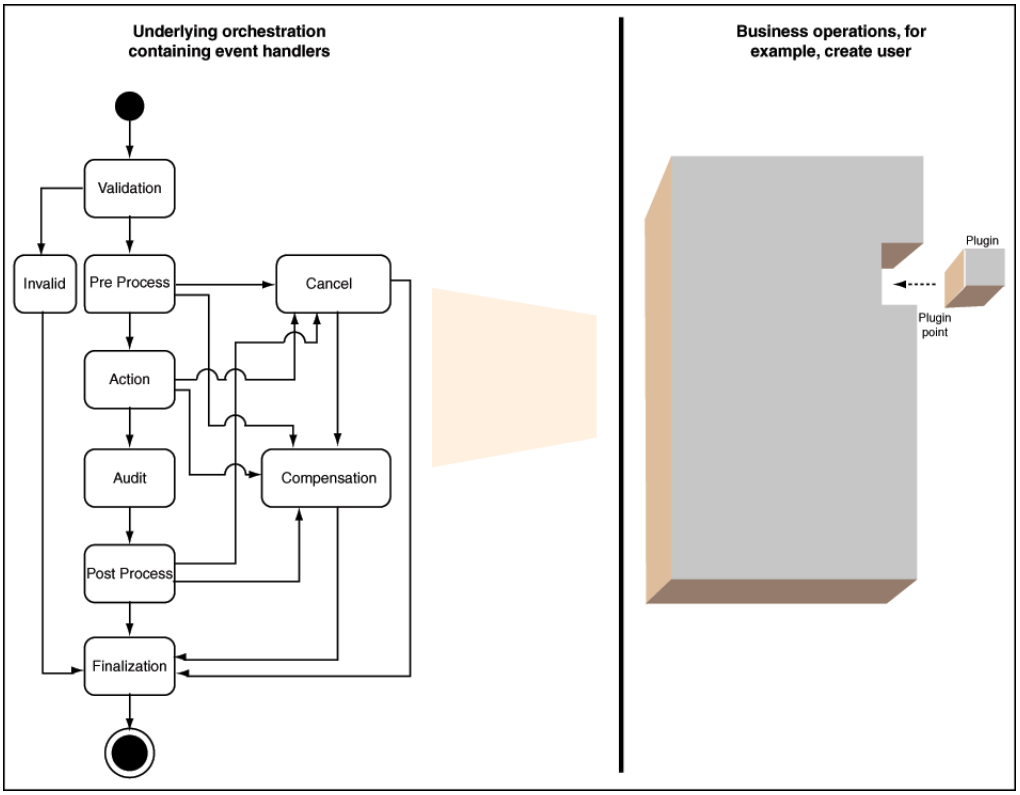
Most of the business operations in Oracle Identity Manager, such as user creation, role assignment to user, and user activation, are executed as orchestrations. Therefore, if

there is a requirement to induce any custom logic in these operations or orchestrations, then you can model that logic as event handlers at stages, such as validation, preprocess, and postprocess, in which customization is supported.

However, you can analyze if any such operation also exposes a plug-in point for inducing the custom logic. If a plug-in point is available, then you can utilize the plug-in point rather than operating the underlying orchestration. For example, you can implement username generation by using the exposed plug-in without writing that as an event handler in the create user orchestration.

Figure 27–1 shows a diagrammatic representation of plug-ins and event handlers.

Figure 27–1 Plug-ins and Event Handlers



27.1.2 Plug-in Stores

The plug-in framework can store plug-ins in two types of stores:

- File system
- The Oracle Identity Manage database

When looking for plug-ins, the framework first examines plug-ins registered in the database, and looks in the file system.

27.1.2.1 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 "Configuring Plug-ins" on page 27-5.

Note: Oracle recommends not to use the file store in production. File store is more suitable during plug-in development because it is easy to change the plug-in, and you are required to change only the file in the file system. There is no need to register. However, in production, plug-ins are not changed often, and therefore, avoid using the file store because of certain disadvantages. It adds the overhead of file store monitoring. In addition, the plug-ins are required to be replicated in all nodes of a cluster for the clustered deployment of Oracle Identity Manager.

27.1.2.2 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 "Registering and Unregistering Plug-ins By Using APIs" on page 27-7 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.

27.2 Using Plug-ins in Deployments

As already mentioned in this document, plug-ins are used for customizing the default functionality in an Oracle Identity Manager deployment. The number of supported plug-in points is a defined and constrained set. Therefore, you can use the plug-in points to extend the functionality only for the list of supported plug-in points. See "Plug-in Points" on page 27-3 for a list of the supported plug-in points.

27.3 Plug-in Points

Table 27–1 lists the Java interfaces that act as plug-in points in Oracle Identity Manager:

Table 27–1 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 28, "Developing Event Handlers" for information about kernel event handlers.

oracle.iam.platform.auth.api.LoginMapper	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.
	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:
	`public class CustomLoginMapper implements LoginMapper{`
	`public String getOIMUserID(String jaasPrincipal) throws MappingException {`
	`return getUserName(jassPrincipal);`
	`}`
	
	`private String getUserName(String emailID){`
	`String userName = null;`
	
	`//Use usermgmt APIs to get the username corresponding to this email id`
	`return userName;`
	`}`
	`}`

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.
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.

27.4 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.

27.5 Developing Custom Plug-ins

This section describes how to develop custom plug-ins. It contains the following sections:

- Developing Plug-ins
- Declaring Plug-ins

27.5.1 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 27-7 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>);
     ```

   - **The META-INF/ directory:** Contains XML files showing plug-in points for event handlers. Some services, such as the notification service, read the XML files from MDS or from the META-INF/ directory of the plug-in.

   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 (the *OIM_HOME*/plugins/ directory) or database store.
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 27-7.

27.5.2 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. By default, all of the current plug-in points load the latest version of the plug-ins.

The XML shows two plug-in declarations. Both the plug-ins extend from the same plug-in point.

27.6 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

27.6.1 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:

```
System.out.println("Creating client....");
String ctxFactory = "weblogic.jndi.WLInitialContextFactory";
String serverURL = "t3://OIM_HOSTNAME:OIM_PORT";
System.setProperty("java.security.auth.login.config", "OIM_CLIENT_
HOME/conf/authwl.conf");
String username = "USER_NAME";
char[] password = "PASSWORD".toCharArray();
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);
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);
```

Note: "Using OIMClient" on page 31-1 for information about using OIMClient for developing clients to integrate with Oracle Identity Manager.

27.6.2 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
   ```

In addition, set the path for mw.home in the ant.properties file. Also, uncomment the following:

```
#login.config=${oim.home}/config/authwl.conf
```

2. Build the wlfullclient.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":

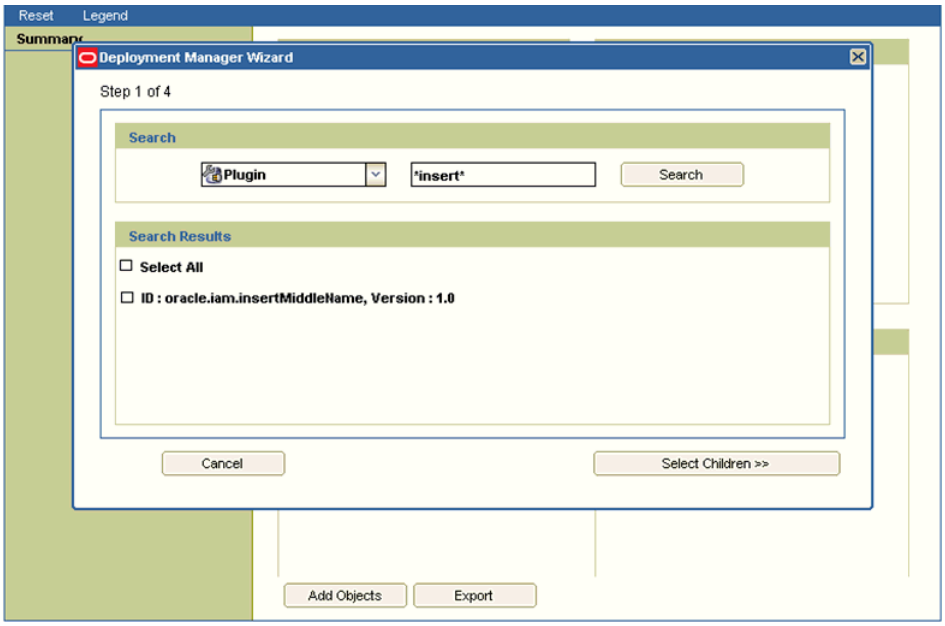
```
ant -f  pluginregistration.xml unregister
```

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.

27.7 Migrating Plug-ins

The Deployment Manager supports migrating plug-ins from one deployement of Oracle Identity Manager to another. For example, the event handlers can be implemented in a test environment, and then migrated to the production environment by using the Deployment Manager. Figure 27–2 shows exporting plug-ins via the Deployment Manager:

Figure 27–2 Exporting Plug-ins



See Also: "Migrating Configurations and Customizations" on page 38-1 for information about the Deployment Manager

28

Developing Event Handlers

This chapter describes the concepts related to orchestration and how to write custom event handlers to extend the functionalities of Oracle Identity Manager. It contains the following topics:

28.1 Orchestration Concepts

In an Identity Management system, any action performed by a user or system is called an operation. Examples of operations are creating users, modifying roles, and creating password policies. The process of any Oracle Identity Manager operation that goes through a predefined set of stages and executes some business logic in each stage is called an *orchestration*. The type of object that is changed by the orchestration is called an orchestration target. The data that is required to carry out the orchestration operation is called orchestration parameter.

A bulk orchestration is the process of orchestrating same operation on multiple entities. For example, if you want to update the organization of multiple users, then you can submit a bulk orchestration. As a result, the operation on all the entities are performed in a single call.

> **Note:** If custom event handlers are required to be introduced for lock/unlock operations, then you must implement bulk orchestrations. From the UI, bulk orchestrations are triggered for a single user lock/unlock operation.

Orchestration is divided into predefined steps called stages. Every operation moves through these stages until it reaches finalization. Orchestration has the following 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.
- **Preprocess:** Stage to perform orchestration parameter manipulations or get approvals or perform Segregation of Duties (SoD) checks.
- **Action:** Stage in which the action takes place.
- **Audit:** Stage in which the auditing of operation is performed.
- **Postprocess:** 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.
- **Finalization:** Last stage in the process to perform any clean up.

Each operation performed can have consequences for users or other entities. For example, creating a user might result in provisioning of resources to that user, and creating a new password policy can make certain user passwords invalid and require changes during next login. Each consequence is represented as an orchestration. A differed consequence is executed before the finalization of the current orchestration. An immediate consequence is executed immediately after the current event handler returns, before proceeding to the next event handler on the current orchestration. You can customize the consequences of some operations, such as create, modify, delete, enable, disable, lock, and unlock users, by writing event handlers, as described in subsequent sections.

There are orchestrations for which the starting point is the postprocess stage. If you are reconciling users from a trusted source or bulk loading users and want to add this data as is in Oracle Identity Manager. When the data is in Oracle Identity Manager, you can perform postprocess operations on the users to compute autogroup membership or evaluate policies. Therefore, reconciliation engine or bulk load utility submits postprocess-only orchestrations.

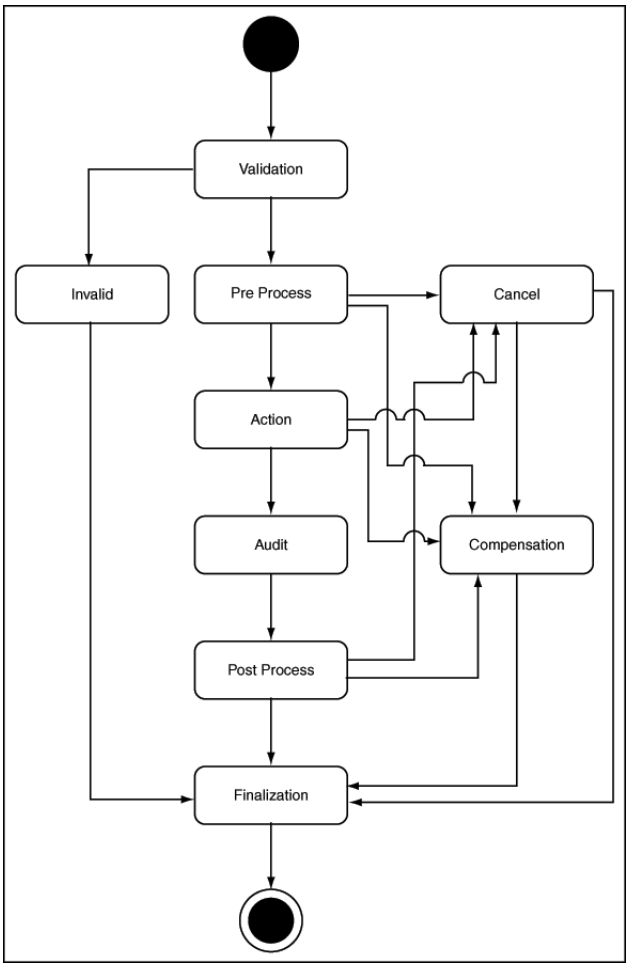
An *event handler* is a piece of code that is registered with an orchestration on various stages. These event handlers are invoked when the relevant orchestration stage is performed. Event handlers can either be asynchronous or synchronous. A synchronous event handler returns with a response right away, whereas an asynchronous event handler completes at a later stage. An event handler can be conditional, which means that the event handler is executed when certain conditions are satisfied.

What happens at each stage of orchestration 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, then the operation follows the default path and moves to the next stage. However, a process can move to some out-of-the-band stages if the event handlers are invalid or canceled. These stages are:

- **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.

Figure 28–1 shows the various orchestration stages:

Figure 28–1 Orchestration Stages



28.2 Using Custom Event Handlers

Oracle Identity Manager allows you to implement Service Provider Interfaces (SPIs) to customize the functionality of orchestration operations. Only customization of preprocess, postprocess, validation, and finalization stages of an operation in an entity orchestration is supported.

The following are examples of event handler implementation:

- When a user is created, the account status (enabled or disabled) is to be set based on some rules. A preprocess event handler can be implemented to achieve this.
- Users of type Contractors must have an email address at the time of creation. Other users can be created without email address. A validation event handler can be used to validate if the user is a Contractor, and then allow or disallow the user creation based on the validation result.
- Users of type Agents are to be notified in the user's alternate email address after the users are created. This can be achieved by implementing a postprocess event handler.

Postprocess event handlers are most commonly implemented to meet business requirements. The following example describes how a postprocess event handler implementation can meet the given requirement:

Requirement

If the enterprise user is a Contractor, then after the user is created in Oracle Identity Manager, the user must be registered in the Contractor Registration System, which is an external application. This application is a database application. The database has a structure that stores the User ID, Contractor ID, First Name, and Last Name attributes of the users. After successful registration, the Contractor ID of the users must be retrieved and updated in the user's profile in Oracle Identity Manager.

Solution

This use case can be developed as a plug-in and deployed on Oracle Identity Manager. The plug-in can be used to retrieve the Contractor ID or any configured column name from specified database table and update the user profile in Oracle Identity Manager.

A postprocess event handler can be implemented and registered for the create operation of the user entity. It is a conditional event handler that executes for users only with type as Contractor. If the user type is Contractor, then the event handler connects to the external application to retrieve the Contractor ID based on the Oracle Identity Manager user ID, and update the user profile in Oracle Identity Manager with contractor ID.

The following is another common example of postprocess implementation of event handlers:

Custom attribute generation if the data that is reconciled into Oracle Identity Manager is not enough to implement all use cases and extra attributes need to be generated based on the reconciled data. This is a common use case, especially when the custom attributes are used in the role membership rules or access policies.

28.3 Developing Custom Event Handlers

An event handler consists of the following:

- **Java code:** Implementation of the operations
- **XML definition:** Association with the relevant orchestration at the right stage
- **Plug-in definition:** Registration of the event handlers and any extension code with Oracle Identity Manager plug-in framework

Developing a custom event handler comprises of implementing the operation through Java code, writing the XML definition, and creating and registering a plug-in. These are described in the following sections:

- Implementing the SPI and Creating a JAR
- Defining Custom Events Definition XML
- Creating and Registering a Plug-in ZIP

28.3.1 Implementing the SPI and Creating a JAR

This section describes how to write the JAVA code by implementing the SPI, and thereafter, create a JAR file in the following sections:

- Development Considerations

- Methods and Arguments
- Code Samples
- Creating a JAR File With Custom Event Handler Code

28.3.1.1 Development Considerations

The following points must be considered for writing custom event handlers:

- The supported orchestration stages in which a custom event handler can be registered are validation, preprocess, and postprocess.
- Validation, preprocess, and postprocess event handlers can be conditional. This means that the event handler will execute only if a particular condition is met.

  You can make the event handler conditional by implementing the oracle.iam.platform.kernel.spi.ConditionalEventHandler interface and its isApplicable method. Context data and orchestration parameters are available in this method. For conditional event handlers, the applicability of event handlers is computed when the operation is initiated. Therefore, if a context or orchestration parameters are modified during the orchestration flow, then it might lead to execution of event handlers that must not be executed.
- The event handlers can handle single as well as bulk entities.
- The event handlers can have associated failure handlers that callbacks certain operations on the parent handlers.
- Because retry of event handlers is supported, the event handlers can be re-entrant.
- When reconciliation submits postprocess orchestrations, it submits bulk orchestrations. The bulkExecute method on the event handlers is called for these orchestrations. Therefore, make sure to implement this method.
- If data is to be passed between custom event handlers, you can pass it by using inter event data. Calling the getInterEventData() method on orchestration returns a hashmap. In this map, you can put any object with key beginning with custom, and you can access this data in subsequent custom handlers. Do not modify or delete any predefined inter event data that is part of the same hashmap.
- To make API calls inside event handlers for write or delete operations, get the API services by using Platform.getServiceForEventHandlers method. API calls that are made using the services obtained through this method are performed synchronously including the postprocessing.
- Return type of event handlers, except validation handlers, are shown in the following table:

Event Handler Type	On Success	On Failure
Synchronous	new EventResult() in the execute method and new BulkEventResult() in bulk version of the execute method	EventFailedException
Asynchronous	Return null	EventFailedException

- You must not define object-level variables at the event handler.

28.3.1.2 Methods and Arguments

Table 28–1 lists the methods that you can implement in the various orchestration stages:

Table 28–1 Methods to Implement Event Handlers

Method	Applicable Orchestration Stage	Description
initialize	preprocess, postprocess	This method is used to open connections and pool state or resources.
execute for single entity	preprocess, postprocess	This method is used to read the input attributes of the underlying operation and update to different values, if required.
execute for bulk orchestration	preprocess, postprocess	This method is used to read the input attributes of multiple underlying operations and update to different values, if required.
isApplicable	conditional	This method is used in conditional handlers to determine if the prerequisite condition for the event handler execution is met.
validate	validation	This method is used for validation handlers to validate input data.
cancel	preprocess, postprocess	This method is called when the orchestration operation is canceled.
compensate	preprocess, postprocess	This method is called when the orchestration operation is compensated.

For methods, such as execute, the following argument values are available:

- IDs that you can include in the code for troubleshooting purpose, which includes:
  - **Process ID:** The ID of the orchestration instance
  - **Event ID:** The ID of the event handler instance
- Orchestration object that consists of details of the underlying entity instance. This consists of:
  - Maps (key value pairs) containing *ENTITY_ATTRIBUTE*, *VALUE* from which the input attributes of the underlying entity is read.
  - Entity ID: To update back the values for the same or a different entity, use Entity Manager API and pass the Entity ID and data to it. For bulk orchestration, you get multiple Entity IDs and Maps.

Note: Use Platform.getServiceForEventHandlers to get the services for calling create, update, and delete operations in event handlers.

28.3.1.3 Code Samples

This section provides code samples that illustrate how to write various kinds of event handlers.

Example 1: Custom Email Validation

Example 28–1 shows a sample custom validation handler code fragment that checks to ensure that the ampersand character (@) is used in the email id of the user.

Example 28–1 Custom Email Validation

```
public void validate(long processId, long eventId, Orchestration orchestration) throws
ValidationException, ValidationFailedException {
```

```
    HashMap<String, Serializable> parameters = orchestration.getParameters();
    String email = (parameters.get("Email") instanceof ContextAware) ? (String) ((ContextAware)
parameters
                .get("Email")).getObjectValue() : (String) parameters
                .get("Email");
        if (!(email.contains("@"))) {
             throw new ValidationFailedException("Email doesn't contain @");
         }
    }
```

Example 2: Custom Preprocess Event Handler to Set Middle Name

Example 28–2 shows a sample custom preprocess event handler code fragment that sets the middle name to the first letter of the first name if the a value is not provided for middle name.

Example 28–2 Custom Preprocess Event Handler to Set Middle Name

```
// 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 ((middleName == null) || middleName.equals("")) {
            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) {
            if(parameters.containsKey(key)){
    String value = (parameters.get(key) instanceof ContextAware) ? (String) ((ContextAware)
parameters
                .get(key)).getObjectValue() : (String) parameters.get(key);
    return value;
            }
            else{
                return null;
            }
    }
```

Example 3: Custom Post-process Event Handler to Provision Resource Object

Example 28–3 shows a sample custom post process event handler code fragment that provisions a resource object OBJ005 to a user whose role is ROLE 005:

Example 28–3 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(null, "Error occurred", null, null, 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<Role> roles = roleManager.getUserMemberships(userKey, true);

  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;
}
```

Example 4: Custom User Postprocess Event Handler With bulkExecute Method

Example 28–4 shows how to loop through users that are part of a bulk user create orchestration.

Example 28–4 Custom User Postprocess Event Handler With bulkExecute Method

```
public BulkEventResult execute(long processId, long eventId, BulkOrchestration
orchestration){

HashMap<String, Serializable>[] orchParamArray =
orchestration.getBulkParameters();

       // Array of user keys
        String [] entityIds = orchestration.getTarget().getAllEntityId();
        for(int i=0; i< entityIds.length; i++){
        }

}
```

Example 5: Using Context in isApplicable method

Any operation in Oracle Identity Manager can take place in more than one context. For example, creating a user can happen in four different contexts, which are administrator creating a user as a direct operation, administrator creating a user by raising a request, creating a user through self registration, and user creation through trusted source reconciliation. In all these scenarios, Oracle Identity Manager submits the same user creation orchestrations having the same parameter names and values with same data types.

Example 28–5 shows how to find the context in which this operation is performed to figure out the applicability of the event handler.

Example 28–5 Using Context in the isApplicable Method

```
public boolean isApplicable(AbstractGenericOrchestration orchestration) {    //
Request Context
    if (ContextManager.getContextType() == ContextTypes.REQUEST) {
    }
    // Recon context
    if (ContextManager.getContextType() == ContextTypes.RECON) {
    }


}
```

> **See Also:** "Understanding Context" on page 29-1 for details about the information that can retrieved from context

28.3.1.4 Creating a JAR File With Custom Event Handler Code

To create a JAR with custom event handler code:

1. Implement one of the SPIs mentioned in Table 28–2 to write a custom preprocess, postprocess, or validation handler.

Table 28–2 SPIs to Write Custom Event Handlers

Stage	SPI to implement
Preprocess	oracle.iam.platform.kernel.spi.PreProcessHandler
Postprocess	oracle.iam.platform.kernel.spi.PostProcessHandler
Validation	oracle.iam.platform.kernel.spi.ValidationHandler
Finalization	oracle.iam.platform.kernel.spi.FinalizationHandler

> **See Also:** See *Oracle Fusion Middleware Java API Reference for Oracle Identity Manager* for information about the SPIs listed in Table 28–2

2. Include the following JAR files in the class path to compile a custom class:

   From the *OIM_ORACLE_HOME*/server/platform/ directory:

   - iam-platform-kernel.jar
   - iam-platform-utils.jar
   - iam-platform-context.jar
   - iam-plaftorm-authz-service.jar

   From the *OIM_ORACLE_HOME*/designconsole/lib/ directory:

   - oimclient.jar
   - xlAPI.jar

   If some other Oracle Identity Manager JAR files are required for compilation, then these can be found in the directories mentioned in this step.

3. Create a JAR file of the custom class.

28.3.1.5 Handling Exceptions

For event handler exception handling, you must use conventional JAVA exception handling methods. The following guidelines can be used for dealing with failures:

- In the event handler code, throw EventFailedException with the right arguments to indicate failure.
- Failures can be handled by registering failure handlers. As part of failure handler, you can implement necessary logic to remediate the failure. The failure handlers must return FailedEventResult with the following options as Response:
  - **CANCEL:** Indicates that operation must get canceled. The Cancel method on all event handlers that are executed and completed so far is called by Kernel in reverse order of execution.
  - **COMPENSATE:** Indicates that operation must get rolled back. The Compensate method on all event handlers that are executed and completed so far is called by Kernel in reverse order of execution.
  - **MANUAL_COMPLETE:** Indicates that the handler that failed is manually completed and will proceed with the rest of the event handlers.
  - **RETRY:** Indicates to kernel that the event handler that failed must be retried.

– **NULL:** Indicates that there is no response or recommendation by the failed handler.

28.3.1.6 Managing Transactions

In the event handler XML file, set the tx attribute to true. If any exception is thrown in the event handler, then the transaction will be rolled back or committed.

28.3.2 Defining Custom Events Definition XML

The custom events definition XML is described in the following sections:

- Elements in the Event Handler XML Files
- Sample Event Definitions

28.3.2.1 Elements in the Event Handler XML Files

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 28–3 lists and describes sub-elements that are typically defined within the `eventhandlers` parent element.

Table 28–3 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 in the default flow 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 in parent orchestration upon consequence orchestration failures.
`out-of-band-handler`	Defines the event handlers for out-of-band orchestration flows, such as veto and cancel.
`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 28–4 lists and describes attributes that are typically defined within elements.

Table 28–4 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.

Table 28–4 (Cont.) Typical Attributes of Sub-elements within the eventhandlers Element

Element Attribute	Description
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. Most commonly defined entity types are user, role, rolerole (role hierarchy), and roleuser (user role membership).
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. Typical operations are create, modify, and delete.
order	Identifies the order (or sequence) in which the event handler is executed. Order value is in the scope of entity, operation, and stage. Order value for each event handler in this scope must be unique. If there is a conflict, then the order in which these conflicted event handlers are executed is arbitrary. Supported values are `FIRST` (same as Integer.MIN_VALUE), `LAST` (same as Integer.MAX_VALUE), or a numeral.
orch-target	Identifies the type of orchestration, such as entity orchestration, Toplink orchestration, and so on. The following is a list of supported values: ■ `oracle.iam.platform.kernel.vo.EntityOrchestration` ■ `oracle.iam.platform.kernel.vo.MDSOrchestration` ■ `oracle.iam.platform.kernel.vo.RelationOrchestration` ■ `oracle.iam.platform.kernel.vo.ToplinkOrchestration` The default value is `oracle.iam.platform.kernel.vo.EntityOrchestration`. This is the only supported type for writing custom event handlers.
sync	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), then the kernel expects the event handler to return an EventResult. If set to `FALSE` (asynchronous), then you must return null as the event result and notify the kernel about the event result later. **Note**: The sync attribute must be set to `TRUE` for validation-handler elements.
stage	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 is executed. The following is a list of supported values: ■ preprocess ■ action ■ audit ■ postprocess ■ veto ■ canceled
tx	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. By default, the value is FALSE.

Namespace Requirement in <eventhandlers> Element

All the event handler definitions must have the following mandatory namespace definition:

```
<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">
```

28.3.2.2 Sample Event Definitions

Create a metadata XML file containing definitions of all the custom events, as shown in Table 28–6:

Example 28–6 Sample Metadata XML File for Custom Event Definitions

```
<?xml version='1.0' encoding='utf-8'?>
    <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">

  <!-- Custom preprocess event handlers -->
  <action-handler
    class="oracle.oim.extensions.preprocess.SamplePreprocessExtension"
    entity-type="User"
    operation="CREATE"
    name="SetUserMiddleName"
    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="CustomResourceProv"
    stage="postprocess"
    order="1000"
    sync="TRUE"/>

  <!-- Custom validation event handlers -->
   <validation-handler
    class="oracle.oim.extensions.validation.SampleValidationExtension"
    entity-type="User"
    operation="CREATE"
    name="ValidateUserEmail"
    order="1000"/>
</eventhandlers>
```

28.3.3 Creating and Registering a Plug-in ZIP

To create plug-ins containing custom event handlers, you need to develop the appropriate event handler classes. See "Developing Plug-ins" on page 27-1 for detailed information about plug-ins and plug-in points.

To create a plug-in ZIP and register it:

1. Define the plug-in XML with the event handler plug-in point.

   Note: Ensure that plug-in point used in the plug-in definition is set to oracle.iam.platform.kernel.spi.EventHandler.

   The following is an example of a plug-in XML file:

```
<?xml version="1.0" encoding="UTF-8"?>
<oimplugins>
  <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>
</oimplugins>
```

2. Package the plug-in XML and the JAR file that contains the custom class or classes into a plug-in ZIP file.
3. Package the event handler XML that is defined using the information described in "Defining Custom Events Definition XML" on page 28-11 into the same zip in a directory called META-INF.
4. Register the plug-in by using plug-in registration utilities. See "Registering Plug-ins" on page 27-7 for additional information.

28.4 Sequencing the Execution of Event Handlers

The list of custom event handlers that you deployed and registered can be viewed by using Oracle Enterprise Manager. The event handlers are displayed in the order of invocation. Using this list of event handlers, you can sequence the order of execution of the event handlers.

To specify the order for any custom event handler, you must know the list of existing event handlers and their order for a given operation. To do so, you must invoke a mbean from the Enterprise Manager by performing the following steps:

1. Login to the Enterprise Manager.
2. On the left navigation pane, expand Weblogic Domain, and select OIM DOMAIN.
3. Right-click the domain name, and select **System Mbean Browser**.
4. Under Application Defined Mbeans, expand oracle.iam.
5. Navigate to ***OIM_SERVER_NAME***, **oim**, **IAMAppDesignMBean**, **ConfigQueryMBeanName**.
6. Click the **Operations** tab.
7. Click the getEventHandlers method.
8. Provide entity name for the p1 parameter and operation name for the p2 parameter, and then click **Invoke**. The parameter values are not case-sensitive. The possible parameter values are:
   - entity name: Values can be User, Role, or RoleUser
   - operation: Values can be CREATE, MODIFY, or DELETE

28.5 Writing Custom Validation Event Handlers

An approver can update the attribute values before approving a request. To ensure sanitization of the data entered by the approver, Oracle Identity Manager invokes validation handlers again when approver updates the request. This means that validation handlers configured for a particular entity and operations are invoked multiple times in a single request flow, when the request is submitted and when the approver modifies the request during approval workflow.

For example, when a self-registration request is submitted, the set of validation handlers configured for USER CREATE is run. Next, when the approver modifies the request to populate Organization or other user attributes, these validation handlers are re-run.

Therefore, custom validation handlers must be developed in such a way that the validation logic is re-entrant because they are invoked multiple times in single request flow.

Note: You can add a custom password validation for cases that are not available through Oracle Identity Manager password policies. For example, you can add a password validation to ensure that the password is not a word in a dictionary.

To add a custom password validation, add a custom validation event handler and set the operation to CHANGEACCOUNTPASSWORD for the event handler. Then, you can organize the order in which Oracle Identity Manager triggers the custom event handler.

Consider the following example use case:

There is a requirement of generating the HR Employee Number UDF by appending a random number to the value of the Department Number field. When the create user request or self-registration request is submitted, the HR Employee Number UDF will be auto-generated based on custom logic. If the approver edits the request during approval and modifies the Department Number value, then the HR Employee Number UDF should be re-calculated by using the new value provided for

Department Number. But, if the approver does not change Department Number, then the previous values generated at the time of request submission should be used.

For this, a new validation handler must be developed for generating the HR Employee Number UDF by appending Department Number and a random number. This logic cannot be written in preprocess handler because preprocess handlers are invoked only once in the lifecycle of a request. The logic in this validation handler is as shown:

```
package custom.handlers;

import java.io.Serializable;
import java.util.HashMap;
import java.util.Random;

import oracle.iam.identity.usermgmt.api.UserManagerConstants;
import oracle.iam.identity.utils.Utils;
import oracle.iam.platform.kernel.ValidationException;
import oracle.iam.platform.kernel.ValidationFailedException;
import oracle.iam.platform.kernel.spi.ValidationHandler;
import oracle.iam.platform.kernel.vo.BulkOrchestration;
import oracle.iam.platform.kernel.vo.Orchestration;

public class EmployeeNumberGenerationHandler implements ValidationHandler {

@Override
public void initialize(HashMap<String, String> parameters) {

}

@Override
public void validate(long processId, long eventId, Orchestration orchestration)
throws ValidationException, ValidationFailedException {

HashMap<String, Serializable> contextParams = orchestration.getParameters();
//1. Generate UDF Employee number during request submission as Department Number
and a random number
//2. If request is in approval stage, then control has come here since approver
has modified the request
//2a: Check if approver has modified Department Number. If yes, then re-generate
if( !Utils.isRequestInApprovalStage()) //Utility method to find if request is in
approval stage or not? If it returns true, it means that approver is attempting to
update the request during approval
{

//Step 1:
String dept = contextParams.get(UserManagerConstants.AttributeName.DEPARTMENT_
NUMBER.getId()).toString();
String en = dept+"_"+random();
contextParams.put("SSN", en);

}
else
{
String dept = contextParams.get(UserManagerConstants.AttributeName.DEPARTMENT_
NUMBER.getId()).toString();
//compare with department number with which request was submitted, if modified by
approver; the regenerate SSN
if( Utils.isAttributeModifiedByApprover(orchestration ,
UserManagerConstants.AttributeName.DEPARTMENT_NUMBER.getId()) )

// //Utility method to find if approver has edited the particular attribute or not
```

```
, during approval?
{
String en = dept+"_"+random();
contextParams.put("SSN", en);
}

}

}

private String random() {
Random random = new Random();
String randomStr = "" + random.nextLong();
randomStr = randomStr.replaceAll("-", "");
return randomStr;
}


@Override
public void validate(long processId, long eventId,
BulkOrchestration orchestration) throws ValidationException,
ValidationFailedException {

}

}
```

28.6 Best Practices

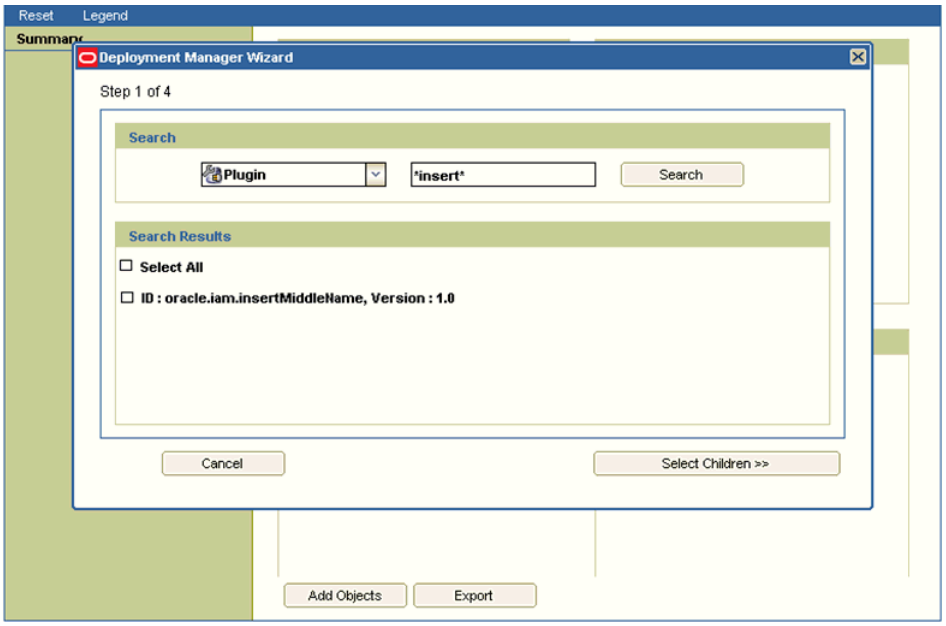
As a best practice, analyze the operation before developing and implementing an event handler. If plug-in is supported for the operation, then use the plug-in for customization rather than developing an event handler. For example, username generation must be implemented by using the available plug-in, and do not attempt writing that as an event handler in the create user orchestration.

For information about points to consider for developing event handlers, see "Development Considerations" on page 28-5.

28.7 Migrating Event Handlers

The Deployment Manager supports migrating plug-ins, and the registered event handlers with the plug-ins, from one deployment of Oracle Identity Manager to another. For example, the event handlers can be implemented in a test environment, and then migrated to the production environment by using the Deployment Manager. Figure 28–2 shows exporting plug-ins via the Deployment Manager:

Figure 28–2 Exporting Plug-ins



See Also: "Migrating Configurations and Customizations" on page 38-1 for information about the Deployment Manager

28.8 Troubleshooting Event Handlers

Table 28–5 lists common problems and causes or solutions to help troubleshoot your event handler if it is not triggered when the operation is executed.

Table 28–5 Troubleshooting Event Handlers

Problem	Cause/Solution
When a user is created through reconciliation, the custom preprocess event handlers are not triggered.	Reconciliation submits postprocess only orchestration where starting stage is postprocess.
When a user is created through reconciliation, the custom postprocess event handler is triggered but the logic inside the execute method is not triggered.	Reconciliation submits bulk orchestrations. Therefore, make sure to implement the bulkExecute method.
The orchestration operation taking too long to complete.	To determine the time spent on each event handler: 1. Connect to http://*OIM_HOST*:*OIM_PORT*/dms/Spy as the WebLogic administrator. 2. In the Metric Tables, click **OIM_EventHandler**. How long each event handler is taking is displayed in the processTime column.

29

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 Self Service 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
- 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.

29.1 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.

29.2 Context Types

Context Manager supports the following context types:

- **SELF:** Operation is initiated through Oracle Identity Self Service.
- **ADMIN:** Operation is initiated through Oracle Identity System 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:

- 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")
- Reconciliation context: The profile from which the reconciliation event has been created can be retrieved by ContextManager.getValue("profileName") method call.
- 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 VII

Customization

This part describes how to customize the user interfaces available with Oracle Identity Manager.

It contains the following chapter:

- Chapter 30, "Customizing the Interface"

30

Customizing the Interface

This chapter explains how to customize various aspects of the user interfaces available in Oracle Identity Manager.

> **Note:** Oracle Identity Manager 11*g* Release 2 (11.1.2.2.0) 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 popup windows and have a different look and feel. These UIs are discussed in the relevant sections in this chapter.

The Identity Self Service user interface (UI) in Oracle Identity Manager is based on Application Development Framework (ADF), which ensures consistent customization. ADF allows UI customization that is safe from patches and upgrades. This means that after you apply patches to Oracle Identity Manager or upgrade Oracle Identity Manager, the UI customizations are preserved.

This chapter describes customizing various aspects of the UI in the following sections:

30.1 Customization Concepts

This section describes the concepts related to UI customization in the following sections:

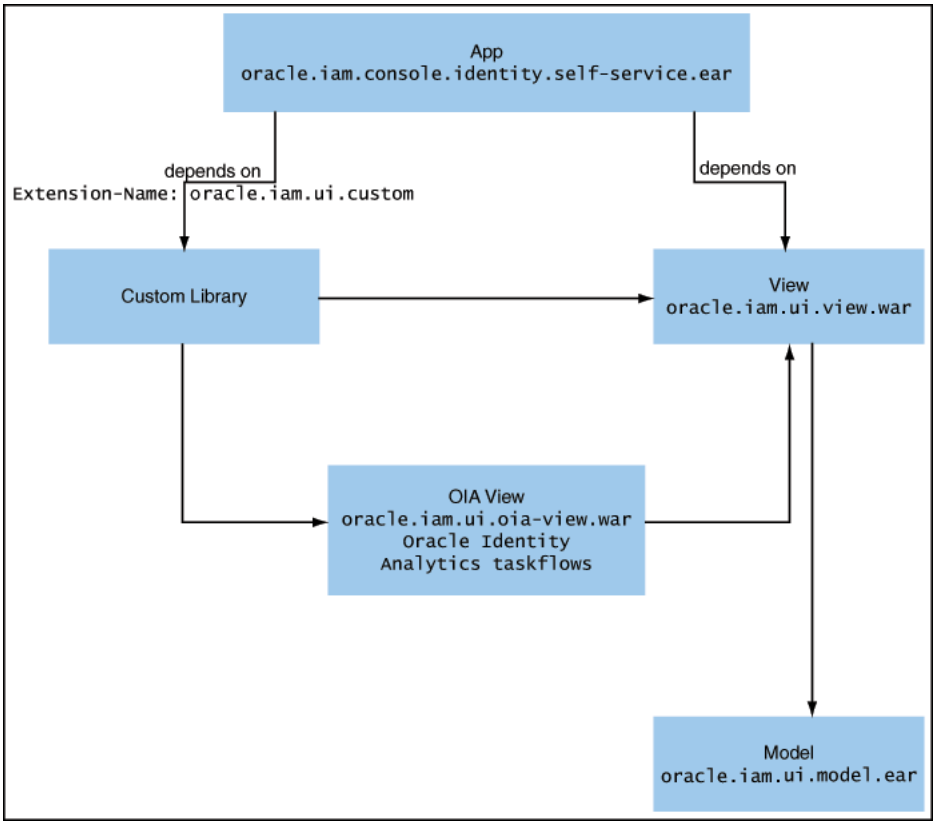
- Deployment of UI Libraries and Applications
- Overview of MDS Customization
- Overview of the Web Composer

30.1.1 Deployment of UI Libraries and Applications

ADF supports inbuilt customization and MDS customization. The customization is achieved by developing new UI artifacts or skins, and packaging them in custom library WAR files. 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.

Figure 30–1 shows the various Oracle Identity Manager UI libraries and their dependency structure.

Figure 30–1 Oracle Identity Manager UI Libraries



The custom library is a placeholder library that you can use to add new taskflows built by using the default libraries. The Self Service EAR declares weblogic.xml dependency on the library-ref name oracle.iam.ui.custom. This library is provided by oracle.iam.ui.custom-dev-starter-pack.war. You can change the WAR file name for the custom library per naming conventions in your organization. However, the deployment name (library extension name in MANIFEST.MF) must be retained as oracle.iam.ui.custom. Otherwise, Self Service ear will not be deployed.

The Oracle Identity Analytics (OIA) deployment name is oracle.iam.ui.oia-view.

30.1.2 Overview of MDS Customization

Using the customization features provided by MDS, you can create applications that fall into the following customization patterns:

- Seeded customization: Seeded customization of an application is the process of taking a generalized application and making modifications to suit the requirements of a particular group, such as a specific industry or site. Seeded customizations exist as part of the deployed application, and endure for the life of a given deployment.
- User customization: User customization allows an end user to change the content of the application at runtime to suit individual preferences (for example, which columns are visible in a table), and have those changes retained the next time the user opens the application.
- Runtime customization: Using the features of Oracle WebCenter, you can create applications that are customizable at runtime. This allows business analysts or administrators to customize the application for their end users by using a Web browser interface.

For detailed information about customizing applications with MDS, see "Customizing Applications with MDS" in the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

30.1.3 Overview of the Web Composer

Oracle Web Composer is an innovative component that enables any application or portal to be customized or personalized after it has been deployed and is in use. Oracle Web Composer runs in all modern Web browsers and enables editing JSF applications and portal pages by selecting information and components from the Business Dictionary or Resource Catalog.

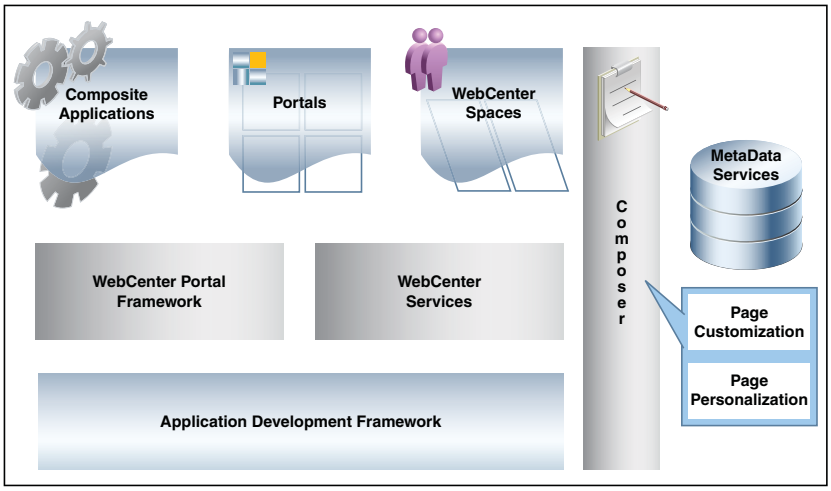
The customization capabilities of the Web Composer include inserting a logo or altering the colors to match those of your company, and complex customizations, adding items to a page, changing the layout of a page, altering a supplied process, and specifically tailoring the delivered application or portal to meet any business requirement. These customizations can be stored in the Oracle Identity Manager database by using Oracle Metadata Services (MDS). For example, to store customizations in the database, the Web Composer creates a copy or sandbox for the pages as they are being edited. The sandbox is a temporary storage area to save a group of runtime page customizations before they are either saved and pushed to other users, or discarded. In this way, customizations can be previewed by others and approved for use before they are visible to all users. See "Managing Sandboxes" on page 30-4 for more information about sandboxes.

Personalization changes your view of the interface or application page. Other users are not affected by the changes you make to a page. The pages can be personalized by individual users to add any combination of components to their page whenever they want them without affecting the view of the page for other users.

Oracle Web Composer is integrated with Oracle WebCenter Framework and WebCenter Services. You can add Oracle Web Composer components to your JSF application pages to enable users to edit those pages at runtime. You can add the components at any time during the development lifecycle, when the requirements of the application demand it. In addition, you can use the page service to enable users to create pages at runtime. Oracle Web Composer has been leveraged inside Oracle

WebCenter Spaces to allow users to customize and personalize personal and group spaces. The Oracle WebCenter Spaces application provides a working example of how users can take an active role in managing and altering their work environment to match their specific requirements. Figure 30–2 shows the Oracle Web Composer architecture.

Figure 30–2 Oracle Web Composer Architecture



30.2 Managing Sandboxes

All customizations and form management are performed in a sandbox. A sandbox allows you to isolate and experiment with customizations without affecting the environment of other users. Any user-interface changes made to a sandbox are visible only in the sandbox. You must create and activate a sandbox to begin using the customization and form management features. After customizations and extending forms are complete, you can publish the sandbox to make the customizations available to other users.

Some of the sandbox operations are:

- **Activate:** You must activate a sandbox to use it. After you activate the sandbox, any changes to UI metadata objects, for example pages and forms, are stored only in the sandbox. There can be only one active sandbox at a time. The information about the active sandbox is stored in the session. Therefore, a sandbox must be activated to continue with customization after every login to Oracle Identity Manager.

  Caution: After creating a sandbox, you must activate it. If you do not activate a sandbox, then you will not be allowed to perform operations on the sandbox later, or might not get the desired result. For example, if you create a sandbox but do not activate it, create a disconnected application instance, run catalog synchronization scheduled job, login to Identity System Administration, and try to checkout the disconnected application instance on the Catalog page, an error will be generated. You can perform this operation without any error, if you activate the sandbox after creating it and after every login to Identity Self Service or Identity System Administration.

- **Deactivate:** Reverse operation to activating a sandbox. If no sandbox is active, then changes to metadata objects are not allowed, and therefore, no UI customization is allowed.
- **Publish:** You must publish a sandbox to merge the changes stored in the sandbox to the mainline and make it available to other users. After you publish the sandbox, the changes are merged to the mainline and cannot be reverted. The sandbox can no longer be activated, deactivated, exported, or deleted.

> **Note:** Before publishing a sandbox, close all tabs and pages of the Identity Self Service or Identity System Administration, and export the sandbox to a ZIP file to have a backup of UI customizations done.
>
> Oracle recommends creating a backup of the MDS before publishing any sandbox. MDS backup can be created by using tools, such as Oracle Enterprise Manager. See "Creating MDS Backup" on page 37-3 for information about creating a backup of the MDS by using Oracle Enterprise Manager.

- **Export:** You can export all changes stored in the sandbox including sandbox metadata to a ZIP file. Then, you can import these changes to the same or another environment.
- **Import:** You can import the sandbox archive (ZIP file) to an environment. Imported sandbox can be used normally as it would have been created in the environment. Beware when importing sandboxes that any available sandbox with the same name will be overwritten by the imported sandbox.

> **Caution:** Any available sandbox with the same name is overwritten by the imported sandbox.

Sandbox management and sandbox operations resemble operations with concurrent versioning system. You can think of a sandbox as a branch in the versioning system. Creating a sandbox is similar to creating a branch. Activating a sandbox is similar to performing changes on top of the branch, and publishing a sandbox is similar to merging the content of the branch to the main branch, sometimes referred to as trunk.

> **Note:** When you create a sandbox, a new branch is created. You can modify MDS content within that branch. Note that you will not be able to view the changes made in other sandboxes that are created later and published to the main branch. Similarly, when you try to merge this sandbox, a concurrent modification exception is generated. It is recommended that you edit the contents of the sandbox manually to remove the conflicting files. However, if manual editing is not possible, then create a new sandbox again and redo the change.

This section describes how to manage sandboxes in the following sections:

- Handling Concurrency Conflicts
- Creating a Sandbox
- Activating and Deactivating a Sandbox
- Viewing and Modifying Sandbox Details

- Exporting and Importing a Sandbox
- Publishing a Sandbox
- Checking Out an Item from Cart
- Deleting a Sandbox
- Reverting Changes to Default Settings

30.2.1 Handling Concurrency Conflicts

Multiple users can customize an application by using sandboxes. While doing so, the following types of concurrency conflicts might take place:

- **Conflicts within a sandbox:** Users overwriting changes created by other users, either directly by changing the same artifact, or indirectly by affecting files that are shared between the artifacts.

  Conflicts within a sandbox can arise when multiple users are customizing an application by using the same sandboxes at the same time, because more than one user may be attempting to customize the same artifact, or performing a customization task that indirectly affects other shared files. An example of a direct conflict is when different users attempt to customize the same page, the same fragment, or the same metadata file in the same layer. An example of an indirect conflict is when two users, each creating their own object, cause a conflict in the metadata file that tracks which new objects have been created by both saving their changes around the same time. Conflicts may also arise when users are editing a shared artifact, such as when a user performs an operation that adds or edits a translatable string. For example, a user edits a field's display label or help text, or a validation rule's error message, while another user performs an operation around the same time that similarly affects translatable strings. Another example of a shared artifact conflict is when two or more users are working in navigator menus which are shared across applications.

- **Conflicts between sandboxes intended for publishing:** Multiple sandboxes with the same customized artifact publishing to the mainline.

  Conflicts between sandboxes can arise when there is more than one sandbox intended for publishing in use. If two sandboxes contain conflicting customization changes to the same artifact and both are being published, then the sandbox that is being published last will not be allowed to be published, and an error describing the conflict will be displayed. To avoid such conflicts, it is recommended to create and use only one sandbox at a time. These types of conflicts can also occur with shared metadata files such as resource bundles that store translatable strings.

When multiple users are working in a single sandbox, these guidelines must be followed:

- Multiple concurrent users in the same sandbox must operate only on different and unrelated objects. For example, if user1 updates object1, then user2 can update object2 but should not update object1. Be aware that if both modifications involve changes to translatable strings, then saving changes to separate objects around the same time may still cause a conflict in the resource bundle that stores the translatable strings.
- Users in the same sandbox can see the changes created by one another. The latest version of each object gets loaded on-demand the first time it is viewed. If there are ADF Business Components customizations, then users must log out and log in again to see those changes reflected in the UI.

When multiple users are working in multiple sandboxes, in addition to all guidelines applicable to multiple users working in a single sandbox, these guidelines must be followed:

- There can be any number of test-only sandboxes operating concurrently. Multiple users can use multiple sandboxes concurrently for testing even if these sandboxes are never published. Sandboxes that are used for testing only, and that are not published, cause no conflicts with each other, but all guidelines for multiple users working in a single sandbox must be followed. However, all modifications are lost when the sandboxes are deleted.
- For sandboxes that will be published, you can have multiple concurrent sandboxes only if they operate on mutually exclusive artifacts. For example, you can have one sandbox that contains a page that is being customized to add a task flow, and another sandbox that contains a different page from a different application.
- If an artifact is updated in both the mainline and in the sandbox (or two different sandboxes), when the sandbox is published, such conflicts are detected and an error is generated.

30.2.1.1 Troubleshooting Concurrency Issues

Table 30–1 lists the issues that you might encounter if there are concurrency conflicts in the sandbox usage and the possible solutions.

Table 30–1 Troubleshooting Concurrency Issues

Example Scenario	Problem	Solution
Working on multiple sandboxes intended for publishing concurrently: Create sandbox S1, create sandbox S2, make changes to S2, publish S2, make changes to S1, and publish S1.	When you try to publish S1, an error is thrown.	Create a new sandbox and redo the changes.

Table 30–1 (Cont.) Troubleshooting Concurrency Issues

Example Scenario	Problem	Solution
Migrating sandboxes out-of-order: In environment 1, create sandbox S1, make changes to S1, export and publish S1. Repeat the same for S2. In environment 2, import S2, publish S2. Then, import S1,and publish S1. Sandboxes S1 and S2 are published in different order.	If there is any overlap between S1 and S2, for example both sandboxes updated the same MDS document), then changes made as part of S2 are overwritten by S1. For example, if AD connector form is created as part of S1 and EBS connector form is created as part of S2, then there will be overlap in CatalogAM.xml.xml and Biz Editor resource bundle file. After the migration, both CatalogAM.xml.xml and Biz Editor resource bundle only contain changes for AD Connector developed as part of S1.	Publish the sandboxes in correct order. You will be able to republish them.
Skipping sandbox during migration: In environment 1, create sandbox S1, make changes to S1, export and publish S1. Repeat the same for S2. In environment 2, import S2, publish S2. Do not migrate S1 at all. S1, which is published in environment 1, is not migrated to environment 2.	If S2 depends on changes made as part of S1, then those changes will be missing in environment 2.	Publish both sandboxes. You will be able to re-publish them.
Migrating sandboxes from multiple source environments: In environment 1, create sandbox S1, make changes to S1, export and publish S1. In environment 2, create sandbox S2, makes changes to S2, export and publish S2. In environment 3, import S1, publish S1. Import S2, and publish S2.	If there is any overlap between S1 and S2, for example both sandboxes updated the same MDS document, then changes made as part of S1 will be lost. For example, if AD connector form is created as part of S1 and EBS connector form is created as part of S2, then there will be overlap in CatalogAM.xml.xml and Biz Editor resource bundle file. After the migration, both CatalogAM.xml.xml and Biz Editor resource bundle only contain changes for EBS Connector developed as part of S2.	Manually merge the sandboxes into one.

30.2.2 Creating a Sandbox

To create a sandbox:

1. Log in to Oracle Identity Self Service or Oracle Identity System Administration.
2. On the upper navigation bar, click **Sandboxes**. The Manage Sandboxes page is displayed. This page has the following sections:
   - **Available Sandboxes:** Displays all the sandboxes that are available for testing the UI customizations, which are not yet published.
   - **Published Sandboxes:** Displays all the published sandboxes.
3. On the toolbar, click **Create Sandbox**. The Create Sandbox dialog box is displayed.
4. In the Sandbox Name field, enter a name for the sandbox. This is a mandatory field.

5. In the Sandbox Description field, enter a description of the sandbox. This is an optional field.
6. Click **Save and Close**. A message is displayed with the sandbox name and creation label.

> **Caution:** Selecting the Activate Sandbox option closes all the open tabs except the Manage Sandboxes tab and activates the created sandbox.

7. Click **OK**. The sandbox is displayed in the Available Sandboxes section of the Manage Sandboxes page.

30.2.3 Activating and Deactivating a Sandbox

To activate a sandbox:

> **Note:** You must close all tabs in the Self Service or System Administration interfaces before activating or deactivating a sandbox.

1. From the table showing the available sandboxes in the Manage Sandboxes page, select the sandbox that you want to activate.
2. On the toolbar, click **Activate Sandbox**.

   The table refreshes and a marker in the Active column is displayed. In addition, the Sandboxes link on the upper navigation bar also displays the active sandbox name in parentheses.

> **Caution:** If any other tabs are open except the Manage Sandboxes tab before activating the sandbox, then Oracle Identity Manager prompts that all the tabs will be closed before the sandbox can be activated.

To deactivate a sandbox:

1. From the table showing the available sandboxes in the Manage Sandboxes page, select the active sandbox that you want to deactivate.
2. On the toolbar, click **Deactivate Sandbox**. The page refreshes and the marker in the Active table disappears.

> **Caution:** If any other tabs are open except the Manage Sandboxes tab before deactivating the sandbox, then Oracle Identity Manager prompts that all the tabs will be closed before the sandbox can be deactivated.

30.2.4 Viewing and Modifying Sandbox Details

To view the details of a sandbox and modify the details:

1. In the table showing the available sandboxes in the Manage Sandboxes page, click the sandbox name link. A dialog box with the sandbox details is displayed.
2. Make the following changes:

- In the Description field, you can enter a description for the sandbox.
- View all the changes to the sandbox in the Change Details table.
- Filter sandbox changes by using the Layer Names, Layer Values, and Change Types lists, and the Filter toolbar icon.
- Delete any changes made in the sandbox by selecting the change in the table, and clicking **Delete Customization**.
- Export the sandbox, if it contains any changes, by clicking **Export Sandbox**.

30.2.5 Exporting and Importing a Sandbox

To export a sandbox from an Oracle Identity Manager deployment to another:

1. From the table showing the available sandboxes in the Manage Sandboxes page, select the sandbox that you want to export.
2. On the toolbar, click **Export Sandbox**.

   If the sandbox contains any changes, then the sandbox content ZIP file starts downloading. You can now take the ZIP file and import it to the same or another environment.

Note: The name of the sandbox ZIP file is not the sandbox name. The sandbox name usually starts with Idem and it is specified in the XML file located inside the ZIP in the /mdssys/sandbox/ directory.

Caution: If the deployment on which the sandbox content ZIP file is being imported already contains a sandbox with the same name, then that sandbox will get overwritten.

To import a sandbox from an Oracle Identity Manager deployment to another:

1. On the toolbar, click **Import Sandbox**. The Import Sandbox dialog box is displayed.
2. In the Sandbox Archive field, enter a path to the sandbox archive that you exported.
3. Click **Import**.
4. Click Refresh. The sandbox, which is imported to the target deployment, is displayed in the Available Sandboxes tab.

30.2.6 Publishing a Sandbox

To publish a sandbox:

Note: Oracle recommends creating a backup of MDS before publishing the sandbox. A backup of MDS can be created by using Oracle Enterprise Manager. See "Creating MDS Backup" on page 37-3 for information about creating a backup of the MDS by using Oracle Enterprise Manager.

1. Before publishing the sandbox, close all the open tabs and pages.

2. From the table showing the available sandboxes in the Manage Sandboxes page, select the sandbox that you want to publish.
3. On the toolbar, click **Publish Sandbox**. A message is displayed asking for confirmation.
4. Click **Yes** to confirm. The sandbox is published and the customizations it contained are merged with the main line.
5. You can click the **Published Sandboxes** tab to view a list of the published sandboxes.

30.2.7 Checking Out an Item from Cart

To check out an item from cart:

1. Log in to Oracle Identity System Administration.
2. Create a sandbox. See "Creating a Sandbox" on page 30-8 for information about creating a sandbox.
3. Activate the Sandbox that you created. See "Activating and Deactivating a Sandbox" on page 30-9 for information about activating sandbox.

> **Caution:** If you do not activate the sandbox, you will not be allowed to save the disconnected application instance date.

4. Create a disconnected application instance "disc1". By default it will be published to top. See "Creating a Disconnected Application Instance" section in the *Oracle Fusion Middleware Developer's Guide for Oracle Identity Manager* for information about creating a disconnected application instance.
5. Publish the Sandbox. See "Publishing a Sandbox" on page 30-10 for information about publishing a sandbox.
6. Run Catalog Synchronization Job.
7. Log in to Oracle Identity Self Service.
8. Click on **Catalog** and search for "disc1".
9. Click **Add** to add the item to the cart.
10. Click **Checkout from Cart**.

30.2.8 Deleting a Sandbox

To delete a sandbox:

1. From the table showing the available sandboxes in the Manage Sandboxes page, select the sandbox that you want to delete.
2. On the toolbar, click **Delete Sandbox**. A message is displayed asking for confirmation.
3. Click **Yes** to confirm. The sandbox is deleted and is no longer displayed in the Manage Sandboxes page.

Note: Deleting a sandbox does not delete the forms created while the sandbox is active. Deleting forms is not supported in this release of Oracle Identity Manager.

30.2.9 Reverting Changes to Default Settings

You must perform all customizations within a sandbox. Until the sandbox is published, the changes are visible to you only and can be easily reverted by deactivating or deleting the sandbox. After the sandbox is published, the changes done cannot be reverted.

You can remove specific changes from the sandbox in any one of the following ways:

- Export the sandbox and modify it manually.
- Navigate to the Manage Sandboxes page, open the details of your sandbox, select a change, and delete it by clicking **Delete Customization**.

When an MDS sandbox is published, the documents are committed to the MAIN line. Your application starts using these documents immediately, and the application user views the effect of publishing the sandbox. Sometimes, you might inadvertently publish an incomplete or a wrong sandbox. In such instances, it is possible to recover your MAIN line to the state just before you created the wrong sandbox.

For example, if you create a sandbox called Show Admin Feature at time T1, and in that you customized a JSFF fragment published at time T2. You realize later that the sandbox you published is wrong, and you want to recover your state to time T1. To do so:

1. Login to Oracle Enterprise Manager.
2. On the left pane, expand **Application Deployments**, and open the oracle.iam.ui.console.self-service application.
3. On the top of the right pane, click **Application Deployment**, and select **MDS Configuration**.
4. At the lower part of the right pane, click the **Runtime MBean Browser** link. The screen refreshes.
5. Click the **Operations** tab.
6. Scroll down. Click and open the **list Metadata Labels** MBean operation, and then click **Invoke**. All the sandboxes recreation labels and post publish labels are displayed. Select the sandbox pre create to which you want to restore and copy it to clipboard.
7. Click **Return** to go back to the operations.
8. Select the promote Metadata Label MBean operation and invoke it by provisioning the value that you copied in step 6.
9. Restart oim_server1.

   Your MDS main line is reverted to the state that was at the time of creation of the sandbox.

Note: You can also restore to the last successful sandbox that was published by restoring to the post label of that sandbox.

30.3 Skin Customization in Oracle Identity Manager

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 "skyros":

```
<?xml version="1.0" encoding="windows-1252"?>
<trinidad-config xmlns="http://myfaces.apache.org/trinidad/config">
  <skin-family>#{iamSkin.skinFamily}</skin-family>
  <skin-version>#{iamSkin.skinVersion}</skin-version>
  <accessibility-mode>#{accessibilityBean.accessibilityMode}</accessibility-mode>
<accessibility-profile>#{accessibilityBean.accessibilityProfile}</accessibility-pr
ofile>
</trinidad-config>
```

There are two console-specific trinidad-config.xml files. The files are in the following directory paths:

- For Identity Self Service: /iam/iam-product/consoles/Identity/MainUI/publicum/WEB-INF/trinidad-config.xml
- For Identity System Administration: /iam/iam-product/consoles/Identity/AdvancedUI/public_html/WEB-INF/trinidad-config.xml

30.3.1 Configuring a New Skin

To create a new skin:

1. In each WEB-INF directory, create the trinidad-skins.xml file, as shown:

```
<?xml version="1.0" encoding='utf-8'?>
<skins xmlns="http://myfaces.apache.org/trinidad/skin">
    <skin>
        <id>myskin.desktop</id>
        <family>myskin</family>
        <extends>#{iamBean.skyros-v1.desktop}</extends>
        <render-kit-id>org.apache.myfaces.trinidad.desktop</render-kit-id>
        <style-sheet-name>skins/myskin/myskin.css</style-sheet-name>
    </skin>
</skins>
```

2. Configure the following system properties:

- Skin Family for OIM UI: The ADF skin family for Oracle Identity Manager UI that the application uses at runtime.
- Skin Version for OIM UI: The skin version, if any, for the skin family being used for Oracle Identity Manager UI.

  See Also: "System Properties in Oracle Identity Manager" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager* for information about the Skin Family for OIM UI and Skin Version for OIM UI system properties

Change the values of these properties to your custom skin. Before setting the values of these properties, the custom ADF skin must be developed and then deployed into Oracle Identity manager.

Oracle ADF faces provide simplified default skins, such as skyros-v1.desktop, which are designed to be extended by custom skins. The ADF skin that you create must extend either one of the ADF skins that Oracle ADF provides or from an existing ADF skin that you created. Skins are reusable components and must be created in their own JDeveloper project or workspace. You can create an ADF skin file in the ADF Skin Editor that defines how ADF faces components render at runtime.

See Also:

- "Inheritance Relationship of the ADF Skins Provided by Oracle ADF" in the *Oracle Fusion Middleware Skin Editor User's Guide for Oracle Application Development Framework* for information about the inheritance relationship between the ADF skins that Oracle ADF provides
- "ADF Skins Provided by Oracle ADF" in the *Oracle Fusion Middleware Skin Editor User's Guide for Oracle Application Development Framework* for information about the levels of customization in the ADF skins provided by Oracle ADF and for a recommendation about the ADF skin to extend
- *Oracle Fusion Middleware Skin Editor User's Guide for Oracle Application Development Framework* and *Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework* for detailed information about ADF custom skin development and deployment

3. If required, adjust the height of the top branding bar to fit the new logo image in the following:

   ```
   oracle.iam.console.identity.self-service.ear/oracle.iam.console.identity.self-service.war/oracle/iam/ui/main
   ```

   For the login page (signin.jspx):

   ```
   <f:attribute name="globalBrandingSize" value="64"/>
   ```

   For the page displaying challenge questions (firstlogin.jspx):

   ```
   <f:attribute name="globalHeaderSize" value="64"/>
   ```

   For the pages in Identity Self Service (identity.jspx):

   ```
   <f:attribute name="globalHeaderSize" value="64"/>
   ```

4. In the oracle.iam.console.identity.self-service.ear/oracle.iam.console.identity.self-service.war/WEB-INF/web.xml file, include the following settings:

```
<context-param>
  <description>No obfuscation of CSS.</description>
 <param-name>org.apache.myfaces.trinidad.DISABLE_CONTENT_
COMPRESSION</param-name>
  <param-value>true</param-value>
</context-param>
<context-param>
  <description>If this parameter is true, there will be an automatic check of
the modification date of your JSPs, and saved state will be discarded when
JSP's change. It will also automatically check if your skinning css files have
changed without you having to restart the server. This makes development
easier, but adds overhead. For this reason this parameter should be set to
false when your application is deployed.</description>
  <param-name>org.apache.myfaces.trinidad.CHECK_FILE_MODIFICATION</param-name>
  <param-value>true</param-value>
</context-param>
```

Note: Oracle recommends not to customize the top section and the left navigation pane of the Identity Self Service. The customization in these sections might not work properly, and any existing customization might be lost.

30.3.2 Configuring Skin for Legacy Advanced Console

To keep the defaults coming from the 'skyros' skin and override certain style sheet elements:

1. Create the skin in the trinidad-skins.xml file. Declare the skin in a new file oim.ear/iam-consoles-faces.war/WEB-INF/trinidad-skins.xml, as shown:

```
<?xml version="1.0" encoding='utf-8'?>
<skins xmlns="http://myfaces.apache.org/trinidad/skin">
    <skin>
        <id>myskin.desktop</id>
        <family>myskin</family>
        <extends>skyros.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</bu
ndle-name>
    </skin>
</skins>
```

2. Register the new "myskin" in both the /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 new skin CSS file oim.ear/iam-consoles-faces.war/skins/myskin/myskin.css.
4. In myskin.css, 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.

Note: To change the branding information as an alternative to the method in "Changing Branding and Logo" on page 30-16, create a custom skin and use the appropriate style classes given in the URL in this section.

30.3.3 Changing Branding and Logo

Customizing or changing UI artifacts, such as logo, buttons, and menu items, can be done at runtime by using Oracle WebCenter Composer.

Note: The procedure documented in this section is for changing the branding and logo by customizing Oracle Identity Self Service. If you want to customize UI artifacts of the window that opens from the Oracle Identity System Administration, for example, the window that opens when you click System Configuration under System Management, then see "Branding Customization" at the following URL:

http://docs.oracle.com/cd/E21764_01/doc.1111/e14309/uicust.htm#BABFCFID

To change the logo image:

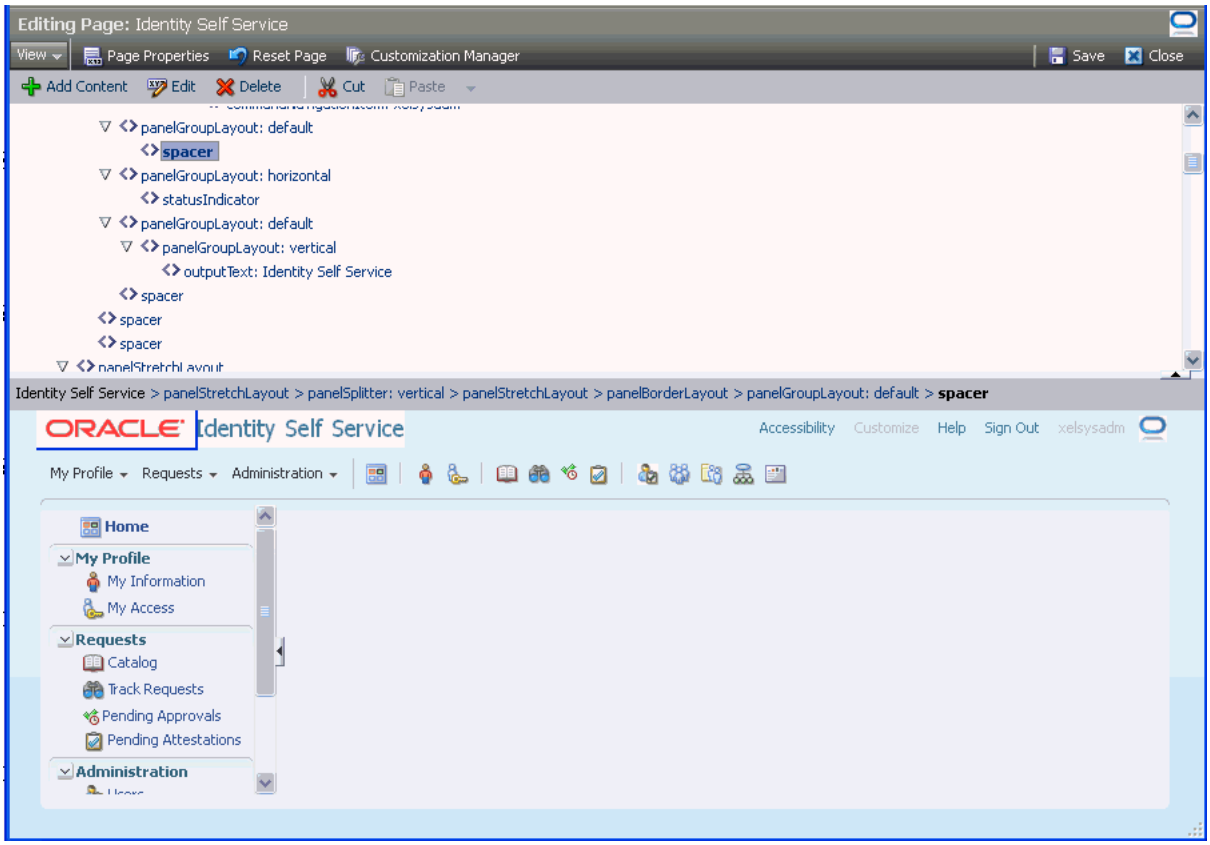
1. Log in to Oracle Identity Self Service as the system administrator.
2. Create and activate a sandbox.

Note: Creating and activating a sandbox is mandatory for customizing the UI by using the Web Composer. Without an active sandbox, Oracle Identity Manager does not allow to open any page in customization mode.

---

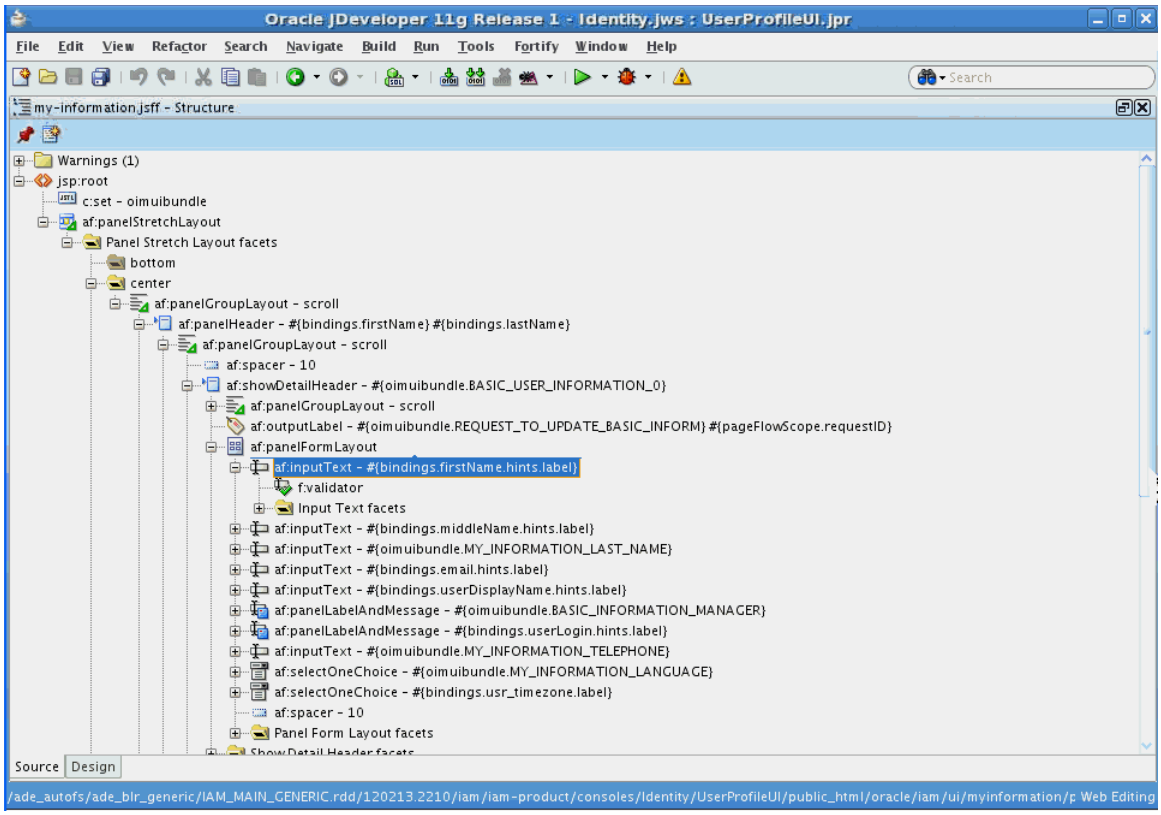
3. Click **Customize**. The Oracle WebCenter Composer opens. The top of the page shows 'Editing Page'.
4. From the View list, select **Source**. The object tree is displayed in the top pane. The object tree shows all the ADF components of the page.
5. Click the logo. The logo object is selected in the object tree, as shown in Figure 30–3:

Figure 30–3 The Object Library in WebCenter Composer



The ADF components displayed in the object tree is similar to the structure pane in JDeveloper, as shown in Figure 30–4:

Figure 30–4 The Structure Pane



> **Note:** For a complete list of UI components, see *Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework*.

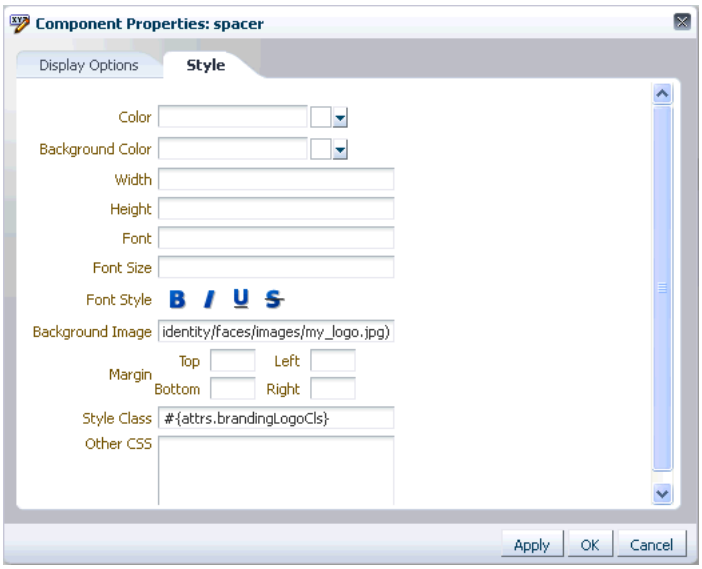
6. Click **Edit**. The Component Properties dialog box is displayed.
7. Click the **Style** tab.
8. In the Background Image field, enter the path to the logo image that you want to set, for example, url(/identity/faces/images/mylogo.jpg).

   Tip:

   - Customizing the default EAR and WAR files, such as Self Service EAR, System Administration EAR, and xlWebApp.war, is not supported.
   - The Oracle logo is 119x25 pixels. Therefore, you can use a custom logo of the same dimensions.

   Figure 30–5 shows the Style tab of the Component Properties dialog box:

Figure 30–5 The Component Properties Dialog Box



Tip: If you want to specify a font for any ADF component by using the Style tab of the Component Properties dialog box, then ensure that your target browsers and platforms support that specific font name. To look at the supported list for Mozilla Firefox, select **Tools**, **Options**, **Content**, **Fonts and Colors**. For Microsoft Internet Explorer, select **Tools**, **Internet Options**, **General**, **Fonts**.

9. To change the Identity Self Service global banner, click the **Identity Self Service** text, and repeat steps 5 to 7.

   Tip: To change the banner in the Oracle Identity Manager login page, you must open the login page in the customization mode. However, the Customize link is not available in the login page. Therefore, to open the login page in customization mode:

   1. Login to Oracle Identity Self Service as an administrator with privileges to customize the UI.
   2. In an active sandbox, click the **Customize** link. The Oracle Identity Self Service is in customization mode.
   3. Perform the steps described in "Customizing the User Registration and Other Unauthenticated Pages" on page 30-30.

10. Click **Save** to save the logo and global banner changes.
11. Click **Close** to close WebCenter Composer.
12. Publish the sandbox.

Note: Runtime UI customization changes the banner text. However, it does not allow you change the browser title, for instance, the title that appears in the browser window, HTML>HEAD>TITLE text or the browser tab title if using tabs.

If you want to change the browser text title, then you have to manually edit the value in the resource bundle file OIMUIBundle_*.properties, and redeploy as follows:

oracle.iam.ui.view.war/WEB-INF/lib/adflibCommonUI.jar/oracle/iam/ui/OIMUIBundle_en.properties:

```
IDENTITY_SELF_SERVICE_TITLE=Identity Self Service
```

See "Internationalization for Resource Strings" on page 30-23 for information about creating and using the custom resource bundles.

30.4 Customizing Pages at Runtime

Customizing Oracle Identity Manager can be broadly categorized into customizing the UI and extending the object definitions of the user, role, catalog, and provisioning target resource entities.

Table 30–2 lists the artifacts that can be customized for each entity.

Table 30–2 Entity Artifacts for Customization

Available	Artifacts
User	Create Page
	Modify Page
	User Attribute Details
	Advanced Search Interface
	My Information
	Self Registration
Role	Create Page
	Modify Page
	Advanced Search Interface, which includes:
	- Query Criteria
	- Results Table columns
Catalog	Catalog Search Page that includes:
	- Results Table columns
	- Catalog Item Details
Provisioning target resource	Provisioning Target Resource Create Form
	Provisioning Target Resource Modify Form
	Provisioning Target Resource Bulk Form

See Also: "Managing Forms" and "Configuring Custom Attributes" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager* for information about creating and managing forms by using the Form Designer

This section contains the following topics:

- Using Expression Language in UI Customization
- Showing or Hiding UI Components Conditionally
- Showing Request Profiles Conditionally
- Validating Input Data Using ADF Validators
- Marking Input Attribute as Required

The Web Composer enables you to customize the UI at runtime. This section describes the following UI customizations:

- Adding a Link or Button
- Hiding and Deleting an ADF Component
- Showing and Hiding Attributes
- Customizing the User Registration and Other Unauthenticated Pages
- Customizing Certification Pages

30.4.1 Using Expression Language in UI Customization

Expression Language (EL) allows you to access application data stored in JavaBeans components. For an introduction to EL and EL expression syntax, refer to the following URL:

http://developers.sun.com/docs/jscreator/help/2update1/jsp-jsfel/jsf_expression_language_intro.html

30.4.1.1 Available EL Expressions in the User Context

The OIMContext bean is defined as an ADF session scope bean in adfc-config.xml in your project. Table 30–3 lists the available EL expressions in the Oracle Identity Manager user context.

Table 30–3 EL Expressions in User Context

EL	Description
`#{oimcontext.currentUser['ATTRIBUTE_NAME']}`	Access value of the *ATTRIBUTE_NAME* attribute of the logged-in user. For the list of default attributes defined for the user entity, see "Attribute Definitions" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager*.
`#{oimcontext.currentUser['UDF_NAME']}`	Access value of the *UDF_NAME* attribute of the logged-in user. UDF attributes can be defined by using the Form Designer.
`#{oimcontext.currentUser.roles}`	Access the *ROLE_NAME* and RoleEntity mapping that contains the roles assigned to the logged-in user. RoleEntity is Java Bean having name, description, key, and displayName properties.
`#{oimcontext.currentUser.roles['SYSTEM ADMINISTRATORS'] != null}`	Boolean EL that evaluates to true if the logged-in user has the System Administrator admin role. Similarly, you can modify the EL to check for any other role.

Table 30–3 (Cont.) EL Expressions in User Context

EL	Description
`#{oimcontext.currentUser.adminRoleMap['OrclOIMSystemAdministrator'] != null}`	Boolean EL that evaluates to true if the logged-in user has the OrclOIMSystemAdministrator admin role. Similarly, you can modify the EL to check for any other admin role.

You can use EL expression to retrieve all available user attribute values from the oimcontext bean, as shown in the following examples:

- To get the user key of the currently logged-in user:

  ```
  #{oimcontext.currentUser.usr_key}
  ```

  OR:

  ```
  #{oimcontext.currentUser['usr_key']}
  ```

- To get the list of role names of the currently logged-in user:

  ```
  #{oimcontext.currentUser.roles}
  ```

- To get the list of admin role names of the currently logged-in user:

  ```
  #{oimcontext.currentUser.adminRoles}
  ```

As an example, if you want to display a message with the user login name when a user logs in to Oracle Identity Self Service, then you can use EL expression to retrieve the login name of the currently logged-in user, and display it on the page. The expression to retrieve the user login name is the following:

```
#{oimcontext.currentUser['User Login']}
```

30.4.1.2 Available EL Expressions in the RequestFormContext

RequestFormContext is a bean available in the pageFlowScope of entity form details task flow. The entity forms include user form, application instance form, role form, and entitlement form. RequestFormContext provides various context information. Using this context information, you can customize the forms based on specific business requirements.

Table 30–4 lists the EL expressions involving RequestFormContext.

Table 30–4 EL Expressions in RequestFormContext

EL	Description
`#{pageFlowScope.requestFormContext}`	Access current instance of RequestFormContext.
`#{pageFlowScope.requestFormContext.operation}`	Access operation type that is being performed on the entity. The possible values are CREATE, MODIFY, ENABLE, DISABLE, and REMOVE.
`#{pageFlowScope.requestFormContext.operation == 'MODIFY'}`	Boolean EL that evaluates to true if current operation being performed on the entity is MODIFY.
`#{pageFlowScope.requestFormContext.actionType}`	Access action that is being performed by the user when the entity form is displayed. The possible values are APPROVAL, FULFILL, REQUEST, VIEW, and SUMMARY.

Table 30–4 (Cont.) EL Expressions in RequestFormContext

EL	Description
`#{pageFlowScope.requestFormContext.actionType == 'REQUEST'}`	Boolean EL that evaluates to true if the action that is being performed by the user when the entity form is displayed is REQUEST, for example, requesting role or application instance.
`#{pageFlowScope.requestFormContext.bulk}`	Boolean EL that evaluates to true if the operation being performed is a bulk operation, for example, requesting multiple application instances at a time.
`#{pageFlowScope.requestFormContext.beneficiaryIds}`	Access the list of beneficiary or target user IDs. For example, if you are requesting an application instance for user John Doe, then the list contains the user ID of John Doe. **Note:** Oracle recommends accessing the list and performing operations on it by using Java code.
`#{pageFlowScope.requestFormContext.cartItemIds}`	Access the list of cart item IDs. For example, if you are requesting an application instance for a user, then the list contains the application instance ID that is being requested. **Note:** Oracle recommended accessing the list and performing operations on it by using Java code.
`#{pageFlowScope.requestFormContext.requestEntityType}`	Get entity type being requested. The possible values are ROLE, ENTITLEMENT, APP_INSTANCE, and USER.
`#{pageFlowScope.requestFormContext.requestEntityType == 'APP_INSTANCE'}`	Boolean EL that evaluates to true if the entity type being requested is APP_INSTANCE.
`#{pageFlowScope.requestFormContext.requestEntitySubType}`	Access subtype of entity being requested. For example, when requesting APP_INSTANCE, requestEntitySubType is the application instance key.
`#{pageFlowScope.requestFormContext.instanceKey}`	Access the key of the instance being modified.

30.4.1.3 Internationalization for Resource Strings

In Oracle Identity Manager, you can create custom resource bundles and reference them in the UI. If you want to modify some of the predefined UI elements such as labels, headers, and so on, or the values displayed on a certain page (for example, values displayed in the Status field of the Request Summary page), then perform the procedure described in this section.

WARNING: Any changes made to the default WAR or EAR files should be redone after any Bundle Patch (BP) is applied as the Bundle Patch will overwrite them.

To create custom resource bundles:

1. Open the custom project WAR file, which is oracle.iam.ui.custom-dev-starter-pack.war.
2. Create a new CustomResourceBundle.properties file.
3. In the new file, enter the key value pairs, for example:

```
CUSTOMRB_BANNER_TEXT=My Identity and Access
```

4. Create all localized files, for example CustomResourceBundle_it.properties and CustomResourceBundle_es.properties, in the same directory.
5. Repackage the custom WAR, and update the custom WAR deployment in the server.

To use the resource bundles:

1. In Oracle Identity Self Service, create a sandbox, and click **Customize**.
2. On the Component Properties dialog box, open the Expression Editor for the specific property, and specify the expression, for example:

```
#{adfBundle['oracle.iam.ui.custom.CustomResourceBundle'].CUSTOMRB_BANNER_TEXT}
```

3. Click **Test** to test the expression. Click **OK**, then click **Apply**.
4. Click **OK** to close the Component Properties dialog box.
5. Export the sandbox, and then publish the sandbox.

Note: Exporting the sandbox is optional, but it is a recommended step.

30.4.2 Showing or Hiding UI Components Conditionally

To conditionally show or hide UI components, use the rendered property of the component and assign EL expression to it that evaluates to Boolean. If the EL expression evaluates to true, then the component is shown. Consider the following examples:

Note: The rendered property of the component corresponds to the Show Component option in Oracle Web Composer.

- To show a UI component if the logged-in user has the System Administrators admin role:

```
#{oimcontext.currentUser.roles['SYSTEM ADMINISTRATORS'] != null}
```

  Similarly, the EL expression can be modified to check if the logged-in user has any other role.

- To show a UI component if signed-in user has the System Administrator admin role:

```
#{oimcontext.currentUser.adminRoles['OrclOIMSystemAdministrator'] != null}
```

  Similarly, the EL expression can be modified to check if the logged-in user has any other admin role.

- To show a UI component if the usr_key attribute of the logged-in user is 1:

```
#{oimcontext.currentUser['usr_key'] == 1}
```

- To show a UI component if the logged-in user's last name is Doe:

```
#{oimcontext.currentUser['Last Name'] == 'Doe'}
```

- To show a UI component if the logged-in user belongs to the Xellerate Users organization:

```
#{oimcontext.currentUser['Organization Name'] == 'Xellerate Users'}
```

- To show a UI component if the user's UDF attribute called UDF_NAME equals to UDF_VALUE:

```
#{oimcontext.currentUser['UDF_NAME'] == 'UDF_VALUE'}
```

Note: "Showing Components Conditionally" on page 30-46 describes showing components based on certain conditions by implementing custom Managed Bean.

30.4.3 Showing Request Profiles Conditionally

To show a catalog request profile conditionally:

1. Login to Oracle Identity Self Service.
2. Activate a sandbox.
3. Navigate to the Catalog page.
4. Click **Customize**. From the View list, select **Source**.
5. Using the source tree, navigate to the iterator component within Request Profiles. The iterator component has panelGroupLayout subcomponent, which represents single request profile.
6. Select **panelGroupLayout:horizontal**, which is a subcomponent of panelGroupLayout:vertical inside the iterator component, and click **Edit** in the Web Composer.
7. Assign a Boolean EL expression to the rendered property. This is the Show Component in Web Composer.

   For example, if you want to display a resource profile called Profile to users of the Suppliers organization only, and display any other profile to other users, then use the following expression:

   ```
   #{(row.profileName == 'Profile' && oimcontext.currentUser['Organization Name']
   == 'Suppliers') || row.profileName != 'Profile'}
   ```

   The EL expression is evaluated for every profile which is available. Similarly, you can modify/extend the EL expression to conditionally display any other profile.
8. Publish the sandbox to globalize the change.

30.4.4 Validating Input Data Using ADF Validators

To validate input component data using predefined ADF validators, you must modify the JSFF page fragment and include one of the ADF validators as a child element of input component. Table 30–5 lists the ADF validators:

Table 30–5 ADF Validators

Validator	Description
<af:validateByteLength>	Validates the byte length of strings when encoded
<af:validateDateRestriction>	Validates that the date entered is within a given restriction
<af:validateDateTimeRange>	Validates that the date entered is within a given range

Table 30–5 (Cont.) ADF Validators

Validator	Description
<af:validateDoubleRange>	Validates that the date entered is within a given range
<af:validateLength>	Validates that the value entered is within a given length
<af:validateLongRange>	Validates that the value entered is within a given range
<af:validateRegExp>	Validates an expression by using Java regular expression syntax

For example, to validate that the only allowed characters for the User Login attribute are alphanumeric ASCII characters, you can include the following RegExp validator as a child element of the User Login input component:

```
<af:validateRegExp pattern="[a-zA-Z0-9]*"/>
```

ADF validators cannot be added directly by using the Web Composer. Instead, you can add another component as a child component of the User Login component, for example, another input text. After that you can export the sandbox containing this change. Finally, update the JSFF page fragment for the form in the exported sandbox, and then import the sandbox.

Note: "Implementing Custom Field Validation" on page 30-50 describes implementing the custom field validator by using custom Managed Bean.

30.4.5 Marking Input Attribute as Required

To conditionally make an input field required, you can use the required property of the component, and assign it a Boolean EL expression. If the EL expression evaluates to true, then the component is marked as required, and the required validation is triggered.

For example EL expressions, see "Showing or Hiding UI Components Conditionally" on page 30-24.

For more information about making field conditionally mandatory based on the value of another field, see "Setting a Conditional Mandatory Field" on page 30-49.

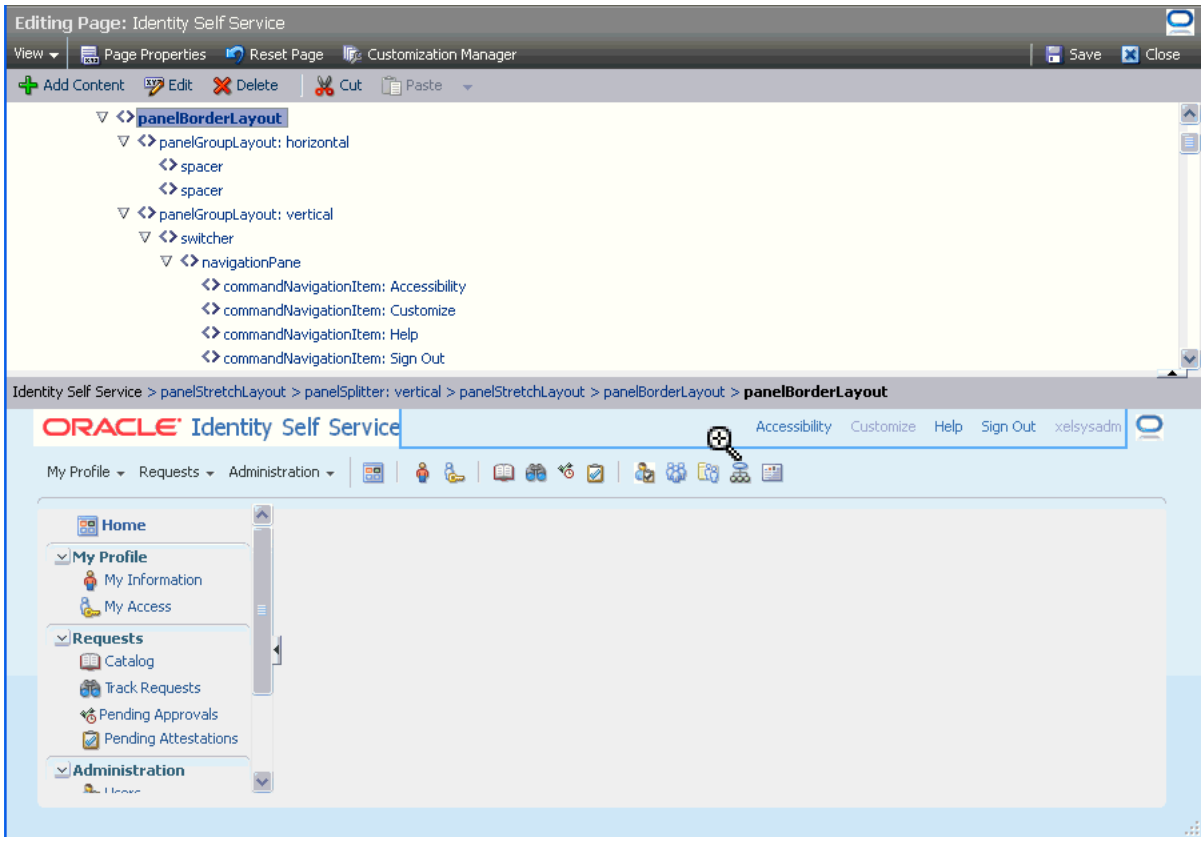
30.4.6 Adding a Link or Button

To add a link to Oracle Identity Self Service:

1. From any page in Oracle Identity Self Service, open WebCenter Composer.
2. Select the top panel on which you want to include the link. The ADF component is selected in the object tree.

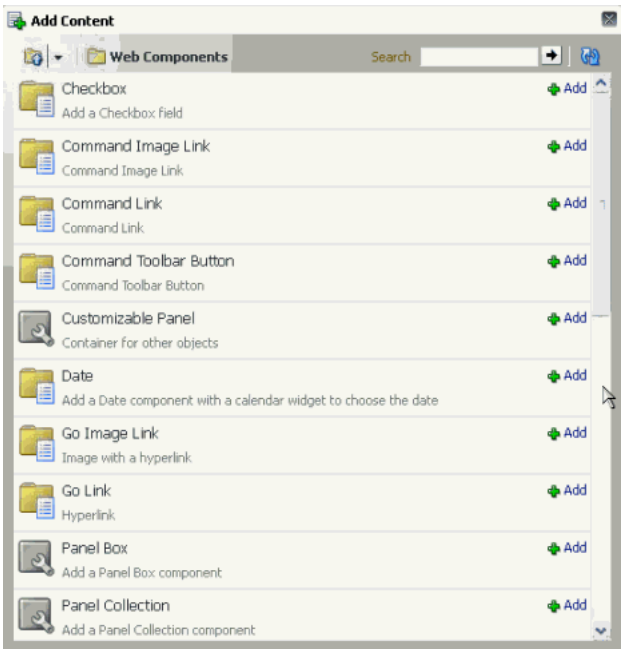
   Figure 30–6 shows the selected top panel and corresponding ADF component in the object tree.

Figure 30–6 Panel Selection for Adding Link



3. Click **Add Content** and open Web Components. The Web Components component in the Add Content dialog box is shown in Figure 30–7:

Figure 30–7 The Add Content Dialog Box



4. Search for the link component that you want to add, and click **Add** in the same row. The link is added to the selected panel.

   Note: For a complete list of UI components, see *Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework*.

5. Click **Save** to save your changes, and close WebCenter Composer.

   Note: For more details, see the following sections:

   - "Launching Taskflows" on page 30-58
   - "Creating an External Link" on page 30-60

Oracle Identity Manager allows you to add your own UI or taskflows, such as goLink, commandLink, commandButton, or launch a taskflow.

Perform the following steps to add your custom UI or taskflow:

1. Write a managed bean and register using adfc-config.xml in oracle.iam.ui.custom-dev-starter-pack.war.
2. Add a new commandLink or commandButton on the page where you want to display the link or button by using Web Composer.
3. Set the actionListener property of the link or button component that you added to point to the actionListener method.
4. Raise the contextual event using the managed bean, which will be handled by Oracle Identity Manager.

Shell launches the taskflow.

30.4.7 Hiding and Deleting an ADF Component

Hiding an ADF component results in the UI artifact being hidden from the user. To hide an ADF component:

1. In Oracle Identity Self Service, go to the page on which you want to hide a component.
2. Click **Customize** to open WebCenter Composer.
3. From the View menu, select **Source**. The object tree is displayed.
4. Click the component on the page that you want to hide. The corresponding ADF component in the object tree is selected.
5. Right-click the selected ADF component in the object tree, and select **Hide**.

To delete an ADF component:

1. From the Oracle Identity Self Service page on which you want to delete any UI component, open Web Composer.
2. From the View menu, select **Source**. The object tree is displayed.
3. Click the component on the page that you want to delete. The corresponding ADF component in the object tree is selected.
4. Right-click the selected ADF component in the object tree, and select **Delete**.

5. In the Delete Component Confirmation box, click **Delete**.

> **Note:** Changing searchable property of default attributes is not supported.

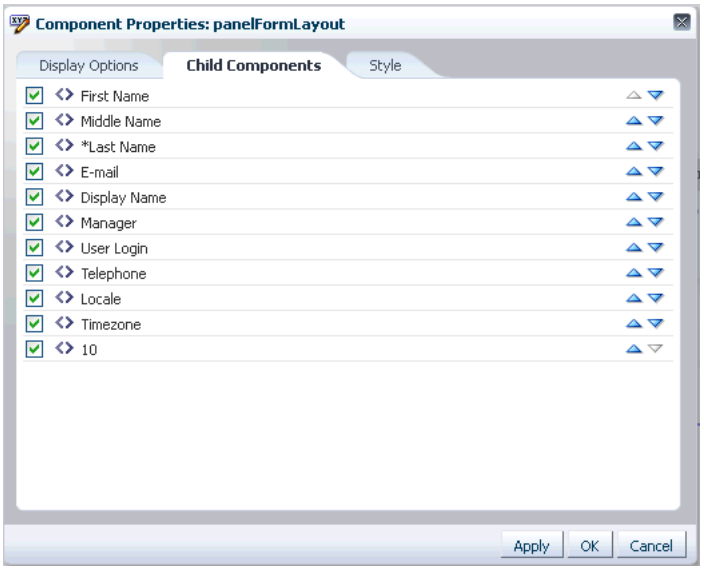
30.4.8 Showing and Hiding Attributes

To show or hide attributes in a page:

1. Go to the page on which you want to show or hide the attribute. For example, navigate to the My Information page in the Oracle Identity Self Service if you want to show or hide the Telephone field.
2. Click **Customize** to open Web Composer.
3. From the View menu, select **Source**. The object tree is displayed.
4. Click the region or section that contains the attribute you want to hide, or you want the attribute to be shown.

   The Confirm Task Flow Edit message box is displayed.
5. Click **Edit**. The ADF component for the selected region is selected in the object tree.
6. On the toolbar, click **Edit**. The Component Properties dialog box is displayed.
7. Click the **Child Components** tab. All the UI components of the selected region are displayed. Figure 30–8 shows a sample Child Components tab in the Component Properties dialog box.

Figure 30–8 The Child Components Tab



8. Select or deselect the checkbox corresponding to the attributes to show or hide the attributes respectively.

Note: If you do not see an attribute listed here, then you must add the attribute into the form. See "Adding a Custom Attribute" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager* for details.

9. Click **Apply**. The selected attributes are hidden or shown based on your selection.
10. Click **OK**, and then click **Save** on the toolbar.

30.4.9 Customizing the User Registration and Other Unauthenticated Pages

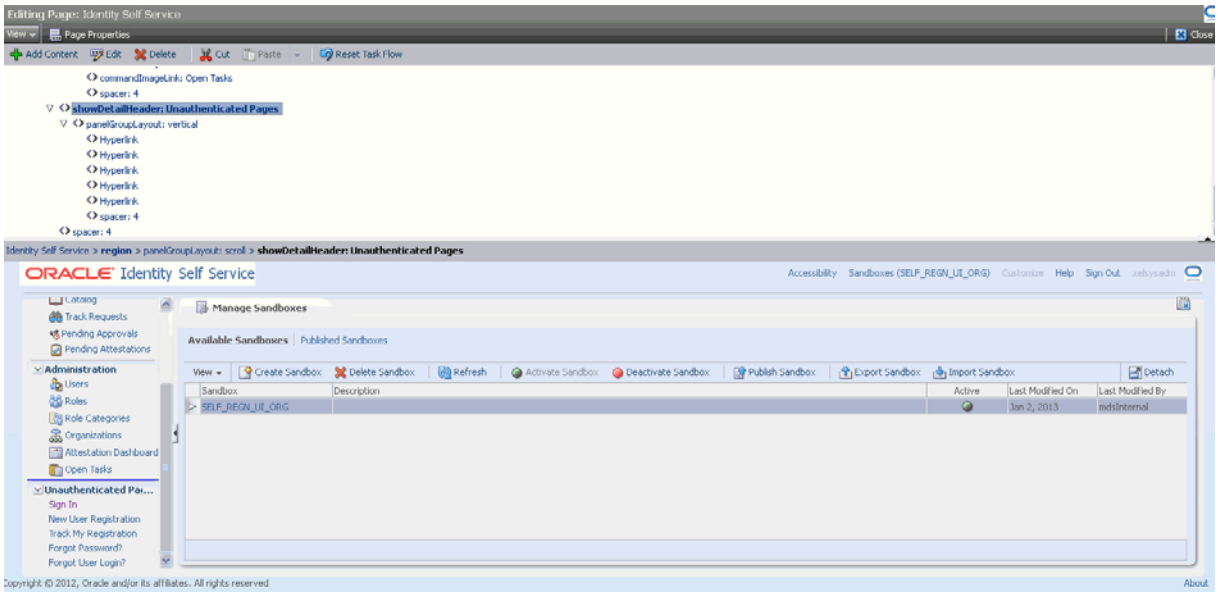
To customize the User Registration and other unauthenticated pages of the Identity Self Service:

1. Login to Oracle Identity Self Service as the system administrator.
2. Create and activate a sandbox, and click **Customize**.
3. Select **View**, **Source**. Click the last visible link in the left navigation pane, for example, Open Tasks.

Note: Do not open any tab. You need to customize the left navigation pane itself.

4. Confirm to edit task flow. Using the source tree pane, scroll down to find the grayed-out item 'Unauthenticated Pages'.
5. Right-click **Unauthenticated Pages**, and select **Show Component**. The links to go to unauthenticated pages are displayed on the left navigation pane, as shown in Figure 30–9.

Figure 30–9 Unauthenticated Page Links



6. Select **View**, **Design**. Click the **New User Registration** link in the left navigation pane. You are redirected to the User Registration screen.

7. After filling required fields, select **View**, **Source**. Click **Add Content** and use the Data Component - User Registration, UserVO1 to add new fields.
8. Click **Cancel** to come back to the home page. After you are done, remember to hide the unauthenticated links in the left navigation pane, then right-click **Unauthenticated Pages**, and select **Hide Component**.

30.4.10 Customizing Certification Pages

The information from the row selected in the certification table can be used for customizing the detail pane found below the table. The procedure in this section can be used to customize the user certification detail pane. The same procedure can be followed for any certfication type.

After you have entered the customization mode, perform the following steps:

1. Edit the panelFormLayout containing the User Detail Information.
2. Click **Add Content**.
3. Select **Data Component - Certification**.
4. Select **UserCertificationUserVO1**.
5. Search for the attribute you want to add, for example Title, and click **Add**.
6. Select ADF Readonly Input Text with Label component.

   The input component is added to the page, but a value for it is not displayed. A label is added that shows the name of the attribute.
7. Select the inputText component in the page source panel, and click **Edit**. The Component Properties dialog box is displayed.
8. Scroll down and find the Value attribute, and open the Expression Builder.
9. Edit the expression value and set it to the following:

   ```
   #{pageFlowScope.p1_row_idcTitle}
   ```

10. Save the changes and close Web Composer. Select a row in the table.

    When a row is selected from the table, the information is stored in the pageFlowScope. To display this information in the detail pane, steps 1 through 10 must be followed to extract the correct data. The format of the EL to follow is:

    ```
    #{pageFlowScope.p1_row_ATTRIBUTE_NAME}
    ```

    Page 1 table information can also be used in page 2 by using the same format. Because the data is stored in the pageFlowScope, the information remains in the scope making it available for display. Page 2 has a Page 1 Detail section at the top of the page showing a reference back to the item on page 1. You can add more page 1 details here using p1_row_*ATTRIBUTE_NAME* in the expression.

    The steps documented in this section apply to page1 or the summmary page, of the current certification. If you want to customize page 2 or the detail page, then use the following format:

    ```
    #{pageFlowScope.p2_row_ATTRIBUTE_NAME}
    ```

30.5 Securing UI Components

This section contains the following topics:

- Securing a Custom Taskflow Using APM
- Securing a Task Flow Region Using EL Expressions

30.5.1 Securing a Custom Taskflow Using APM

You can add permissions to custom taskflows by using the Authorization Policy Manager (APM) UI to secure the taskflow. To do so:

1. Login to Authorization Policy Manager as WebLogic user by navigating to the following URL:

   http://*ADMIN_HOST*:*ADMIN_PORT*/apm

2. Navigate to **Applications**, **OracleIdentityManager**, **Resource Types**. Click **Open**.
3. Click **New** to create a new resource type. Provide following details, and then click **Save**.

   Display Name: A display name for this resource, for example, ADF Taskflows

   Name: A name for this resource, for example, ADFTaskflows

   Actions: personalize, customize, grant, or view. Click **New** to add each action.

   Supports Resource Hierarchy: No

   Resource Delimiter: Slash(/)

   Evaluation Logic: Permission Class

   Permission Class: oracle.adf.controller.security.TaskFlowPermission

   Action Name Delimiter: Comma(,)

4. Navigate to **Applications**, **OracleIdentityManager**, **Default Policy Domain**, **Resource Catalog**, **Resources**. Click **Open**.
5. Click **New** to create a new resource. Provide following values, and then click **Save**.

   Resource Type: Select the resource type created in step 3.

   Display Name: Provide a display name for your custom taskflow.

   Name: Provide the name of the custom taskflow in the following format:

   TASKFLOW_DOCUMENT#TASKFLOW_ID

   For example:

   /WEB-INF/request-approval-details-tf.xml#request-approval-details-tf

   Description: Provide a description for the custom taskflow.

   > **Note:** For each custom taskflow, you must create a resource as mentioned in step 5. You can use the same resource type that you created in step 3 for all your custom taskflows.

6. Navigate to **Applications**, **OracleIdentityManager**, **Default Policy Domain**, **Authorization Policies**. Click **Open**.
7. Select **Find By Principal**, and click **Search**. If you want to add to a displayed existing policy, then select it and click **Open**. Otherwise, click **New** to create a policy.
8. Add name and principals for the new policy.

9. Click **Add Targets**. The Search Targets dialog box is displayed.
10. Click the **Resources** tab. Provide the resource type as defined in step 3, and then click **Search**.
11. Select the resource created in step 5. Click **Add Selected**.
12. Click **Add Targets**. The resource is added to the Targets table.
13. Expand the resource that you added to the table. Select the permissions you want to apply to the taskflow.
14. Click **Apply**.

30.5.2 Securing a Task Flow Region Using EL Expressions

For each new task flow, there is an entry in the jazn-data.xml file, as shown in the following example:

```
<permission>
<class>oracle.adf.controller.security.TaskFlowPermission</class>
<name>/WEB-INF/oracle/iam/ui/catalog/tfs/request-summary-details-tf.xml#request-su
mmary-details-tf</name>
<actions>view</actions>
</permission>
```

This is the basic level of permission required for any task flow to be visible on the Identity Self Service UI. For advanced permissions dependent on admin roles, you can use EL expressions to enforce functional security.

For securing task flows, the task flow must be used as a region in the parent JSFF file. You can define EL expression for the region so that the task flow can be shown or hidden to the logged-in user based on the user's permissions.

For securing a region, consider the following example:

On the my-access-accounts.jsff page, the details-information-tf task flow is rendered selectively to the users by using the following EL expression:

```
rendered="# {oimappinstanceAuth.view [bindings.appInstanceKey].allowed}"
```

Here:

- oimappinstanceAuth is the mapped name of the ApplicationInstanceAuthz.java authorization bean in the adfc-config.xml file.
- view is the name of the UIPermission that needs to be checked. The following permission is defined in ApplicationInstanceAuthz.java, which is the actual bean file for reference of oimappinstanceAuth:

  ```
  Private UIPermission view = new UIPermission
  (PolicyConstants.Resources.APPLICATION_INSTANCE.getId(),
  PolicyConstants.ApplicationInstanceActions.VIEW_SEARCH.getId());
  ```

- appInstanceKey is the ID of the application instance that the user is trying to view, which is passed as a parameter.

30.6 Customizing Oracle Identity Manager Help

Oracle Identity Manager lets you develop and use the following online Help systems in the Oracle Identity Self Service and Oracle Identity System Administration:

- Adding Custom Help Topics

- Adding Inline Help

30.6.1 Adding Custom Help Topics

In addition to the Oracle Identity Manager help topics, you can also create and use custom help topics.

To view the custom help topics:

1. Login to Oracle Identity Self Service.
2. On the navigation bar at the top, click **Help**. The Oracle Help for the Web window is displayed.
3. From the Book list, select **Custom Help Topics for Oracle Identity Manager**.
4. Expand the contents to view the help topics.

The custom help book is provided as a separate JAR file. This is the *OIM_HOME*/help/CUSTOMOHW.jar file. You can create your own help topics and custom help book JAR, and then replace the CUSTOMOHW.jar file to display your custom help topics in the UI.

You create the custom help topics by using Oracle Help for the Web (OHW). For detailed information about creating custom OHW help topics, see *Oracle Fusion Middleware Developer's Guide for Oracle Help*.

After creating the new custom help books, modify the following configuration files in the OIM_HOME/help/ directory to reference the new help books:

- **ohwconfig_identity.xml:** Configuration file for custom help topics in Oracle Identity Self Service
- **ohwconfig_sysadmin.xml:** Configuration file for custom help topics in Oracle Identity System Administration

---

Note: The configuration files are overwritten when you upgrade Oracle Identity Manager, and you must modify the configuration files again to reference the custom help books.

After creating the custom help topics, create the custom help JAR file, and replace the CUSTOMOHW.jar file with the new JAR file. You can now add your custom help topics on the UI pages. The following procedure shows how to add a custom help topic to the Getting Started with Help container of the Home page in the Oracle Identity Self Service:

1. In Oracle Identity Self Service, activate a sandbox from the Manage Sandboxes page.
2. In the Home page, click **Customize**.
3. From the View menu, select **Source**. The object tree is displayed. Open Web Components from Top List.
4. Click the Getting Started Help Topics container. Click **Edit** in the Confirm Edit Task Flow popup.
5. Click **Add Content**. The Add Content dialog box is displayed.
6. Select Command Image Link, and then click **Add** in the same row. The selected component is added to the Getting Started Help Topics section.

7. Select the added component, and click **Edit**. The Component Properties dialog box is displayed.
8. Click the **Display Options** tab.
9. In the Text field, enter the text for the help topic that will be displayed in the page.
10. In the Image field, enter the path and file name for the help icon image.
11. In the Action Listener field, enter the URL with the HelpTopicID of the custom help topic.
12. Click **Apply**, and then click **OK**.
13. Save and close customization mode. The help topic is added to the Getting Started with Help container of the Home page. Clicking the help topic displays the help topic in the custom help book JAR file.

30.6.2 Adding Inline Help

Oracle Identity Manager does not provide inline help by default. However, you can add your inline help for the various UI components, such as add tooltip text for fields and buttons.

The content for the inline help is picked up from the files in the custom WAR library (oracle.iam.ui.custom-dev-starter-pack.war), such as the /oracle/iam/ui/custom/help/CustomHelpResourceBundle.properties file. If the CustomHelpResourceBundle.properties file is not available in the WAR library, then you can create it.

Note: You can change the custom WAR library name. If you do so, then specify the same library name in the entry for the WAR file in Manifest.MF.

You can specify the inline help content through the entries in the CustomHelpResourceBundle.properties file. The entries have a CUSTOMRB prefix, and have any one of the following suffixes:

- _DEFINITION: This specifies inline help for a field or UI component. For example:

  CUSTOMRB_EMAIL_DEFINITION=Enter your official e-mail ID if available.

  EMAIL is the field name, and the value of the entry is the inline help text displayed on placing your mouse pointer on the field.

- _INSTRUCTIONS: This specified inline help for a page layout. For example:

  CUSTOMRB_MY_INFO_INSTRUCTIONS=Profile update will get reflected post approvals.

  MY_INFO is the page, and the value of the entry is the inline help text displayed on the top of the page.

As an example, the following procedure shows how to add inline help to the Telephone field in the My Information page of Oracle Identity Self Service:

1. In the Oracle Identity Self Service, navigate to the My Information page, and expand the Basic User Information section.
2. Click **Customize**.
3. From the View menu, select **Source**. The object tree is displayed.

4. Click the Telephone field.
5. Click **Edit**. The Component Properties dialog box for the Telephone field is displayed.
6. In the Help Topic ID field, enter the help topic ID of the inline help that you want to associate with the Telephone field, such as CUSTOMRB_TELEPHONE.

   Note that specifying the _DEFINITION suffix is not required.
7. Click **Apply**, and then click **OK**.
8. Save and close customization mode. An information image with the interrogation sign (?) is displayed before the Telephone field. When you place the mouse pointer on the icon, the inline help text is displayed.

> **See Also:** "Displaying Tips, Messages, and Help" on the *Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework* for information about defining tips and messages and providing help information for ADF components

30.7 Customizing the Home Page

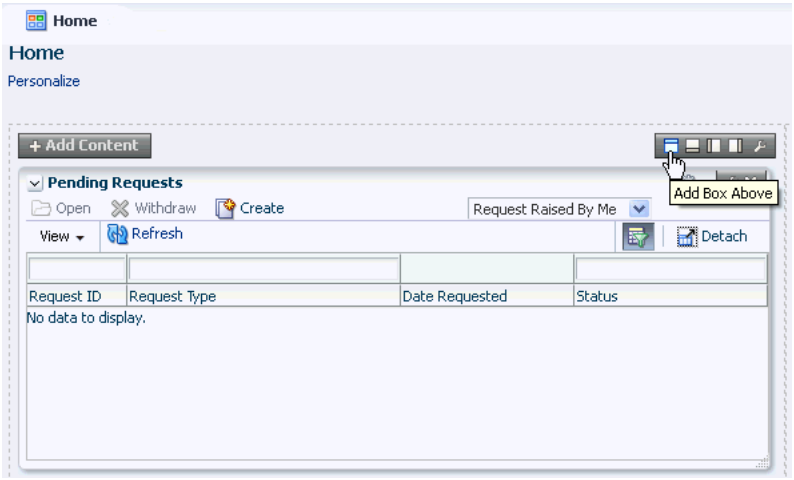
The Home page provides you a snapshot of the various functions in the Oracle Identity Self Service. You can personalize the Home page by adding and removing containers, and rearranging containers. See "Personalizing the Home Page" in the *Oracle Fusion Middleware User's Guide for Oracle Identity Manager* for details.

When you add a container, you must add UI components to the container to use it. To add containers in the Home page and then add UI components:

> **Note:** The personalization capabilities, such as adding and removing containers and changing layouts, are not governed by authorization policies. However, the contents available to each user for adding is governed by authorization policies.

1. Login to Oracle Identity Self Service, and navigate to the Home page.
2. Click **Personalize**. The Home is displayed in customization mode with toolbars that consist of icons.
3. Click the Add Box Above icon on the toolbar. Figure 30–10 shows the Add Box Above icon on the toolbar.

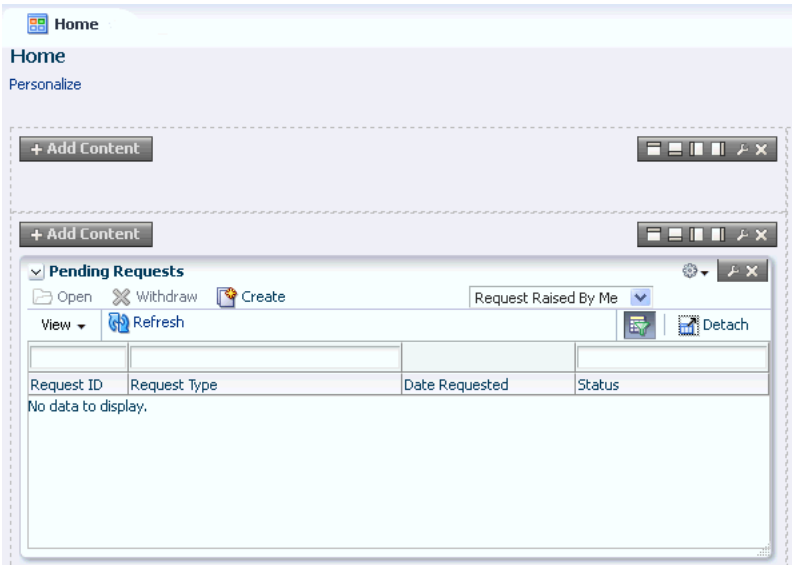
Figure 30–10 The Add Box Above Icon on the Toolbar



You can click the respective icons to add a box below, right, or left of the container in which you are clicking the icon.

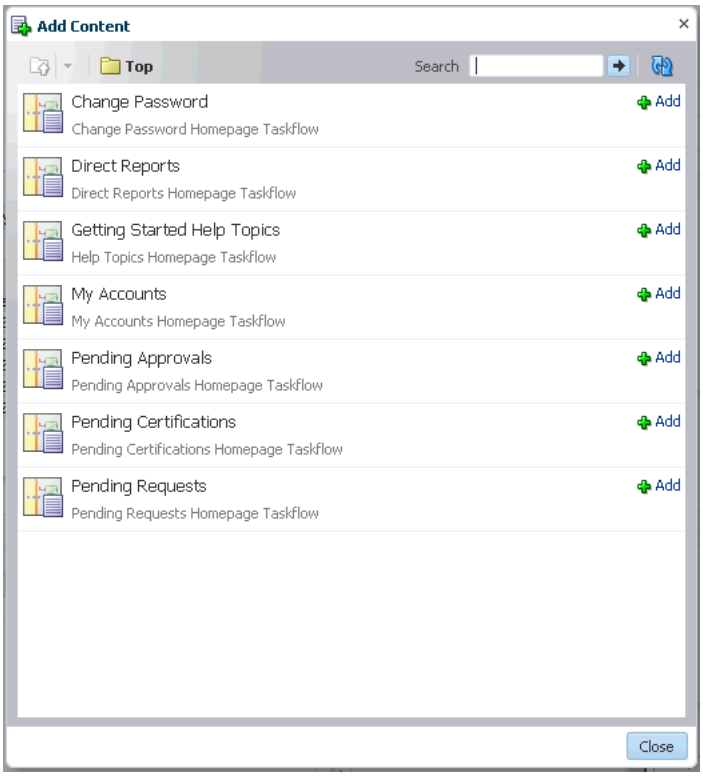
After clicking the icon, a container is added to the Home page, as shown in Figure 30–11:

Figure 30–11 A New Container



4. Click **Add Content**. The Add Content dialog box is displayed with the list of homepage task flows that you can add to the container. Figure 30–12 shows the Add Content dialog box.

Figure 30–12 The Add Content Dialog Box



5. Select the homepage task flow, and click **Add**. Then, close the Add Content dialog box.
6. Click **Close** on the navigation bar at the top to quit customization mode.

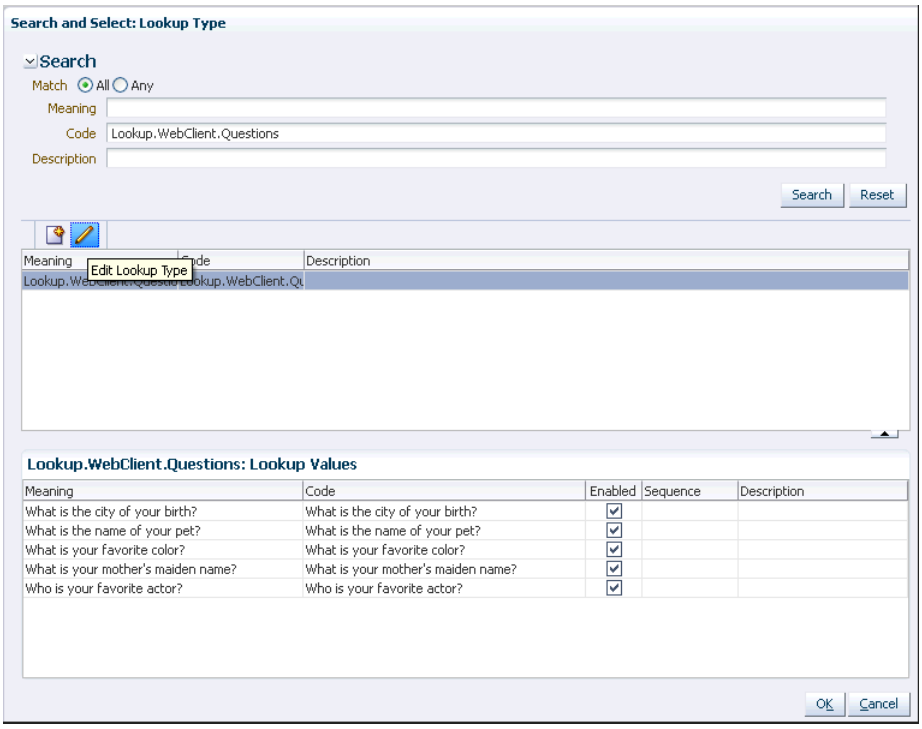
30.8 Customizing Challenge Questions

You can customize the number of challenge questions in multiple pages of the UI. To do so, change the value of the PCQ.NO_OF_QUES system property to specify the number of challenge questions that you want to display to the user. In addition, you customize the pages to show or hide the challenge questions that you add or remove respectively.

To set the number of challenge questions:

1. Login to Oracle Identity System Administration.
2. On the left pane, under System Management, click **System Configuration**. The System Configuration tab is displayed in a new window.
3. Search and open the 'Number of Questions' system property that has the PCQ.NO_OF_QUES keyword.
4. Change the value from 3 to 5, and click **Save**.
5. Close the new window.
6. In the Oracle Identity System Administration, under Configuration, click **Lookups**, and search for the Lookup.WebClient.Questions code, as shown in Figure 30–13.

Figure 30–13 The Lookup.Weblciient.Questions Lookup Code



7. Add additional questions by editing the lookup type.

> **Note:** You can localize the questions for en_US bundle, as described in "Localizing Challenge Questions and Responses" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager*.

To set challenge questions for the user who is logging in to Oracle Identity Self Service or Oracle Identity System Administration:

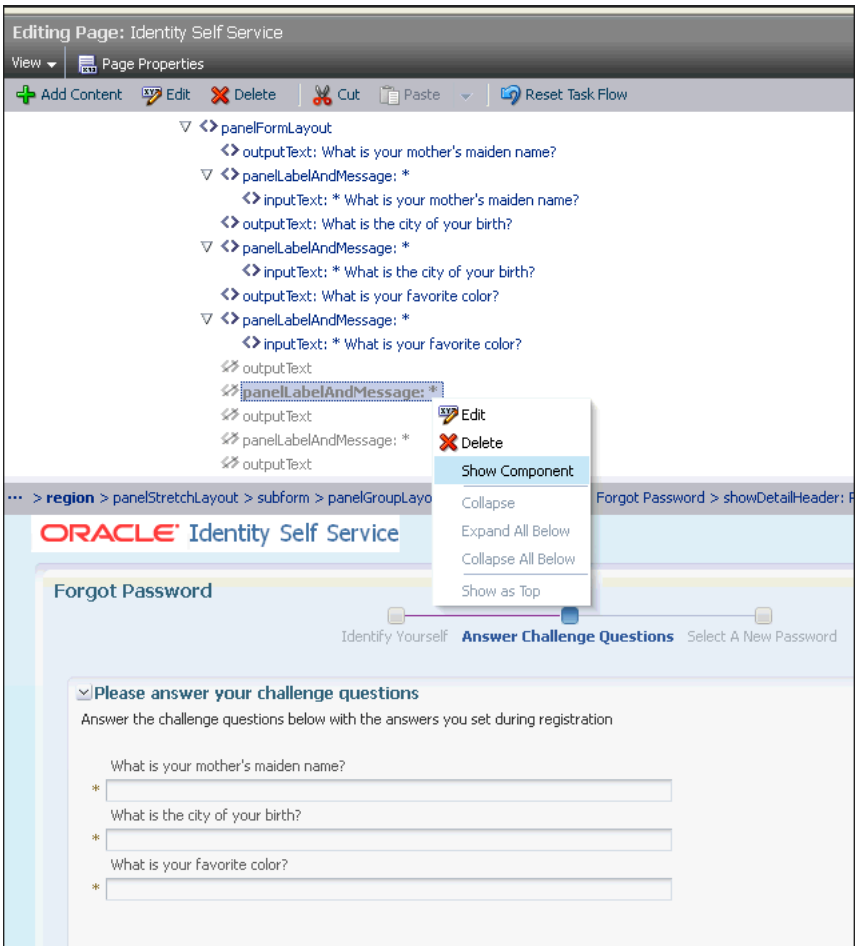
1. Login to Oracle Identity Self Service as the System Administrator, and on home page click on Sandboxes.
2. On the Home page, click **Sandboxes** on the upper navigation bar. The Manage Sandboxes page is displayed.
3. On the toolbar, click **Create Sandbox**. In the Create Sandbox dialog box, enter a sandbox name, for example ChallengeQ, and click **Save and Close**.

   Leave the **Activate Sandbox** check box selected.
4. On the confirmation dialog box, click **OK**. The active sandbox is displayed in the table in the Manage Sandboxes page. Close the **Manage Sandboxes** page.
5. Open the My Information page. Collapse the Basic Information section, and expand the Challenge Questions section.
6. Click **Customize**. Then the toolbar, select **View**, **Source**.
7. Click the third question, and click **OK** to confirm editing the taskflow.
8. In the source tree, select the greyed-out panelGroupLayout just after the third question. This contains the fourth question. Right-click this panelGroupLayout, and then select **Show Component**.

9. Similarly, show the fifth question.
10. Click **Close** on the toolbar.
11. Repeat steps 4 through 9 for the other three pages on which challenge questions are displayed by pointing the browser to the following URLs:
    - The User Registration page: /identity/faces/register
    - The Forgot Password page: /identity/faces/forgotpassword
    - The Login page when the user logs in for the first time: /identity/faces/firstlogin?action=setchallenges

    Note that there is some variation by design, for example, the question on the Forgot Password page is rendered as outputText rather than a list, as shown in Figure 30–14:

Figure 30–14 Challenge Question on the Forgot Password Page



The number of questions displayed in the pages is relative. For example, JohnD is the user login provided at the first step of forgot password. If JohhD does not existent or JohnD has not set the challenge responses for self, then all the configured questions are displayed in the Forgot Password page. If JohnD has set some challenge questions for self, then only those questions are displayed.

12. Publish the sandbox and logout.

30.9 Customizing the Transitional UI

In the transitional UI pages of the Identity Self Service, 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

30.9.1 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 Identity Self Service or Identity System Administration.
   - The order in which the items appear in the Code Key list are the order they appear in the drop-down list in the Identity Self Service. If you delete an entry and add it back, it is displayed last in the list.
5. Save your changes.

30.9.2 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 30–6, and edit it as required.

Table 30–6 Properties that Determine the Number of Menus on a Search Page

Property Name	Default	Page
global.property.numsearchaccesspolicyfields	2	Access Policies
global.property.numsearchresourcefields	2	Search Resources
global.property.numsearchattestationprocessfields	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.

30.10 Developing Managed Beans and Task Flows

To implement advanced customization in Oracle Identity Manager, you can develop new task flows and managed beans by using JDeveloper IDE and then package them in the custom WAR file.

> **See Also:** Figure 30–1, "Oracle Identity Manager UI Libraries" for information about the various Oracle Identity Manager UI libraries and their dependency structure

The beans are of the following types:

- **Request beans:** New instance of the bean is created for every request.jsff component bindings, and listeners are usually bound to request beans.
- **State beans:** Beans holding the state of the application, user session, or a particular flow. Values of components, such as af:inputText, can be bound to state beans. State beans must be serializable (implement java.io.Serializable) as ADF serializes/deserializes these beans between requests.

This section describes how to develop managed beans in the following topics:

- Setting Up the ViewController Project
- Setting Up a Model Project
- Adding Custom Managed Bean
- Deploying Custom Code to Oracle Identity Manager
- Using Managed Beans
- Using Managed Beans to Populate Request Attributes
- Using Public Taskflows

30.10.1 Setting Up the ViewController Project

Managed beans are created in a ViewController project. All your custom taskflows, pages, and managed beans must be present in the ViewController project.

To setup the ViewController project:

1. Create a new JDeveloper application. To do so:
   a. Start JDeveloper.

b. Select **File**, **New**.

c. Select **Generic Application**, and then click **OK**.

d. Provide the application name and directory, and then click **Finish**. The application is created using a sample project.

e. To delete the sample project, right-click the project, and select **Delete**.

2. Setup the ViewController project. To do so:

a. Select **File**, **New**.

b. Find and select **ADF ViewController Project**, and then click **OK**.

c. Provide the project name, for example CustomUI, and project directory, and then click **Next**.

d. Enter the default package name as **oracle.iam.ui.custom**, and then click **Finish**. The new project is created.

3. Add Oracle Identity Manager libraries to the project classpath. To do so:

a. Right-click the new project, and select **Project Properties**.

b. On the left navigation bar, select **Libraries and Classpath**.

c. Click **Add Library**. Add ADF Model Runtime.

d. Click **Load Dir**, provide the path as ***IDM_HOME*/server/jdev.lib**, and then click **OK**.

e. From the list of libraries, select the following:

– **OIM View Shared Library**

– **OIM Model Shared Library**

– **OIM Client Library**

f. Click **OK**.

4. Define the deployment profile for the newly created ViewController project. To do so:

a. Right-click the project, and select **Project Properties**.

b. On the left navigation bar, select **Deployment**.

c. Delete any existing deployment profiles.

d. Click **New**, and select **ADF Library JAR File** as the archive type.

Note: The ADF Library JAR File and JAR File archive types are different. Make sure that you select the **ADF Library JAR File** archive type.

---

e. Provide and confirm the archive name, such as adflibCustomUI, and then click **OK**.

Your ViewController project setup is complete. You can now start adding custom taskflows, pages, and managed beans.

Note: Some examples in the consecutive sections in this document use the FacesUtils class. For information about this class, see Appendix B, "The FacesUtils Class".

30.10.2 Setting Up a Model Project

All your custom EOs/VOs and classes interacting directly with Oracle Identity Manager APIs must be present in a model project. To setup the model project:

1. Click **File**, **New**.
2. Find and select **ADF Model Project**, and then click **OK**.
3. Provide the Project Name, for example CustomModel, and Project Directory, and then click **Next**.
4. Enter Default Package name as **oracle.iam.ui.custom**, and then click **Finish**. The new project is created.
5. Add Oracle Identity Manager libraries to the project classpath:
   a. Right-click the project, and select **Project Properties**.
   b. On the left navigation bar, select **Libraries and Classpath**.
   c. Click **Add Library**.
   d. Click **Load Dir**, provide the path as ***IDM_HOME*/server/jdev.lib**, and then click **OK**.
   e. From the list of libraries select the following:
      – **OIM Model Shared Library**
      – **OIM Client Library**
   f. Click **OK**.
6. Define the deployment profile for the newly created model project. To do so:
   a. Right-click the project, and select **Project Properties**.
   b. On the left navigation bar, select **Deployment**.
   c. Delete any existing deployment profiles.
   d. Click **New**, and select **ADF Library JAR File** as the archive type.

      Note: The ADF Library JAR File and JAR File archive types are different. Make sure that you select the **ADF Library JAR File** archive type.

   e. Provide and confirm the archive name, such as adflibCustomModel, and then click **OK**.

Your model project setup is complete. You can now start adding custom EOs, VOs, and classes for interacting with Oracle Identity Manager APIs.

Note: Some examples in the consecutive sections in this document use the FacesUtils class. For information about this class, see Appendix B, "The FacesUtils Class".

30.10.3 Adding Custom Managed Bean

To add your custom managed bean:

1. Right-click the ViewController project, and select **New**.
2. Select the Java Class category.
3. Provide the class name, for example CustomReqBean or CustomStateBean, and the package name.
4. After creating the class, to register it with a taskflow:
   a. If you are developing your own bounded task flow, then navigate to your task flow definition file, and open it. Otherwise, locate the adfc-config.xml file in your ViewController project, and open it.
   b. Click the **Overview** tab, and select **Managed Beans**.
   c. Add a new managed bean entry. To do so:

      i) Provide managed bean name, for example customReqBean or customStateBean. This is the name that you will later use to refer to an instance of your bean.

      ii) Provide the managed bean class name.

      iii) Provide the scope. For request beans use backingBean scope. For state beans, use pageFlow scope.

Note:

- The pageFlow scope beans are visible only in the taskflow for which they are defined.
- To refer to your managed bean from JSFF/taskflow definition or other places, you can use EL expression. For example, if you register your bean under the name customReqBean and put the bean to backingBean scope, then you can reference your bean by using the following EL expression:

  ```
  #{backingBeanScope.customReqBean}
  ```

  If you put the bean to pageFlow scope, you can reference your bean by using the following EL expression:

  ```
  #{pageFlowScope.customStateBean}
  ```

30.10.4 Deploying Custom Code to Oracle Identity Manager

To deploy an ADF library JAR file produced by your custom model or ViewController projects:

1. Locate the oracle.iam.ui.custom shared library, which is oracle.iam.ui.custom-dev-starter-pack.war. The shared library is in the *IDM_HOME*/server/apps/ directory.
2. Repackage the WAR and include your ADF libraries in WEB-INF/lib/.
3. Redeploy the shared library.

30.10.5 Using Managed Beans

This section provides the following use cases for developing managed beans to customize Oracle Identity Manager interface:

- Showing Components Conditionally
- Prepopulating Fields Conditionally
- Setting a Conditional Mandatory Field
- Implementing Custom Field Validation
- Implementing Custom Cascading LOVs
- Customizing Forms By Using RequestFormContext
- Overriding the Submit Button in Request Catalog
- Developing Home Page Portlets
- Launching Taskflows
- Creating an External Link

Note: The examples in this section use the FacesUtils class. For information about this class, see "The FacesUtils Class" on page B-1.

30.10.5.1 Showing Components Conditionally

You can show or hide certain fields conditionally based on the values of other fields. For example, to show the Contact Information panel on the Create User page only when the User Type is Full-Time Employee, perform the following steps:

1. In your custom request bean, define properties for component bindings of the User Type field and any parent component of the Contact Information panel, for example, the form root panel. To do so, use the following code:

```
private UIComponent rootPanelPGL;
    private UIComponent userTypeSOC;

    public void setRootPanelPGL(UIComponent rootPanelPGL) {
        this.rootPanelPGL = rootPanelPGL;
    }

    public UIComponent getRootPanelPGL() {
        return rootPanelPGL;
    }

    public void setUserTypeSOC(UIComponent userTypeSOC) {
        this.userTypeSOC = userTypeSOC;
    }

    public UIComponent getUserTypeSOC() {
        return userTypeSOC;
    }
```

2. Create or extend existing valueChangeListener that will be invoked when user selects the new value in the User Type list. To do so, use the following code:

Note: The listener will refresh the form.

```
public void valueChangeListener(ValueChangeEvent valueChangeEvent) {
        if (valueChangeEvent.getSource().equals(userTypeSOC)) {
            // refresh form
            AdfFacesContext.getCurrentInstance().addPartialTarget(component);
        }
    }
```

3. Create a method that returns boolean value. The method will determine if the Contact Information panel is to be displayed when the page is rendered. In this example, the Contact Information panel will be shown if the User Type is Full-Time Employee.

   The method is as follows:

```
private static final String USER_TYPE_ATTRIBUTE = "usr_emp_type__c";

    public boolean isFullTimeEmployeeUserTypeSelected() {
        // return true if value of "usr_emp_type__c" binding attribute equals
to "Full-Time"
       // "usr_emp_type__c" binding attribute is used to display value of User
Type in the User Type drop-down
        return "Full-Time".equals(FacesUtils.getListBindingValue(USER_TYPE_
ATTRIBUTE, String.class));
    }
```

4. Package and deploy the managed bean, as described in "Developing Managed Beans and Task Flows".

5. To bind the code with JSFF:

   a. Set component bindings for the User Type list and root panel components to point to the properties that you defined.

   b. Define the valueChangeListener for the User Type list.

   Note: Make sure that the autosubmit property is set to true for the User Type list.

   c. Set EL expression for the rendered property, which is Show Component in Web Composer, on the Contact Information panel to point to the isFullTimeEmployeeUserTypeSelected() method defined in step 3.

   Note: Ignore if the following error is displayed while setting EL expression for the rendered property:

```
"javax.faces.validator.ValidatorException:
java.lang.IllegalArgumentException: Control Binding 'usr_emp_type__
c' not found"
```

30.10.5.2 Prepopulating Fields Conditionally

You prepopulate certain fields based on the values of other fields. For example, to prepopulate values in the User Login and E-mail fields on the Create User page based on the values of the First Name and Last Name fields, perform the following steps:

1. In your custom request bean, define properties for component bindings of First Name and Last Name fields and any parent component of the User Login and E-mail fields, for example, form root panel. To do so, use the following code:

```
private UIComponent firstNameIT;
    private UIComponent lastNameIT;
    private UIComponent rootPanelPGL;

    public void setFirstNameIT(UIComponent firstNameIT) {
        this.firstNameIT = firstNameIT;
    }

    public UIComponent getFirstNameIT() {
        return firstNameIT;
    }

    public void setLastNameIT(UIComponent lastNameIT) {
        this.lastNameIT = lastNameIT;
    }

    public UIComponent getLastNameIT() {
        return lastNameIT;
    }

    public void setRootPanelPGL(UIComponent rootPanelPGL) {
        this.rootPanelPGL = rootPanelPGL;
    }

    public UIComponent getRootPanelPGL() {
        return rootPanelPGL;
    }
```

2. Create or extend existing valueChangeListener that will be invoked when the user updates the First Name or Last Name fields. To do so, use the following code:

> **Note:** The listener will update User Login and E-mail accordingly and refresh the form.

```
private static final String USER_LOGIN_ATTRIBUTE = "usr_login__c";
    private static final String EMAIL_ATTRIBUTE = "usr_email__c";
    private static final String LAST_NAME_ATTRIBUTE = "usr_last_name__c";
    private static final String FIRST_NAME_ATTRIBUTE = "usr_first_name__c";

    public void valueChangeListener(ValueChangeEvent valueChangeEvent) {
        if (valueChangeEvent.getSource().equals(firstNameIT)) {
            // get new value of first name from the event
           String firstName = (String)valueChangeEvent.getNewValue();
            // get existing value of last name through binding
            String lastName = FacesUtils.getAttributeBindingValue(LAST_NAME_
ATTRIBUTE, String.class);
            setUserLoginAndEmail(firstName, lastName);
        } else if (valueChangeEvent.getSource().equals(lastNameIT)) {
            // get existing value of first name through binding
            String firstName = FacesUtils.getAttributeBindingValue(FIRST_NAME_
ATTRIBUTE, String.class);
            // get new value of last name from the event
            String lastName = (String)valueChangeEvent.getNewValue();
            setUserLoginAndEmail(firstName, lastName);
        }
        // refresh form
        FacesUtils.partialRender(rootPanelPGL);
    }
```

```
    private void setUserLoginAndEmail(String firstName, String lastName) {
        StringBuilder sb = new StringBuilder();
        if (firstName != null) {
            sb.append(firstName);
        }
        if (firstName != null && !firstName.isEmpty() && lastName != null &&
!lastName.isEmpty()) {
            sb.append(".");
        }
        if (lastName != null) {
            sb.append(lastName);
        }
        String userLogin = sb.toString();
        // set new value for User Login and E-mail through binding
        FacesUtils.setAttributeBindingValue(USER_LOGIN_ATTRIBUTE, userLogin);
        FacesUtils.setAttributeBindingValue(EMAIL_ATTRIBUTE, userLogin +
"@acme.com");
}
```

3. Package and deploy the managed bean, as described in "Developing Managed Beans and Task Flows".
4. Add the code to the JSFF. To do so:
   a. Set the component bindings for First Name, Last Name, and root panel to point to the properties that you defined.
   b. Define valueChangeListener for First Name and Last Name input texts, and make sure that the autosubmit property is set to true on both input texts.

30.10.5.3 Setting a Conditional Mandatory Field

You can make a field conditionally mandatory based on the value of another field. For example, to make the Manager field on the Create User page mandatory only if the User Type is Intern, perform the following steps:

> **Note:** Enforcing field validation cannot be performed by setting the required property in Web Composer. You must develop a managed bean to perform field validation, as described in this section.

1. In your custom request bean, define properties for component bindings of the User Type field and any parent component of Manager field, for example, form root panel. To do so, use the following code:

```
private UIComponent rootPanelPGL;
    private UIComponent userTypeSOC;

    public void setRootPanelPGL(UIComponent rootPanelPGL) {
        this.rootPanelPGL = rootPanelPGL;
    }

    public UIComponent getRootPanelPGL() {
        return rootPanelPGL;
    }

    public void setUserTypeSOC(UIComponent userTypeSOC) {
        this.userTypeSOC = userTypeSOC;
    }
```

```
    public UIComponent getUserTypeSOC() {
        return userTypeSOC;
    }
```

2. Create or extend existing valueChangeListener that will be invoked when user selects new value in the User Type list. To do so, use the following code:

   Note: The listener will refresh the form.

```
public void valueChangeListener(ValueChangeEvent valueChangeEvent) {
        if (valueChangeEvent.getSource().equals(userTypeSOC)) {
            // refresh form
            FacesUtils.partialRender(rootPanelPGL);
        }
    }
```

3. Create a method that returns boolean value. The method determines whether or not the field is mandatory. In this example, the Manager field will be marked as mandatory if User Type is Intern.

   The method is as follows:

```
    public boolean isInternUserTypeSelected() {
        // return true if value of "usr_emp_type__c" binding attribute equals
to "Intern"
       // "usr_emp_type__c" binding attribute is used to display value of User
Type in the User Type drop-down
return "Intern".equals(FacesUtils.getValueFromELExpression("#{bindings.usr_emp_
type__c.attributeValue}", String.class));
    }
```

4. Package and deploy the managed bean, as described in "Developing Managed Beans and Task Flows".

5. Add the code to the JSFF. To do so:

   a. Set component bindings for the User Type list and root panel components to point to the properties you defined.

   b. Define valueChangeListener for the User Type list. Make sure that the autosubmit property is set to true for the User Type list.

   c. Set EL expression for the required property on the Manager field to point to the isInternUserTypeSelected() method defined is step 3.

   d. Set EL expression for the Show required property on the Manager field panelLabelAndMessage to point to the isInternUserTypeSelected() method defined is step 3.

30.10.5.4 Implementing Custom Field Validation

Managed beans can be used to introduce custom validations. For example, you can implement the following validations for the Start Date and End Date fields on the Account Effective Dates panel of the Create User page:

- Start Date cannot be after End Date.
- The interval between Start Date and End Date cannot exceed 180 days for Contractors.

To implement custom validation using Managed Beans:

1. In your custom request bean, define properties for component bindings of the Start Date and End Date fields, as shown:

```
private UIComponent startDateID;
private UIComponent endDateID;

public void setStartDateID(UIComponent startDateID) {
    this.startDateID = startDateID;
}

public UIComponent getStartDateID() {
    return startDateID;
}

public void setEndDateID(UIComponent endDateID) {
    this.endDateID = endDateID;
}

public UIComponent getEndDateID() {
    return endDateID;
}
```

2. Add method for validation in your managed bean that will be invoked when the user selects new value for the Start Date or End Date field. The validator generates an error message when validation fails and attaches it to the field being updated. To do so, use the following code:

```
private static final String START_DATE_END_DATE_VALIDATION_MSG = "Start Date -
End Date interval cannot exceed 180 days for Contractors.";
    private static final String START_DATE_AFTER_END_DATE_VALIDATION_MSG =
"Start Date cannot be after End Date.";

    private static final String USER_TYPE_ATTRIBUTE = "usr_emp_type__c";
    private static final String START_DATE_ATTRIBUTE = "usr_start_date__c";
    private static final String END_DATE_ATTRIBUTE = "usr_end_date__c";

    public void validator(FacesContext facesContext, UIComponent uiComponent,
Object object) {
        if (uiComponent.equals(startDateID)) {
            // get value of End Date through binding
            oracle.jbo.domain.Date jboEndDate =
FacesUtils.getAttributeBindingValue(END_DATE_ATTRIBUTE,
oracle.jbo.domain.Date.class);
            // only validate if both Start Date and End Date are set
            if (jboEndDate != null) {
                // value of Start Date is passed to validator
                Date startDate = ((oracle.jbo.domain.Date)object).getValue();
                Date endDate = jboEndDate.getValue();
                validateStartDateEndDate(facesContext, uiComponent, startDate,
endDate);
            }
        } else if (uiComponent.equals(endDateID)) {
            // get value of Start Date through binding
            oracle.jbo.domain.Date jboStartDate =
FacesUtils.getAttributeBindingValue(START_DATE_ATTRIBUTE,
oracle.jbo.domain.Date.class);
            // only validate if both Start Date and End Date are set
            if (jboStartDate != null) {
                Date startDate = jboStartDate.getValue();
```

```
                // value of End Date is passed to validator
                Date endDate = ((oracle.jbo.domain.Date)object).getValue();
                validateStartDateEndDate(facesContext, uiComponent, startDate,
endDate);
            }
        }
    }

    private void validateStartDateEndDate(FacesContext facesContext,
UIComponent uiComponent, Date startDate, Date endDate) {
        Date startDatePlus180Days = new Date(startDate.getTime() + 180L * 24 *
60 * 60 * 1000);
        if (startDate.after(endDate)) {
            // queue error message for the component which is being validated
(either Start Date or End Date)
            facesContext.addMessage(uiComponent.getClientId(facesContext),
                                    new FacesMessage(FacesMessage.SEVERITY_
ERROR, START_DATE_AFTER_END_DATE_VALIDATION_MSG, null));
        } else if (isContractorUserTypeSelected() &&
startDatePlus180Days.before(endDate)) {
            // queue error message for the component which is being validated
(either Start Date or End Date)
            facesContext.addMessage(uiComponent.getClientId(facesContext),
                                    new FacesMessage(FacesMessage.SEVERITY_
ERROR, START_DATE_END_DATE_VALIDATION_MSG, null));
        } else {
           // re-render -- in case there was an error message in queue for any
of the two components it will be released
            FacesUtils.partialRender(startDateID);
            FacesUtils.partialRender(endDateID);
        }
    }

    public boolean isContractorUserTypeSelected() {
        // return true if value of "usr_emp_type__c" binding attribute equals
to "Contractor"
       // "usr_emp_type__c" binding attribute is used to display value of User
Type in the User Type drop-down
        return "Contractor".equals(FacesUtils.getListBindingValue(USER_TYPE_
ATTRIBUTE, String.class));
    }
```

See Also: "The FacesUtils Class" on page B-1 for more information about the FacesUtils class

3. Package and deploy the managed bean, as described in "Developing Managed Beans and Task Flows".
4. Bind the code to the JSFF. To do so:
   a. Set component bindings for the Start Date and End Date fields to point to the properties that you defined.
   b. Define EL expression for validator property on Start Date and End Date fields to point to the validator method that you defined in step 2. For example:

```
<mds:attribute name="binding"
value="#{backingBeanScope.validatorBean.startDateID}"/>
      <mds:attribute name="validator"
value="#{backingBeanScope.validatorBean.validator}"/>
```

Note: The validator property cannot be added directly by using the Web Composer. This must be set manually in the MDS file for the JSFF. To do so:

1. Export the sandbox after setting component bindings for the Start Date and End Date fields by using the Web Composer.
2. Extract the contents of the ZIP file and locate the XML file for the form on which Start Date and End fields are modified. For example, the XML file for the Create User form is oracle/iam/ui/runtime/form/view/pages/mdssys/cust/site/site/userCreateForm.jsff.xml.
3. In a text editor, open the XML file and set validator for StartDate and EndDate fields. For Example:

```
<mds:modify element="_xg_36">
       <mds:attribute name="binding"
value="#{backingBeanScope.validatorBean.startDateID}"/>
      <mds:attribute name="validator"
value="#{backingBeanScope.validatorBean.validator}"/>
   </mds:modify>
   <mds:modify element="_xg_13">
      <mds:attribute name="binding"
value="#{backingBeanScope.validatorBean.endDateID}"/>
      <mds:attribute name="validator"
value="#{backingBeanScope.validatorBean.validator}"/>
   </mds:modify>
```

4. Save the changes, repackage the ZIP file (the sandbox archive), and then import it back to your environment.

30.10.5.5 Implementing Custom Cascading LOVs

Cascading LOVs are LOV components for which the list of values in one component is dependent on the currently selected value in another component. For example, based on the selected value in the User Type list on the Create User page, you might want to display the Job Code list or another LOV component whose list of values is dependent on the currently selected value in the User Type list.

The following are the high-level guidelines to implement custom cascading LOVs:

1. Define component binding for the User Type field and any parent component of Job Code, for example, form root panel.
2. Implement the model for Job Code LOV component by ensuring the following:
   - The model must take into account the current value of the User Type field.
   - For af:selectOneChoice, you must implement a method that returns List<javax.faces.model.SelectItem>.
   - For af:inputListOfValues, you must implement a method that returns an instance of oracle.adf.view.rich.model.ListOfValuesMode.

     See Also: "Using List-of-Values Components" in the *Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework* for information about using a LOV component to display a model-driven list of objects from which a user can select a value

3. Implement valueChangeListener for the User Type field. Set the autosubmit property to true for the User Type field.

   valueChangeListener must update model of Job Code LOV component with the current value of the User Type field. In addition, valueChangeListener must re-render the form so that Job Code LOV component is updated with the current list of values.

30.10.5.6 Customizing Forms By Using RequestFormContext

RequestFormContext is a bean available in the pageFlowScope of entity form details taskflow. The entity forms include user form, application instance form, role form, and entitlement form. The instance provides various context information. Using this context information, you can customize various forms based on specific business requirements.

You can get an instance of the class by using Java code, as shown:

```
RequestFormContext.getCurrentInstance();
```

You can also get an instance of the class by using EL, as shown:

```
#{pageFlowScope.requestFormContext}
```

RequestFormContext provides the following context information:

- **operation:** The operation that is being performed on the entity. The possible values are CREATE and MODIFY.
- **actionType:** The action that is being performed by the user when the entity form is displayed. The possible values are: APPROVAL, FULFILL, REQUEST, VIEW, SUMMARY.
- **bulk:** Whether or not it is a bulk operation.
- **beneficiaryIds:** The list of beneficiary or target user IDs. For example, if you are requesting an application instance for the user John Doe, then the list contains the user ID of John Doe.
- **cartItemIds:** The list of cart item IDs. For example, if you are requesting an application instance for a user, then the list contains the application instance ID that is being requested.
- **requestEntityType:** The entity type being requested, which is any one of ROLE, ENTITLEMENT, APP_INSTANCE, USER.
- **requestEntitySubType:** The subtype of entity being requested. For example, when requesting for an application instance, the requestEntitySubType is the application instance key.
- **instanceKey:** The key of the instance being modified.

The following is an example usage of the RequestFormContext:

You might want to add new Prepopulate button to the Create Application Instance form, and make the button visible only when there is only one target user. When the button is clicked, some of the application instance fields, such as User Login, First Name, and Last Name) will be prepopulated based on the current target user. To achieve this, perform the following steps:

1. In your custom request bean, define properties for component bindings of the Prepopulate button and the form root panel, as shown:

   ```
   private UIComponent rootPanel;
   ```

```
    private UIComponent prepopulateButton;

    public void setRootPanel(UIComponent rootPanel) {
        this.rootPanel = rootPanel;
    }

    public UIComponent getRootPanel() {
        return rootPanel;
    }

    public void setPrepopulateButton(UIComponent prepopulateButton) {
        this.prepopulateButton = prepopulateButton;
    }

    public UIComponent getPrepopulateButton() {
        return prepopulateButton;
    }
```

2. Implement an actionListener that will be invoked when the Prepopulate button is clicked. The actionListener uses the target user ID and fetches user data, such as First Name and Last Name, by using Oracle Identity Manager API. Use the fetched data, and set certain application instance attributes through attribute binding, and finally refresh the form so that new values are displayed. The actionListener is as shown:

```
private static final String
ACCOUNT_LOGIN_ATTRIBUTE = "UD_EBS2722_LOGIN__c";
    private static final String ACCOUNT_ID_ATTRIBUTE = "UD_EBS2722_ACCOUNTID__
c";
    private static final String FIRST_NAME_ATTRIBUTE = "firstName__c";
    private static final String LAST_NAME_ATTRIBUTE = "lastName__c";
public void actionListener(ActionEvent e) {
        if (e.getSource().equals(prepopulateButton)) {
            RequestFormContext requestFormContext =
RequestFormContext.getCurrentInstance();
            List<String> beneficiaryIds =
requestFormContext.getBeneficiaryIds();
            if (beneficiaryIds.size() == 1) {
                // prepopulate fields based on selected beneficiary
                User user = getUser(beneficiaryIds.get(0));
                FacesUtils.setAttributeBindingValue(ACCOUNT_LOGIN_ATTRIBUTE,
user.getLogin());
                FacesUtils.setAttributeBindingValue(ACCOUNT_ID_ATTRIBUTE,
user.getId());
                FacesUtils.setAttributeBindingValue(FIRST_NAME_ATTRIBUTE,
user.getFirstName());
                FacesUtils.setAttributeBindingValue(LAST_NAME_ATTRIBUTE,
user.getLastName());
            }
        }
        FacesUtils.partialRender(rootPanel);
    }

    private User getUser(String userId) {
        UserManager userManager = OIMClientFactory.getUserManager();
        try {
            return userManager.getDetails(userId, null, false);
        } catch (NoSuchUserException e) {
            throw new RuntimeException(e);
        } catch (UserLookupException e) {
```

```
            throw new RuntimeException(e);
        }
    }
```

3. Create a method that returns Boolean value. The method determines if the Prepopulate button is to be displayed when the form is rendered. In this example, the Prepopulate button will be displayed when the number of target users is equal to 1. The method is as follows:

```
public boolean isPrepopulateButtonRendered() {
        RequestFormContext requestFormContext =
RequestFormContext.getCurrentInstance();
        return requestFormContext.getActionType() ==
RequestFormContext.ActionType.REQUEST &&
requestFormContext.getBeneficiaryIds().size() == 1;
    }
```

4. Bind the code with JSFF. To do so:

   a. Add a Prepopulate button to the Create Application Instance form.

   b. Set bindings for the Prepopulate button and the root panel.

   c. Set the Prepopulate button actionListener property to point to the actionListener method implemented in step 2.

   Note: The actionListener property cannot be set by using the Web Composer. This must be set manually as follows:

   1. Export the sandbox.
   2. Edit the JSFF to set the actionListener attribute value. For example:

      ```
      <mds:attribute name="actionListener"
      value="#{backingBeanScope.accountFormReqBean.submitButtonAction
      Listener}"/>)
      ```

   3. Import the updated sandbox.

   This procedure is applicable to setting the actionListener property in all the examples in this document.

   ---

   d. Set the rendered property to point to the isPrepopulateButtonRendered() method implemented in step 3.

30.10.5.7 Overriding the Submit Button in Request Catalog

Can override the Submit button in the request catalog and execute additional logic based on your requirements. For example, to add additional check for number of target users or beneficiaries when submitting a request, and allow submitting the request when the number of beneficiaries is not more than five, perform the following steps:

1. Implement actionListener that will override the original Submit button.

   The actionListener will be invoked when the user clicks the Submit button. The actionListener performs the extra check and either display error messages or executes the original actionListener bound to the Submit button. Original Submit button actionListener can be executed using the following EL expression:

   ```
   #{backingBeanScope.cartReqBean.submitActionListener}
   ```

The actionListener code is as shown:

```
private static final String MORE_THAN_FIVE_TARGET_USERS_MSG = "Cannot submit
request for more than five target users.";
public void submitButtonActionListener(ActionEvent e) {
        // only submit request if there is no more than 5 beneficiaries
        Boolean moreThanFiveTargetUsers =
FacesUtils.getValueFromELExpression("#{backingBeanScope.cartReqBean.targetUserS
ize > 5}", Boolean.class);
        if (moreThanFiveTargetUsers) {
            // display error message
            FacesMessage fm = new FacesMessage();
            fm.setSeverity(FacesMessage.SEVERITY_ERROR);
            fm.setSummary(MORE_THAN_FIVE_TARGET_USERS_MSG);
            FacesUtils.showFacesMessage(fm);
        } else {
            // execute original submit button action listener
            MethodExpression originalActionListener =

FacesUtils.getMethodExpressionFromEL("#{backingBeanScope.cartReqBean.submitActi
onListener}", null, new Class[] { ActionEvent.class });
            originalActionListener.invoke(FacesUtils.getELContext(), new
Object[] { e });
        }
    }
```

2. Update the Submit button actionListener property to point to the new actionListener implementation.

30.10.5.8 Developing Home Page Portlets

Hompage portlet is a regular bounded taskflow that can be added to the user's Home page in Oracle Identity Self Service. Home page portlets must be created in ViewController projects. ViewController projects must have the deployment profile of type ADF Library JAR File. For information about setting up the ViewController project, see "Setting Up the ViewController Project" on page 30-42. For information about adding a custom managed bean, see "Adding Custom Managed Bean" on page 30-45.

> **See Also:** *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework* for information about developing ADF taskflows

After developing the bounded taskflow, you must deploy it. For information about deploying the bounded taskflow, see "Deploying Custom Code to Oracle Identity Manager" on page 30-45.

Every new taskflow must be granted a permission to make it visible in the Identity Self Service. See "Securing a Custom Taskflow Using APM" on page 30-32 for details.

The final step in developing Home page portlets is updating the resource catalog for the Home page. This allows the users to add the portlet to the homepage.

To update the Home page resource catalog:

1. Locate the oracle.iam.ui.custom shared library, which is oracle.iam.ui.custom-dev-starter-pack.war. This is available in the *OIM_HOME*/server/apps/ directory.
2. Unpack the WAR and locate the oracle/adf/rc/metadata/custom-catalog.xml file.

3. Update the file and include reference to your newly added bounded taskflow. The entry looks similar to the following:

```
<resource id="helloWorld" name="Hello World"
description="Hello World Taskflow Reference"
repository="application.classpath"
path="adflibHomepagePortletsUI.jar/ADF_
TaskFlow/WEB-INF+oracle+iam+ui+sample+homepage+tfs+hello-world-tf.xml#hello-wor
ld-tf">
<attributes>
<attribute value="coreDefault" attributeId="attr.background" isKey="false"/>
</attributes>
</resource>
```

Note: Make sure that the format of the path property is correct. The format must follow the same pattern as shown in step 3. In addition, ensure that the value of the repository property is application.classpath. For more information about the format of this file, see "resource" of the "Catalog Definition Attributes" section in the *Oracle Fusion Middleware Developer's Guide for Oracle WebCenter Portal*.

4. Redeploy the shared library.

Note: Verify the following if you have performed the steps to update the Home page resource catalog, but the custom task flow is not displayed in the resource catalog after clicking the **Add Content** button:

- Ensure that the task flow permission is defined properly. See "Securing a Custom Taskflow Using APM" on page 30-32 for details.
- Ensure that the ADF library JAR file containing your task flow contains the META-INF/task-flow-registry.xml file and that the file contains a reference to your custom task flow. If the file is missing, then rebuild just the particular JDeveloper project containing your custom task flow, and check if it resolves the issue. To rebuild just the particular JDeveloper project, go to JDeveloper, right-click the ViewController project containing the custom task flow, select **Deployment**, and then select the deployment profile. The deployment profile must be of type ADF Library JAR File.

30.10.5.9 Launching Taskflows

You can launch a taskflow in the Self Service interface. For example, if you want to launch a tab with a bounded taskflow running in it, then perform the following steps:

1. Develop a custom managed bean with the following method:

```
public void launchMyTaskFlow(ActionEvent evt){

User user =
OIMClientFactory.getAuthenticatedSelfService().getProfileDetails(null);
       String taskFlowId =
"/WEB-INF/oracle/iam/ui/taskflows/public/tfs/user-details-tf.xml#user-details-t
```

```
f";
       // This id uniquely identifies the taskflow after launch. Add a suffix,
for example entityPrimaryKey, to make it unique.
       String id = "user-detail-tf";
       String name = user.getDisplayName() ;  // this is shown as the tab title
       String description = ""; // Add any suitable description
       String icon = "/images/user.png";
       String  helpTopicId = ConstantsDefinition.DEFAULT_HELP_TOPIC_ID; // Or
your custom OHW integrated help topic id
       boolean inDialog = false;
       Map params = new HashMap();  // These are your taskflow's input
parameters being passed from this launcher method
       params.put("userLogin",  user.getLogin());

       String jsonPayLoad = TaskFlowUtils.createContextualEventPayLoad(id,
taskFlowId, name, icon, description, helpTopicId, inDialog, params);
       TaskFlowUtils.raiseContextualEvent(TaskFlowUtils.RAISE_TASK_FLOW_LAUNCH_
EVENT, jsonPayLoad);
 }
```

Note: The above code snippet uses the user details public taskflow to display the user details when the user login is provided. For a list of available public taskflows that you can use for customization of UI, see "Using Public Taskflows" on page 30-66.

Package and deploy the managed bean, as described in "Developing Managed Beans and Task Flows" on page 30-42.

2. Using sandbox and Web Composer customization, add an ADF CommandLink to the correct page (JSFF file), and then export the sandbox. Open the sandbox zip, and edit the jsff.xml to bind actionListener for that link to the managed bean method.

3. Ensure that the page definition of the jsff has the raiseTaskFlowLaunchEvent binding. To find the name of the page definition file, you first need to know the name of the jsff page on which you have the launch link.

   If the jsff page is the existing left navigation page, then the name is left-nav.jsff. The page definition file for this jsff page is left-nav_pageDef.xml. This page definition XML file already contains the eventBinding, and therefore, nothing is required to be changed.

   If your launch link is on a custom jsff page, for example, your page name is my-custom.jsff, then look for a file named my-custom_pageDef.xml within the same JDev project. JDev automatically creates this file for each jsff. You must add the following eventBinding into this pageDef xml file:

```
<eventBinding id="raiseTaskFlowLaunchEvent">
      <events xmlns="http://xmlns.oracle.com/adfm/contextualEvent">
        <event name="oracle.idm.shell.event.TaskFlowLaunchEvent"/>
      </events>
    </eventBinding>
```

Note: Existing Oracle Identity Manager pages already contain the eventBinding. You must define the eventBinding for JSFF pages that you build.

30.10.5.10 Creating an External Link

To add a link or button that redirects the user to a certain URL:

1. In your custom request bean, create the following actionListener that will be invoked when the user clicks a link or button:

```
public void actionListener(ActionEvent e) {
        FacesUtils.redirect("http://www.oracle.com");
    }
```

2. Add a new commandLink or commandButton to the page on which you want to display the link or button by using Web Composer. See "Adding a Link or Button" on page 30-26 for details.
3. Set the actionListener property of the link or button component that you added to point to the actionListener method.

30.10.6 Using Managed Beans to Populate Request Attributes

This section describes the following approaches for populating request attributes:

- Populating Request Attributes Using Managed Beans
- Populating Request Attributes by Using the Prepopulate Plug-in

30.10.6.1 Populating Request Attributes Using Managed Beans

This approach involves creating a managed bean that gets invoked when the user clicks a custom button. The managed bean must be deployed to the Oracle Identity Manager customization placeholder library, which is oracle.iam.ui.custom-dev-starter-pack.war. The button, referred to as the Prepopulate button, is part of the UI customization and must be manually added to the page by using Web Composer.

The managed bean code is responsible for fetching the information to be populated in the request form. It uses Oracle Identity Manager APIs to get the beneficiary information from the request and from the user management layer, and uses JSF/ADF APIs to update the request form UI components.

To populate request attributes by using managed beans and UI customization:

1. Create the JDev application workspace and project. See "Setting Up the ViewController Project" on page 30-42 for details.
2. Create a Java class. In this example, the complete class name is com.oracle.demo.iam.prepop.view.PrePopulateMBean. This class contains:
   - Two member variables that hold references to the UI components, the custom Prepopulate button and its parent container.
   - Accessor methods (get and set) for the variables member variables.
   - An action listener type method to be invoked when the user clicks the custom Prepopulate button.
   - A method that returns a boolean value determines when the custom Prepopulate button must be disabled

   The custom code for this example is:

```
public class PrePopulateMBean {

    private UIComponent rootPanel;
    private UIComponent prepopulateButton;
```

```
    public PrePopulateMBean() {
        super();
    }

    public void setRootPanel(UIComponent rootPanel) {
        this.rootPanel = rootPanel;
    }

    public UIComponent getRootPanel() {
        return rootPanel;
    }

    public void setPrepopulateButton(UIComponent prepopulateButton) {
        this.prepopulateButton = prepopulateButton;
    }

    public UIComponent getPrepopulateButton() {
        return prepopulateButton;
    }

    public boolean isPrepopulateButtonRendered() {

        boolean ret = false;
        RequestFormContext requestFormContext =
RequestFormContext.getCurrentInstance();
        if (requestFormContext != null) {

            boolean isActionRequest   = (requestFormContext.getActionType() ==
RequestFormContext.ActionType.REQUEST);
            boolean singleUserRequest = false;

            if (requestFormContext.getBeneficiaryIds()!=null) {
                singleUserRequest =
(requestFormContext.getBeneficiaryIds().size() == 1);
            }
            ret = isActionRequest && singleUserRequest;
        }
        return (ret);
    }

    public void actionListener(ActionEvent e) {

        if (e.getSource().equals(prepopulateButton)) {

            RequestFormContext requestFormContext =
RequestFormContext.getCurrentInstance();
            List<String> beneficiaryIds =
requestFormContext.getBeneficiaryIds();

            if (beneficiaryIds.size() == 1) {

                try {
                    User user = getUser(beneficiaryIds.get(0));
                    FacesUtils.setAttributeBindingValue("UD_OID_USR_FNAME__c",
user.getFirstName());
                    FacesUtils.setAttributeBindingValue("UD_OID_USR_LNAME__c",
user.getLastName());

                } catch (NoSuchUserException f) {
```

```
                    f.printStackTrace();
                } catch (UserLookupException f) {
                    f.printStackTrace();
                }
            }
        }
        FacesUtils.partialRender(rootPanel);
    }

    private User getUser(String userKey) throws NoSuchUserException,
UserLookupException {

        UserManager userMgr = OIMClientFactory.getUserManager();

        HashSet<String> searchAttrs = new java.util.HashSet<String>();
        searchAttrs.add(AttributeName.USER_LOGIN.getId());
        searchAttrs.add(AttributeName.LASTNAME.getId());
        searchAttrs.add(AttributeName.FIRSTNAME.getId());

        return userMgr.getDetails(userKey,searchAttrs, false);
    }
}
```

In the code for the Java class:

- The `isPrepopulateButtonRendered` method returns `true` if a RequestContext is available, and if there is only one request beneficiary. The check on the RequestContext availability is required to avoid issues at the time of customization. This method is invoked when the custom Prepopulate button is loaded, or its container is refreshed.
- The `actionListener` method executes a user search in Oracle Identity Manager by invoking the `getUser` method, which uses the request beneficiary information. Then, it directly sets values on the UD_OID_USR_FNAME__c and UD_OID_USR_LNAME__c UI components with the information returned from the user search, and invokes a partial rendering on the rootPanel. This is the panel that holds the custom button and the request form. The partial rendering will display the values in the respective fields. It is important to mention here that this custom code contains a direct reference to the UI components, and that these direct references can be found by exporting the sandbox. This method is invoked when the custom Prepopulate button is loaded or its container refreshed.
- The FacesUtil class is responsible for rendering the UI changes. See Appendix B, "The FacesUtils Class" for the code for this class.

3. Declare the PrePopulateMBean class as a managed bean in the JDev project. This makes the MBean available in the UI so that it can be invoked by using EL expressions. To configure this, specify the following values in the Managed Beans section of the View Controller project:
   - **Name:** prepopMBean
   - **Class:** com.oracle.demo.iam.prepop.view.PrePopulateMBean
   - **Scope:** backingBean
4. Deploy the View Controller project as an ADF library JAR file. This type of deployment can be created in JDeveloper through the deployment profiles option. The deployment generates a JAR file. Copy this file into oracle.iam.ui.custom-dev-starter-pack.war, which is Oracle Identity Manager

placeholder library. This file is available along with the other Oracle Identity Manager application packages, such as EAR and WAR files, at the $*OIM_ORACLE_HOME*/server/apps/ directory. Create a backup of this file before modifying it.

5. To deploy the custom code:
   a. Copy the oracle.iam.ui.custom-dev-starter-pack.war to a temporary location.
   b. Open the oracle.iam.ui.custom-dev-starter-pack.war.
   c. Add the custom jar file to the WEB-INF/lib directory. If the lib directory does not exist, then create it.
   d. Save the oracle.iam.ui.custom-dev-starter-pack.war file.
   e. Copy the oracle.iam.ui.custom-dev-starter-pack.war file back to its original location in the $*OIM_ORACLE_HOME*/server/apps/ directory.
   f. Stop Oracle Identity Manager Managed Server.
   g. In WebLogic Administration Console, update the oracle.iam.ui.custom library deployment, and activate the changes.
   h. Start Oracle Identity Manager Managed Server.
6. Customize the UI. To do so:
   a. Create and activate a sandbox. In this example, the sandbox name is RequestPrePop.
   b. Login to Oracle Identity Self Service, and navigate to the Catalog page.
   c. Search for the specific application instance to be customized. In this example, the application instance is called Local OID. Add the application instance to the shopping cart, and **Checkout**.
   d. Click **Customize**.
   e. Select **View**, **Source**.
   f. In the Cart Items and Details sections of the page, click close to the Details label. Make sure that the showDetailHeader:Details component is selected.
   g. On the top-left, click **Edit**. In the dialog box that opens, edit the Binding property, and configure the following EL using the Expression Builder:

      ```
      #{backingBeanScope.prepopMBean.rootPanel}
      ```

      This expression bind will make the UI invoke the setRootPanel method in the custom managed bean. Click **OK**.
   h. Make sure that the showDetailHeader:Details component selected. Click **Add Content**.
   i. Scroll down, and open the Web Components section.
   j. Click **Add** on the right of the Command Toolbar Button component. A button is added on the Details section.
   k. Click the button, and then click **Edit**.
   l. Edit the Text property, and set PrePopulate as the label.
   m. Edit the Binding property and configure the following EL using the Expression Builder:

      ```
      #{backingBeanScope.prepopMBean.prepopulateButton}
      ```

This bind is for invoking the setPprepopulateButton method in the custom managed bean. Click **OK**.

n. Edit the Disabled property, and configure the following EL by using the Expression Builder:

```
#{!backingBeanScope.prepopMBean.prepopulateButtonRendered}
```

This is to invoke the isPrepopulateButtonRendered method in the managed bean. Click **Ok**.

o. Click the **Style** tab. Set the Width property to 100, and the Margin - Left property to 100. Click **OK**. This configuration will properly place the PrePopulate button in the UI.

p. Exit the customization mode by clicking **Close**.

7. To manually configure the properties of the Prepopulate button:

a. Navigate to the Sandbox page. De-activate and export the sandbox.

b. Save the sandbox ZIP file in the local file system.

c. Extract the ZIP file. In a text editor, open the XML file corresponding to the customization. In this example, the file is oracle/iam/ui/runtime/form/view/pages/mdssys/cust/site/site/OIDUserFormCreateForm.jsff.xml.

d. Search for the section defining the custom Prepopulate button, which can be similar to the following:

```
<af:commandToolbarButton xmlns:af="http://xmlns.oracle.com/adf/faces/rich"
id="e8829502064"
binding="#{backingBeanScope.prepopMBean.prepopulateButton}"
text="PrePopulate"
```

e. Add the actionListener property to the custom Prepopulate button, as shown:

```
actionListener="#{backingBeanScope.prepopMBean.actionListener}
```

f. Save the file and repackage the ZIP. Make sure that the path is preserved when repacking the content.

g. Import the sandbox, and import the ZIP file. Make sure that the sandbox is not active when importing it.

h. Activate the sandbox.

8. Test the customization. To do so:

a. Navigate to the Catalog, find the application instance and add it do the shopping cart.

b. In the cart summary page, the custom Prepopulate button is displayed.when clicking on it, the First Name and Last Name fields will be updated with the beneficiary's information

c. Click the Prepopulate button. The First Name and Last Name fields are updated with the beneficiary's information.

9. Publish the sandbox.

30.10.6.2 Populating Request Attributes by Using the Prepopulate Plug-in

Prepopulate plug-ins can be used when the same logic is to be executed for both UI and API request creation, and can also be used when a UI interaction is not required. In this approach, a plug-in is present for each attribute that must be prepopulated in the request. The same plug-in can be used across different resources and different attributes.

The plug-in code implements the oracle.iam.request.plugins.PrePopulationAdapter interface. The following is an example code:

```
package com.oracle.demo.iam.prepop.plugin;

import java.io.Serializable;

import java.util.HashSet;
import java.util.List;

import oracle.iam.identity.usermgmt.api.UserManager;
import oracle.iam.identity.usermgmt.api.UserManagerConstants.AttributeName;
import oracle.iam.identity.usermgmt.vo.User;
import oracle.iam.platform.Platform;
import oracle.iam.request.vo.Beneficiary;
import oracle.iam.request.vo.RequestData;

public class UserLoginPrePop implements
oracle.iam.request.plugins.PrePopulationAdapter {

    public UserLoginPrePop() {
        super();
    }

    public Serializable prepopulate(RequestData requestData) {

        String prePopUserId = null;

        List<Beneficiary> benList = requestData.getBeneficiaries();

        if(benList.size()==1){

            UserManager  usersvc = Platform.getService(UserManager.class);

            for(Beneficiary benf: benList){

                HashSet<string> searchAttrs = new java.util.HashSet<String>();
                searchAttrs.add(AttributeName.USER_LOGIN.getId());

                try {
                    User userBenef =
usersvc.getDetails(benf.getBeneficiaryKey(),searchAttrs, false);
                    if (userBenef!= null) {
                        prePopUserId = userBenef.getLogin();
                    }
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        }
        return prePopUserId;
    }
}
```

A prepopulate plug-in is similar to any other plug-in in Oracle Identity Manager. The plug-in class is compiled and deployed to a JAR file. The JAR file must be added to a ZIP file in the lib directory. The ZIP file must contain in the root path a XML file declaring the plug-in. The XML used in this example is as follows:

```
<?xml version="1.0" encoding="UTF-8" ?>
<oimplugins xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <plugins pluginpoint="oracle.iam.request.plugins.PrePopulationAdapter">
 <plugin pluginclass= "com.oracle.demo.iam.prepop.plugin.UserLoginPrePop"
version="1.0" name="UserLoginPrePop">
  <metadata name="PrePopulationAdapater">
   <value>OracleDBUMForm::Username|OIDUserForm::User ID</value>
  </metadata>
 </plugin>
</plugins>
</oimplugins>
```

In the XML code:

- The xmlns tag attribute must be present in the XML. Otherwise, the plug-in is not invoked by Oracle Identity Manager.
- The value in the pluginpoint element must be oracle.iam.request.plugins.PrePopulationAdapter.
- The metadata tag contains a value child node. This value child node must contain the pairs of FormName::AttributeName. Each pair indicates a form attribute that will be populated by the prepopulate plug-in. In this example, such attributes are Username in the OracleDBUMForm form and User ID in the OIDUserForm form. The form names are configured when the ApplicationInstances and their forms were created, and not the process form created when the connector is imported into Oracle Identity Manager.

The prepopulate plug-in can be deployed to the $*OIM_HOME*/server/plugins/ directory, or it can be registered using the plug-in registration script. In production environments, it is always recommended to deploy the plug-in by using the command line so that the plug-in Zip file is uploaded to the database.

30.10.7 Using Public Taskflows

Oracle Identity Manager provides default taskflows for using them in the customized pages of Oracle Identity Self Service and to invoke other taskflows. For example, you can customize the user details page so that the user details of the manager will be displayed if you click the manager login name in the user details page.

The default or predefined taskflows are called public taskflows. While launching the public taskflows, you must provide appropriate values for some parameters. For example, to launch the request details page for a particular request, you must provide the request ID for the request.

Table 30–7 lists the public taskflows provided by Oracle Identity Manager along with the input parameters that are required to invoke the taskflows.

Table 30–7 Public Taskflows

Taskflow name	Taskflow path	Description	Parameter	Mandatory
Request Details	/WEB-INF/oracle/iam/ui/taskflows/public/tfs/request-details-tf.xml#request-details-tf	This is launched to view the details of a request that is submitted for approval.	**requestID:** The ID of the request whose details is to be displayed.	Yes
User Details	/WEB-INF/oracle/iam/ui/taskflows/public/tfs/user-details-tf.xml#user-details-tf	This is launched to view the details of a user.	**userLogin:** User Login attribute value of the user whose details is to be displayed.	Yes
Role Details	/WEB-INF/oracle/iam/ui/taskflows/public/tfs/role-details-tf.xml#role-details-tf	This is launched to view the details of a role.	**roleName:** Name of the role whose details is to be displayed.	Yes
Request Role	/WEB-INF/oracle/iam/ui/taskflows/public/tfs/request-role-tf.xml#request-role-tf	This is launched to request for assignment of role(s) for beneficiaries.	**roleNames:** Names of the role(s) that are to be assinged. The names must be separated by commas.	Yes
			userLogins: User Login attribute values of the user(s) or beneficaries for whom the roles are to be assigned. The values must be separated by commas. If a value is not provided, then the request action is applicable for the currently logged-in user.	No
Revoke Role	/WEB-INF/oracle/iam/ui/taskflows/public/tfs/revoke-role-tf.xml#revoke-role-tf	This is launched to request for revoking of role(s) that are assigned to beneficiaries.	**roleNames:** Names of the role(s) that are to be revoked.	Yes
			userLogins: User Login attribute values of the user(s) or beneficiaries for whom the roles are to be revoked. The values must be separated by commas. If a value is not provided, then the revoke action is applicable for the currently logged-in user.	No

Table 30–7 (Cont.) Public Taskflows

Taskflow name	Taskflow path	Description	Parameter	Mandatory
Request Account	/WEB-INF/oracle/iam/ui/taskflows/public/tfs/request-account-tf.xml#request-account-tf	This is launched to request for creation of account(s) for the beneficiaries.	**appInstNames:** Names of the application instance(s) where accounts are to be created. The values must be separated by commas.	Yes
			userLogins: User Login attribute values of the user(s) or beneficiaries for whom the accounts are to be assigned. The values must be separated by commas. If a value is not provided, then the request action is applicable for the currently logged-in user.	No
Modify Account	/WEB-INF/oracle/iam/ui/taskflows/public/tfs/modify-account-tf.xml#modify-account-tf	This is launched to modify the account details created for a user or beneficiary.	**accountNames:** Name of the accounts whose details are to be modified. The values must be separated by commas.	Yes
			userLogins: User Login attribute values of the users or beneficiaries whose account details are to be modified. The values must be separated by commas. If a value is not provided, then the revoke action is applicable for the currently logged-in user.	No
Enable Account	/WEB-INF/oracle/iam/ui/taskflows/public/tfs/enable-account-tf.xml#enable-account-tf	This is launched to enable accounts assigned to user(s) or beneficiaries.	**accountNames:** Names of the accounts that are to be enabled. The values must be separated by commas.	Yes

Table 30–7 (Cont.) Public Taskflows

Taskflow name	Taskflow path	Description	Parameter	Mandatory
			userLogins: User Login attribute values of the user(s) or beneficiaries whose accounts are to be enabled. The values must be separated by commas. If a value is not provided, then the request action is applicable for the currently logged-in user.	No
Disable Account	/WEB-INF/oracle/iam/ui/taskflows/public/tfs/disable-account-tf.xml#disable-account-tf	This is launched to disable accounts assigned to user(s) or beneficiaries.	**accountNames:** Names of the accounts that are to be disabled. The values must be separated by commas.	Yes
			userLogins: User Login attribute values of the user(s) or beneficiaries whose accounts are to be disabled. The values must be separated by commas. If a value is not provided, then the request action is applicable for the currently logged-in user.	No
Delete Account	/WEB-INF/oracle/iam/ui/taskflows/public/tfs/delete-account-tf.xml#delete-account-tf	This is launched to delete accounts assigned to user(s) or beneficiaries.	**accountNames:** Names of the accounts that are to be deleted. The values must be separated by commas.	Yes
			userLogins: User Login attribute values of the user(s) or beneficiaries whose accounts are to be deleted. The values must be separated by commas. If a value is not provided, then the request action is applicable for the currently logged-in user.	No
Request Entitlement	/WEB-INF/oracle/iam/ui/taskflows/public/tfs/request-entitlement-tf.xml#request-entitlement-tf	This is launched to request for the assignment of entilement(s) for beneficiaries.	**entlmntNames:** Names of the entilement(s) that are to be assigned. The values must be separated by commas.	Yes

Table 30–7 (Cont.) Public Taskflows

Taskflow name	Taskflow path	Description	Parameter	Mandatory
			userLogins: User Login attribute values of the user(s) or beneficiaries to whom the entitlements are to be assigned. The values must be separated by commas. If a value is not provided, then the request action is applicable for the currently logged-in user.	No
Revoke Entitlement	/WEB-INF/oracle/iam/ui/taskflows/public/tfs/revoke-entitlement-tf.xml#revoke-entitlement-tf	This is launched to request for revoking of entilement(s) assigned to beneficiaries.	**entlmntNames:** Names of the entilement(s) that are to be revoked. The values must be separated by commas.	Yes
			userLogins: User Login attribute values of the user(s) or beneficiaries from whom the entitlements are to be revoked. The values must be separated by commas. If a value is not provided, then the request action is applicable for the currently logged-in user.	No
Create User	/WEB-INF/oracle/iam/ui/taskflows/public/tfs/create-user-tf.xml#create-user-tf	This is launched to create an user entity.	No parameters are required.	No
Modify User	/WEB-INF/oracle/iam/ui/taskflows/public/tfs/modify-user-tf.xml#modify-user-tf	This is launched to modify the user details.	**userLogins:** User Login attribute values of the user(s) whose details are to be modified. If more than one userLogin attribute is provided as parameter, then bulk modify page is displayed. The values must be separated by commas.	Yes
Enable User	/WEB-INF/oracle/iam/ui/taskflows/public/tfs/enable-user-tf.xml#enable-user-tf	This is launched to enable the disabled user(s).	**userLogins:** The User Login attribute values of the users that are to be enabled. The values must be separated by commas.	Yes

Table 30–7 (Cont.) Public Taskflows

Taskflow name	Taskflow path	Description	Parameter	Mandatory
Disable User	/WEB-INF/oracle/iam/ui/taskflows/public/tfs/disable-user-tf.xml#disable-user-tf	This is launched to disable the enabled user(s).	**userLogins:** The User Login attribute values of the users that are to be disabled. The values must be separated by commas.	Yes
Delete User	/WEB-INF/oracle/iam/ui/taskflows/public/tfs/modify-user-tf.xml#modify-user-tf	This is launched to delete user(s).	**userLogins:** The User Login attribute values of the users that are to be deleted. The values must be separated by commas.	Yes

Table 30–7 (Cont.) Public Taskflows

Taskflow name	Taskflow path	Description	Parameter	Mandatory
Catalog Search	/WEB-INF/oracle/iam/ui/taskflows/public/tfs/catalog-search-tf.xml#catalog-search-tf	This is launched to specify the catalog search criteria and display the search results page or cart details page directly without using the catalog search page.	**searchCrtieria:** Various string attributes in the following format: `{criteriaName: "string", allowSearch: "true/false", profileName: "string", directCheckout: "true/false", showEntityTypeSelector: "true/false", hiddenTag: "string", allowedEntityTypes: "string", tags: "string", entityType: "string", auditObjective: "string", riskLevel: "string", ANY_UDF: "string"}` Here: ■ **criteriaName:** Optional string attribute that will be displayed in the catalog results page. ■ **allowSearch:** Optional boolean attribute to control rendering of tag search field in results page. ■ **profileName:** Optional string attribute to take user to cart page by simulating the saved profile click. ■ **directCheckout:** Optional parameter to add search results to the cart and take user to checkout page (true/false). ■ **showEntityTypeSelector**Optional boolean attribute to show entityTypeSelector dropdown. This is displayed only if allowSearch is also set to true.	Yes

Table 30–7 (Cont.) Public Taskflows

Taskflow name	Taskflow path	Description	Parameter	Mandatory
			userLogins: The User Login attribute values of the users to be displayed in the beneficiary table in the catalog search results page. The values must be separated by commas. If value is not passed, then the current logged-in user is shown in the beneficiary table.	
Catalog Item Details	/WEB-INF/oracle/iam/ui/taskflows/public/tfs/catalog-item-details-tf.xml#catalog-item-details-tf	This is launched to display the details of a catalog item.	**catalogItemName:** Name of the catalog item whose details are to be displayed.	Yes
			catalogItemType: Type of the catalog item whose details are to be displayed. The valid values are Role, ApplicationIstance, or Entitlement.	Yes
User Roles	/WEB-INF/oracle/iam/ui/taskflows/public/tfs/user-roles-tf.xml#user-roles-tf	This is launched to view the roles page of a given user.	**userLogin:** User Login attribute value of the user whose roles page is to be displayed.	Yes
User Accounts	/WEB-INF/oracle/iam/ui/taskflows/public/tfs/user-accounts-tf.xml#user-accounts-tf	This is launched to view the accounts page of a given user.	**userLogin:** User Login attribute value of the user whose accounts page is to be displayed.	Yes
User Entitlements	/WEB-INF/oracle/iam/ui/taskflows/public/tfs/user-entitlements-tf.xml#user-entitlements-tf	This is launched to view the entitlements page of a given user.	**userLogin:** User Login attribute value of the user whose entitlements page is to be displayed.	Yes
User Assigned Admin Roles	/WEB-INF/oracle/iam/ui/taskflows/public/tfs/user-assigned-adminroles-tf.xml#user-assigned-adminroles-tf	This is launched to view the assigned admin roles page of a given user.	**userLogin:** User Login attribute value of the user whose assigned admin roles page is to be displayed.	Yes
Organization Details	/WEB-INF/oracle/iam/ui/taskflows/public/tfs/org-details-tf.xml#org-details-tf	This is launched to view the organization details page.	**orgName:** Name of the organization whose details page is to be displayed.	Yes
My Access	/WEB-INF/oracle/iam/ui/taskflows/public/tfs/my-access-tf.xml#my-access-tf	This is launched to display the access page of the currently logged-in user.	No parameters are required.	No
Change User Account Password	/WEB-INF/oracle/iam/ui/taskflows/public/tfs/account-passwd-reset-tf.xml#account-passwd-reset-tf	This is launched to display the change user account password page for a given user.	**accountName:** Name of the user's account whose password is to be changed.	Yes

Table 30–7 (Cont.) Public Taskflows

Taskflow name	Taskflow path	Description	Parameter	Mandatory
			userLogin: User Login attribute value of the user whose account password is to be changed.	Yes
Account Details	/WEB-INF/oracle/iam/ui/taskflows/public/tfs/account-details-tf.xml#account-details-tf	This is launched to display the details of a user's account.	**accountName:** Name of the user's account whose details is to be displayed.	Yes
			userLogin: User Login attribute value of the user whose account is to be displayed.	Yes

Note:

- The parameters of all the public taskflows listed in Table 30–7 are of type java.lang.String.
- The public taskflows can be launched by using contextual event as described in "Launching Taskflows" on page 30-58. Otherwise, public taskflows can be embedded in an ADF faces page. To embed public taskflows in an ADF faces page, the following parameter (in addition to the parameters listed in Table 30–7) must be added to the taskflow definition in the page definition file of the ADF faces page:

  Parameter name: "uiShell"

  value: "#{pageFlowScope.uiShell}"

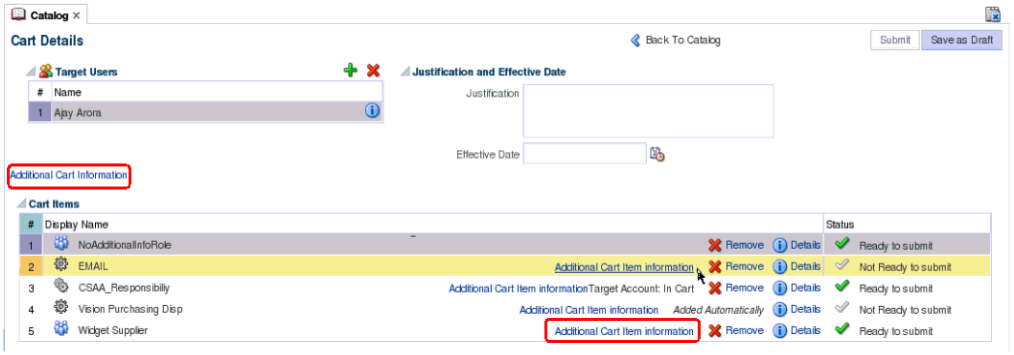
30.11 Customizing the UI for Providing Additional Request Information

Users can enter additional information for a request that can be useful for the request approvers. The following topics are discussed in this section:

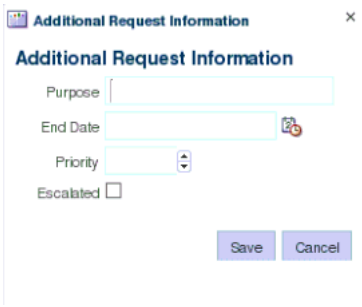
- Additional Request Information Concepts
- AdditionalRequestInfo Interface
- Implementing Custom Taskflow for Additional Request Information
- Launching Custom Taskflow for Additional Request Information
- Customizing the UI at Cart (Request) Level
- Customizing the UI at Cart Item Level

30.11.1 Additional Request Information Concepts

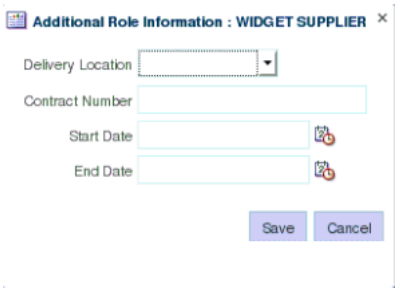
You can customize the UI on Cart Details page to enable users to provide additional information when they are raising a request. You can add a link or a button at the cart or request level and at the cart item level, as shown in the following sample screenshot.



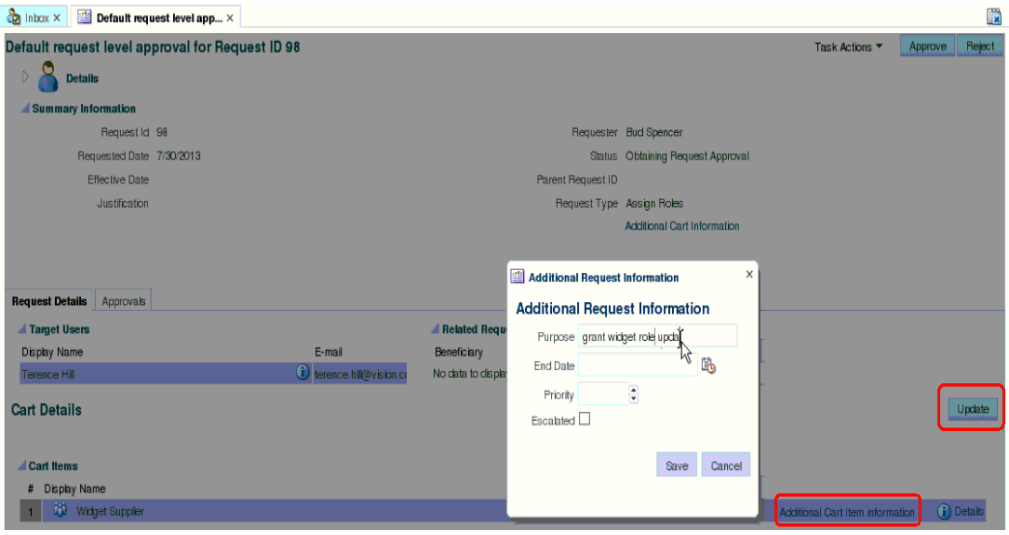
When a user clicks a link, a popup window is displayed with custom fields. The user can enter appropriate information useful for the request approvers. The following sample screenshot shows Additional Request Information popup window for the cart/request:



The following sample screenshot shows Additional Request Information popup window for the individual cart item in the request:



The request approver can view and/or update additional request information provided at the cart (request) level and/or at the cart item level. The request approver can access the request from Identity Self Service on the Inbox page.



30.11.2 AdditionalRequestInfo Interface

AdditionalRequestInfo interface includes `setAttribute` and `getAttribute` methods. The setAttribute methods allow a custom taskflow to set additional request-only attributes that can be provided as part of the request. The `getAttribute` methods allow a custom taskflow to get the additional request attributes during tasks such as request approval, view, summary, and so on.

AdditionalRequestInfo interface supports setting and getting attributes at the cart/request level and at the cart item level. For cart item level attributes, the cartItemId parameter must be specified. You can use the `getUniqueCartItemIdentifier` EL expression in the RequestFormContext to access the selected cartItemId. For more information, see "Available EL Expressions in the RequestFormContext" on page 30-22.

AdditionalRequestInfo interface extends Serializable as shown in the following code. Therefore, an instance of AdditionalRequestInfo can be stored in pageFlowScope of the custom taskflow.

```
public interface AdditionalRequestInfo extends Serializable {
    public void setAttribute(String name, Serializable value);
    public void setAttribute(String cartItemId, String name, Serializable value);
    public Serializable getAttribute(String name);
    public Serializable getAttribute(String cartItemId, String name);
}
```

30.11.3 Implementing Custom Taskflow for Additional Request Information

To implement a custom taskflow that can be used for providing and viewing additional request-only information:

1. Develop the custom taskflow as a bounded taskflow in ViewController projects. For information about setting up the ViewController project, see "Setting Up the ViewController Project" on page 30-42.
2. Define taskflow input parameters as required.

   The following table lists pre-defined input parameters that, if defined, pass appropriate values automatically when the taskflow is launched:

Parameter Name	Type	Description
additionalRequestInfo	oracle.iam.ui.common.model.catalog.requestdetails.AdditionalRequestInfo	An instance of AdditionalRequestInfo interface. It is used to set and get values of additional request-level and cart item level request attributes. For more information about the interface methods, see "AdditionalRequestInfo Interface" on page 30-76.
requestFormContext	oracle.iam.ui.platform.view.RequestFormContext	An instance of RequestFormContext. RequestFormContext provides various context information related to the request and currently selected cart item. For example, you can leverage the information provided by RequestFormContext to decide whether the input components of the custom taskflow must be rendered in read-only or editable mode. For more information, see "Available EL Expressions in the RequestFormContext" on page 30-22.

In addition to the pre-defined parameters, you can define other input parameters to be passed to the taskflow. These parameters are not known to Oracle Identity Manager. Therefore, the values to be passed to the taskflow are defined using clientAttribute of the link or the button that launches the custom taskflow.

Information about launching the taskflow and passing parameters to the taskflow is discussed in "Launching Custom Taskflow for Additional Request Information" on page 30-77.

3. Deploy the taskflow as part of the `oracle.iam.ui.custom` shared library. For information about deploying the bounded taskflow, see "Deploying Custom Code to Oracle Identity Manager" on page 30-45.
4. Add permissions to the custom taskflow by using the Authorization Policy Manager (APM) UI to secure the taskflow, as described in "Securing a Custom Taskflow Using APM" on page 30-32.
5. Grant permission to make it visible in the Identity Self Service. See "Securing a Task Flow Region Using EL Expressions" on page 30-33 for details.

30.11.4 Launching Custom Taskflow for Additional Request Information

You can enable users to provide additional information specific to a cart item, such as Application Instance, Role, or Entitlement. To do so, you can add a link or a button to Cart Items table using Oracle Web Composer. Cart Items table is displayed in the Cart Checkout, Request Summary, Request view, and Request Approval pages whenever the UI operation involves cart items.

A set of attributes defined as descendants of the component determine which custom taskflow to be launched and how the popup window is displayed. The following table summarizes the pre-defined attributes:

Name	Type	Required	Description
taskFlowId	java.lang.String	Yes	ID of the custom taskflow to be launched by the component.

Name	Type	Required	Description
dialogTitle	java.lang.String	Optional	Title of the popup window in which the custom taskflow is launched. If no value is specified, then the title is blank.
dialogTitleIcon	java.lang.String	Optional	Path to the icon that is displayed in the popup window in which the custom taskflow is launched. If no value is specified, then the default icon is used.

For example, the following code snippet creates the **Additional Cart Item Information** link:

```
<af:commandLink xmlns:af="http://xmlns.oracle.com/adf/faces/rich" id="e655321996"
text="Additional Cart Item Information"
actionListener="#{catalogRequestBean.launchAdditionalRequestInfoTaskFlow}">
         <af:clientAttribute xmlns:af="http://xmlns.oracle.com/adf/faces/rich"
name="taskFlowId"
value="/WEB-INF/oracle/iam/ui/sample/catalog/tfs/additional-cart-item-info-tf.xml#
additional-cart-item-info-tf"/>
         <af:clientAttribute xmlns:af="http://xmlns.oracle.com/adf/faces/rich"
name="dialogTitleIcon" value="/images/request_ena.png"/>
         <af:clientAttribute xmlns:af="http://xmlns.oracle.com/adf/faces/rich"
name="headerText" value="Additional Cart Item Information"/>
</af:commandLink>
```

In this example code snippet:

- `/WEB-INF/oracle/iam/ui/sample/catalog/tfs/additional-cart-item-info-tf.xml#additional-cart-item-info-tf` is the taskflow launched in a popup window.
- `#{catalogRequestBean.launchAdditionalRequestInfoTaskFlow}` is the mandatory EL to be specified as a value of the component's **actionListener** property.

  The action listener launches the custom taskflow in a popup window.
- `/images/request_ena.png` is the path of the icon for the popup window.
- headerText is the taskflow input parameter.

Note: When the Cart Item form contains mandatory fields, the Additional Request Information taskflows cannot be launched without entering values in the mandatory fields. In such a case, if you click on an Additional Request Information link or a button, an ADF error pops up as a result of the mandatory field validation.

To provide additional request information without entering values for the mandatory fields, the validation needs to be disabled. This can be achieved by UI customization, by setting **immediate** property of the commandLink or commandButton component to `true`.

Alternatively, you can edit the customized **cart-details.jsff.xml** file to set the value of immediate property to true within Additional Cart Information component, as follows:

```
<af:commandLink xmlns:af="http://xmlns.oracle.com/adf/faces/rich"
text="Additional Cart Information" immediate="true"...>
```

30.11.5 Customizing the UI at Cart (Request) Level

When you enable the users to provide additional request information, you can customize the UI at the cart (request) level in the following ways:

- Selectively display a link or a button at the cart level.

  For example, the custom UI component (link or button) can be displayed only when a user is creating or approving a request, but not when the user is tracking a request.

- Launch different taskflows based on the context.

  For example, different custom taskflows can be launched when a user is requesting and revoking access.

- Customize the popup window, such as icon and title for the window.

The following code snippet shows a sample customization using appropriate EL expressions:

```
<af:commandLink xmlns:af="http://xmlns.oracle.com/adf/faces/rich" id="e8065932664"
text="Additional Cart Information"
actionListener="#{catalogRequestBean.launchAdditionalRequestInfoTaskFlow}"
rendered="#{backingBeanScope.additionalInfoHelperBean.renderAdditionalDetailsForRe
quest}">
         <af:clientAttribute xmlns:af="http://xmlns.oracle.com/adf/faces/rich"
name="taskFlowId"
value="#{backingBeanScope.additionalInfoHelperBean.additionalInfoTaskFlowIdForRequ
est}"/>
         <af:clientAttribute xmlns:af="http://xmlns.oracle.com/adf/faces/rich"
name="dialogTitleIcon" value="/images/request_ena.png"/>
         <af:clientAttribute xmlns:af="http://xmlns.oracle.com/adf/faces/rich"
name="headerText" value="Additional Cart Information"/>
         <af:clientAttribute xmlns:af="http://xmlns.oracle.com/adf/faces/rich"
name="dialogTitle"
value="#{backingBeanScope.additionalInfoHelperBean.popupTitle}"/>
      </af:commandLink>
```

In this sample code snippet, a custom Additional Cart Information link is added to cart details page. Here:

- The custom link is selectively displayed for the rendered property using the EL expression, `#{backingBeanScope.additionalInfoHelperBean.renderAdditionalDetailsForRequest}`.

  To use this EL expression, develop a managed JAVA bean, say additionalRequestInfoHelperBean, with the isRenderAdditionalDetailsForRequest method. This method should return a boolean value (true or false) based on whether the link has to be displayed or not displayed. For more information about the managed bean, see "Developing Managed Beans and Task Flows" on page 30-42.

- Launch different taskflows using the EL expression, `#{backingBeanScope.additionalInfoHelperBean.additionalInfoTaskFlowIdForRequest}`, specified as the value of the taskFlowId attribute.

  To use this EL expression, include the getAdditionalInfoTaskFlowIdForRequest method in the managed bean named additionalRequestInfoHelperBean. This method should return a String value representing the ID of the custom taskflow deployed as part of oracle.iam.ui.custom-dev-starter-pack.war. For example, `/WEB-INF/oracle/iam/ui/sample/catalog/tfs/additional-cart-info-tf.xml#additional-cart-info-tf`. For information about the taskFlowId attribute, see "Launching Custom Taskflow for Additional Request Information" on page 30-77.

- Set different popup window titles using the EL expression, `#{backingBeanScope.additionalInfoHelperBean.popupTitle}`, specified as the value of the dialogTitle attribute.

  To use this EL expression, include the getPopupTitle method in managed bean named additionalRequestInfoHelperBean. This method should return the desired String value to be displayed as popup window title. For information about the dialogTitle attribute, see "Launching Custom Taskflow for Additional Request Information" on page 30-77.

You can use the following available data to achieve the required customization:

- Data available in the pageFlowScope map by one of the following ways:
  - By evaluating the EL expression, `#{pageFlowScope}`
  - By invoking the AdfFacesContext.getCurrentInstance().getPageFlowScope() method

    This method returns an instance of java.util.Map, which contains data as key-value pairs. A few useful values (Map Key values) stored in this map are:

    - `requestId`: The value corresponding to this key would be a Request ID, which varies depending on the UI action performed by the user.

      For example, when the user is requesting for or modifying access, or performing any other operation supported by request, the value will be -1. For all other UI actions, such as tracking or approving request, the value will be a valid Request ID.

      You can use this value to distinguish request tracking or approval flows from request creation.

    - `requestAction`: The value for this key (if *not* null) corresponds to the UI action performed by the user, such as APPROVAL_UPDATE, APPROVAL_VIEW, ACCOUNT_UPDATE, and ACCOUNT_VIEW.

You can use this value to distinguish request tracking flow from request approval flow. APPROVAL_VIEW corresponds to request tracking flow and APPROVAL_UPDATE corresponds to request approval flow.

  - `entityType`: The value for this key (if *not* null) corresponds to the Oracle Identity Manager entity type involved in the operation, such as role and entitlement.

- Data obtained from RequestFormContext, which holds information about the cart, by invoking the oracle.iam.ui.platform.view.RequestFormContext.newInstance(null).getCurrentContext() method

  This method returns an object of type oracle.iam.ui.platform.view.RequestFormContext. For more information about RequestFormContext, see "Available EL Expressions in the RequestFormContext" on page 30-22.

30.11.6 Customizing the UI at Cart Item Level

When you enable the users to provide additional request information, you can customize the UI at the cart item level in the following ways:

- Selectively display a link or a button at the cart level.

  For example, when granting VPN Administrator and IT Administrator roles to a user, a delegated administrator may be able to provide additional information for VPN Administrator role, but not for the IT Administrator role.

- Launch different taskflows based on the context.

  For example, different custom taskflows or forms can be launched for widget supplier and IT Administrators roles.

- Customize the popup window, such as icon and title for the window.

The following code snippet shows a sample customization using appropriate EL expressions:

```
<af:commandLink xmlns:af="http://xmlns.oracle.com/adf/faces/rich" id="e3664710724"
text="Additional Cart Item information"
actionListener="#{catalogRequestBean.launchAdditionalRequestInfoTaskFlow}"
rendered="#{backingBeanScope.additionalInfoHelperBean.renderAdditionalDetailsForCa
rtItem}">
         <af:clientAttribute xmlns:af="http://xmlns.oracle.com/adf/faces/rich"
name="taskFlowId"
value="#{backingBeanScope.additionalInfoHelperBean.additionalInfoTaskFlowIdForCart
Item}"/>
         <af:clientAttribute xmlns:af="http://xmlns.oracle.com/adf/faces/rich"
name="dialogTitleIcon" value="/images/request_ena.png"/>
         <af:clientAttribute xmlns:af="http://xmlns.oracle.com/adf/faces/rich"
name="headerText" value="Additional Cart item Information"/>
         <af:clientAttribute xmlns:af="http://xmlns.oracle.com/adf/faces/rich"
name="dialogTitle"
value="#{backingBeanScope.additionalInfoHelperBean.popupTitle}"/>
      </af:commandLink>
```

In this sample code snippet, a custom Additional Cart Information link is added to cart details page. Here:

- The custom link for a cart item or row is selectively displayed for the rendered property using the EL expression,

`#{backingBeanScope.additionalInfoHelperBean.renderAdditionalDetailsForCartItem}`.

To use this EL expression, develop a managed JAVA bean, say additionalRequestInfoHelperBean, with the isRenderAdditionalDetailsForCartItem method. This method should return a boolean value (true or false) based on whether the link has to be displayed or not displayed. For more information about the managed bean, see "Developing Managed Beans and Task Flows" on page 30-42.

- Launch different taskflows using the EL expression, `#{backingBeanScope.additionalInfoHelperBean.additionalInfoTaskFlowIdForCartItem}`, specified as the value of the taskFlowId attribute.

  To use this EL expression, include the getAdditionalInfoTaskFlowIdForCartItem method in the managed bean named additionalRequestInfoHelperBean. This method should return a String value representing the ID of the custom taskflow deployed as part of oracle.iam.ui.custom-dev-starter-pack.war. For example, `/WEB-INF/oracle/iam/ui/sample/catalog/tfs/additional-info-email-tf.xml#additional-info-email-tf`. For information about the taskFlowId attribute, see "Launching Custom Taskflow for Additional Request Information" on page 30-77.

- Set different popup window titles using the EL expression, `#{backingBeanScope.additionalInfoHelperBean.popupTitle}`, specified as the value of the dialogTitle attribute.

  To use this EL expression, include the getPopupTitle method in managed bean named additionalRequestInfoHelperBean. This method should return the desired String value to be displayed as popup window title. For information about the dialogTitle attribute, see "Launching Custom Taskflow for Additional Request Information" on page 30-77.

You can use the following available data to achieve the required customization:

- Data about the selected cart item or row by evaluating the `#{row}` EL expression

  This would return an object of type oracle.adfinternal.view.faces.model.binding.FacesCtrlHierNodeBinding. You can cast the result to FacesCtrlHierNodeBinding and use its API to get the cart item information.

  For example:

```
//Assume "row" is the result of evaluating EL expression "#{row}", and is NOT
null.

FacesCtrlHierNodeBinding rowData = (FacesCtrlHierNodeBinding)row;

//This code populates cartItemDataMap with the data available in the row

Map<String, Object> cartItemDataMap = new HashMap<String, Object>();

for (String name : rowData.getAttributeNames()) {
cartItemDataMap.put(name, rowData.getAttribute(name));
}

//Rest of the code can make use of the attrMap, which contains keys like
entityType, entityName, entityDisplayName etc.

//use the following cart item properties to achieve various customization
usecases
String entityType = (String)cartItemDataMap.get("entityType");
```

```
String entityName = (String)cartItemDataMap.get("entityName");
String entityDisplayName = (String)cartItemDataMap.get("entityDisplayName");
```

- Data available in the pageFlowScope map in one of the following ways:
  - By evaluating the EL expression, `#{pageFlowScope}`
  - By invoking the AdfFacesContext.getCurrentInstance().getPageFlowScope() method

    This method returns an instance of java.util.Map, which contains data as key-value pairs. A few useful values (Map Key values) stored in this map are:

    - `requestId`: The value corresponding to this key would be a Request ID, which varies depending on the UI action performed by the user.

      For example, when the user is requesting for or modifying access, or performing any other operation supported by request, the value will be -1. For all other UI actions, such as tracking or approving request, the value will be a valid Request ID.

      You can use this value to distinguish request tracking or approval flows from request creation.

    - `requestAction`: The value for this key (if *not* null) corresponds to the UI action performed by the user, such as APPROVAL_UPDATE, APPROVAL_VIEW, ACCOUNT_UPDATE, and ACCOUNT_VIEW.

      You can use this value to distinguish request tracking flow from request approval flow. APPROVAL_VIEW corresponds to request tracking flow and APPROVAL_UPDATE corresponds to request approval flow.

    - `entityType`: The value for this key (if *not* null) corresponds to the Oracle Identity Manager entity type involved in the operation, such as role and entitlement.

- Data obtained from RequestFormContext, which holds information about the cart, by invoking the oracle.iam.ui.platform.view.RequestFormContext.newInstance(null).getCurrentContext() method

  This method returns an object of type oracle.iam.ui.platform.view.RequestFormContext. For more information about RequestFormContext, see "Available EL Expressions in the RequestFormContext" on page 30-22.

30.11.7 Validating Additional Request Information

If the additional request information has any mandatory attribute values to be submitted, you can validate the submission using a RequestDataValidator plugin, which can validate RequestData.

> **See Also:** Chapter 21, "Developing Workflows for Approval and Manual Provisioning" for more information about defining and configuring plugins for requests related to application instance.

The following is a sample configuration for Assign roles operation in the plugin.xml file while registering the validator plugin. This configuration ensures that mycompany.iam.plugin.validation.AssignRolesDataValidator is invoked for all Assign roles operations.

```
<plugins pluginpoint="oracle.iam.request.plugins.RequestDataValidator">
```

```
    <plugin pluginclass="mycompany.iam.plugin.validation.AssignRolesDataValidator"
version="1.0" name="AssignRolesDataValidator">
           <metadata name="DataValidator">
                  <value>AssignRolesDataset</value>
           </metadata>
    </plugin>
</plugins>
```

Similarly, for Assign entitlements operation, plugin.xml can be configured as follows:

```
<plugins  pluginpoint="oracle.iam.request.plugins.RequestDataValidator">
    <plugin
pluginclass="mycompany.iam.plugin.validation.AssignEntitlementsDataValidator"
version="1.0" name="AssignEntitlementsDataValidator">
           <metadata name="DataValidator">
                  <value> EBSForm.UD_EBS_RESP</value>
           </metadata>
    </plugin>
</plugins>
```

30.12 Migrating UI Customizations

Migrating UI customizations from one Oracle Identity Manager environment to another environment or test to production (T2P) is described with the help of the following scenarios:

Scenario I: Incremental T2P

During the development cycle, you want to incrementally build configuration and keep moving the configuration from one Oracle Identity Manager setup to another. To do this, you use the Deployment Manager, as described in Chapter 38, "Migrating Configurations and Customizations". But exporting and importing data using the Deployment Manager does not include the UI customization, except migrating the resource bundles. For this reason, Oracle Identity Manager provides sandboxes, using which you can create customizations bound by sandboxes, test them, and eventually export/import them on an incremental basis.

However, incremental migration of customizations has a problem. You have to keep your sandboxes exported in advance, and then only publish the changes. You can test your changes only after publishing them. But if you have already published the changes, then you cannot export. This is a known issue.

Scenario II: Fusion Middleware Framework-Based Full T2P

After completion of the development and testing cycles, you want to setup the production environment on the first day of the real deployment. You want to move all configurations and customizations from the test to the production environment.

Full T2P of Oracle Identity Manager via Fusion Middleware framework-based utility also supports the movement of UI customizations. Fusion Middleware Framework-Based Full T2P is performed by using Fusion Middleware T2P utilities, such as copyBinary, pasteBinary, copyconfig, extractMoveplan, and pasteconfig, and does not use the Deployment Manager. See "Moving from a Test to a Production Environment" in the *Oracle Fusion Middleware Administrator's Guide* for detailed information about Full T2P.

If in the T2P, the list of published sandboxes are not showing up, then it is not an issue because you are expected to track published/unpublished changes in your test environment (T2P source), not in the destination or production environment.

Any unpublished sandboxes in the source or test environment means:

- You want to move the unpublished sandboxes later as incremental work after increasing the scope of those sandboxes, but currently those have not been included in the production environment.
- The unpublished sandboxes have not been tested, and you do not want to include those in the production environment.

30.13 UI Customization Best Practices

This section describes the following UI customization best practices and guidelines:

Create sandboxes with detailed description

When creating a sandbox, create it with a detailed description and list all the entities for which you are creating the sandbox. For example, if you are creating an application instance, note that this sandbox is created for application instance creation. When the application instance is created, publish the sandbox, and then go to Identity Self Service to create another sandbox to perform the UI customization. This is to avoid issues when two or more users create different sandboxes to create the same entity (application instance in this example) and try to publish it at different times.

Do not invoke Platform APIs from custom managed bean

Invoking Platform APIs directly by using Oracle Identity Manager data source in custom managed bean is not supported. Only Public APIs that are exposed through `OIMClient` can be invoked.

30.14 Rolling Back UI Customization

If you are unable to login to Oracle Identity System Administration after customizing the interface and publishing the sandbox, then perform the following steps:

1. Login to Oracle Enterprise Manager.
2. In Application Deployments, select **oracle.iam.ui.console.self-service.ear**.
3. On the top-right of the page, select **Application Deployment**, and then select **MDS Configuration** from the list.
4. At the bottom of the screen, select **Runtime MBean Browser** under the Advanced Configuration section. The right side of the screen refreshes.
5. Click the **Operations** tab.
6. Scroll down and identify the listMetadataLabels MBean operation and invoke it. Select the MBean operation that does not take any parameters. Select the sandbox precreate that you want to restore, and copy it to the clipboard.

   For example, the value you copy can be similar to: Creation_IdM_test_09:25:00.
7. Click **Return** to go back to the Operation tab.
8. Find the promoteMetadataLabel MBean operation.
9. Invoke the promoteMetadataLabel MBean operation, and enter the value that you copied in step 6.
10. Restart Oracle Identity Manager.
11. Login to Oracle Identity System Administration.

Part VIII

Interfaces to Integrate With Other Applications

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"
- Chapter 33, "Using URLs"

31

Using APIs

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:

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:

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, t3://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);
OR
tcLookupOperationsIntf lookupIntf =
oimClient.getService(tcLookupOperationsIntf.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);
```

31.1.3 Using OIMClient and tcUtilityFactory in Integrated Deployments

In Oracle Identity Manager deployment that is integrated with Access Manager (OAM), OIMSignatureAuthenticator is not configured in the Oracle Identity Manager domain's security realm. Therefore, all the custom or partner applications that you want to integrate with Oracle Identity Manager must not use signature-based login to

Oracle Identity Manager. Instead, you must follow any one of the following approaches:

- **Oracle Platform Security Services (OPSS) Framework:** If the partner or client application is a J2EE application based on Fusion Middleware stack, then it can use the following from OPSS framework:

  Note: See "Introduction to Oracle Platform Security Services" in the *Oracle Fusion Middleware Application Security Guide* for information about OPSS and its main features

  – **OPSS credential store:** This allows credentials to be managed (store, retrieve, modify) in a secure manner. You can store the password in the OPSS credential store, and retrieve it while performing a Oracle Identity Manager client login by using user ID and password. See "Managing the Credential Store" in the *Oracle Fusion Middleware Application Security Guide* for more information about OPSS credential store.

  – **OPSS SubjectSecurity API:** If a partner application wants to invoke Oracle Identity Manager EJB/Service APIs with a higher privilege, such as system administrator user, then OPSS SubjectSecurity API can be used. The following sample partner application code tries to invoke Oracle Identity Manager API with higher privilege:

  See Also: *Oracle Fusion Middleware Java API Reference for Oracle Platform Security Services* for detailed information about the SubjectSecurity API

```
//Get ActionExecutor for OIM System administrator, xelsysadm
ActionExecutor actionExecutor =
SubjectSecurity.getInstance().getActionExecutor("xelsysadm");
actionExecutor.execute(new PrivilegedAction<Object>() {
  public Object run() {
       //OIM EJB method invocation goes here….
                   Hashtable env = new Hashtable();
      //serverURL – OIM server's RMI URL
                   // ctxFactory – WLS/WAS context factory class
         env.put(OIMClient.JAVA_NAMING_PROVIDER_URL, serverURL);
         env.put(OIMClient.JAVA_NAMING_FACTORY_INITIAL,ctxFactory);
         OIMClient client = new OIMClient(env);
  //Invoking EJB service method as "xelsysadm"
         RequestService reqSrvc = client.getService(RequestService.class);
         reqSrvc.getBasicRequestData("1");//1 is the request ID.
              }
});
```

- If using the OPSS framework is not possible for some reason, then it is recommended to invoke the OIMClient API with user ID and password. However, it is up to the client or partner to store and manage the Oracle Identity Manager user's password in a secure manner.

31.2 Oracle Identity Manager Services

The Oracle Identity Manager API provides access to services available in Oracle Identity Manager. Because the APIs in Oracle Identity Manager 11*g* Release 1(11.1.1)

onwards and the legacy APIs use different conventions, this section discusses them separately in the following topics:

- Services in Oracle Identity Manager 11g
- Legacy Services or Utilities

31.2.1 Services in Oracle Identity Manager 11*g*

Services in Oracle Identity Manager 11*g* onwards are based on 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 11*g* 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
  ```

Some new services introduced in Oracle Identity Manager 11*g* Release 2 (11.1.2.2.0) are:

```
oracle.iam.api.OIMService
oracle.iam.platform.authopss.api.AuthorizationService
oracle.iam.provisioning.api.ProvisioningService
oracle.iam.provisioning.api.ApplicationInstanceService
```

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 operations to submit, withdraw, close, and search requests.

Table 31–1 (Cont.) Commonly Used Services

Service Name	Description
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.
oracle.iam.api.OIMService	Provides method to perform an operation in Oracle Identity Manager. You can pass an intent while calling API of this service. Intent here can be request or direct.

31.3.1 Mapping Between Legacy and New Services

In Oracle Identity Manager, 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 **Note:** The Group Manager APIs related all delegated admin APIs for adding and removing admins have been deprecated.	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.

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, and oimclient.jar.
2. Copy the following files manually after unzipping oimclient.zip in the remote machine:
   a. Copy $*OIM_HOME*/server/client/oimWASClient.ear to the same location of oimclient.jar.
   b. Copy $*OIM_HOME*/server/client/config/xl.policy to the conf directory.
   c. Copy $*OIM_HOME*/server/client/config/authws.conf to the conf directory.
   d. Copy $*MW_HOME*/oracle_common/modules/oracle.jrf_11.1.1/jrf-api.jar to the lib directory.
3. 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.
4. Pass the following system properties for running API clients:
   - java.security.auth.login.config=OIM_CLIENT_HOME/conf/authwl.conf
   - APPSERVER_TYPE=wls
5. Make sure following jars are in the class path:
   - commons-logging.jar
   - spring.jar
   - oimclient.jar
   - wlfullclient.jar
   - jrf-api.jar

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 Samples

This section contains the following code samples:

- Retrieving Oracle Identity Manager Information
- Using Certification APIs
- Using OIMService API

31.6.1 Retrieving Oracle Identity Manager Information

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 intented 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";
        char[] password = "xelsysadm".toCharArray();
        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 criteria1;
        criteria1 = 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 'criteria1' and
        // 'criteria2' as arguments. This showcases how to construct
        // compound search criterias.
        SearchCriteria criteria = new SearchCriteria(criteria1, 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 'Defaut'
            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
```

31.6.2 Using Certification APIs

This section provides code examples for using APIs related to certification, such as `CertificationService`.

Example 31–2 provides the code sample to get certifications belonging to a user.

Example 31–2 Retrieving Certifications Belonging to a User

```
public List<CertificationInstance> findCertifications(SearchCriteria
searchCriteria, Set<String> retAttrs, Map<String,Object> configParams) throws
CertificationServiceException;
Example of searchCriteria to use:
  SearchCriteria searchCriteria1 = new
SearchCriteria(CertificationConstants.CERTIFICATION_SEARCH_FIELDPRIMARY_REVIEW
ER_ID, userKey, SearchCriteria.Operator.EQUAL);
  SearchCriteria searchCriteria2 = new
SearchCriteria("certificationStatusForQuery",
CertificationConstants.STATE_IN_PROGRESS.toString(),
SearchCriteria.Operator.EQUAL);
  SearchCriteria searchCriteria = new SearchCriteria(searchCriteria1,
searchCriteria2, SearchCriteria.Operator.AND);
```

Example 31–3 provides the code sample to retrieve an application instance certification.

Example 31–3 Retrieving an Application Instance Certification

```
public List<IDCAccountAttributeAndRoleWrapper> loadBatchUserEntitlements(Long
certificationId, String taskUid, Long userId, PaginationContext context,
SearchCriteria searchCriteria) throws CertificationServiceException;
```

Example 31–4 provides the code sample to certify or deny entitlements.

Example 31–4 Certify or Deny Certifications

```
public void certifyUserEntitlements(Long certId, String taskUid, Long
userEntityId, Set<Long> roleEntityIds, Set<Long> accountEntityIds, Set<Long>
```

```
accountAttributeEntityIds, Integer certified, Date statusEndDate, String
comments) throws CertificationServiceException;
```

Example 31–5 provides the code sample to complete the certification.

Example 31–5 Complete the Certification

```
public CertificationInstance completeCertification(final Long certificationId,
String taskUid, char[] cleartextPassword)
```

31.6.3 Using OIMService API

This section lists out sample usage for few operations by using the OIMService API. It contains the following topics:

- RequestData Object Construction
- Samples of OIMService API Usage

31.6.3.1 RequestData Object Construction

For invoking the operations supported through OIMService API, you must first construct the RequestData object.

For all the operations that involve target user and cart item, such as role, application instance, or entitlement, construct the RequestData object as follows:

1. Create an instance of the RequestData object.
2. Create List of Beneficiary object(s), set the fields, and associate the object with RequestData object by invoking:

   ```
   requestData.setBeneficiaries();
   ```

3. Create List of RequestBeneficiaryEntity object(s), and set the cart item data, such as entity type, entity key, and operation. Associate the object with Beneficiary object by invoking:

   ```
   beneficiary.setTargetEntities();
   ```

   The entityKey, which is set by using the `entity.setEntityKey()` method for a given operation, must be based on Table 31–3.

Table 31–3 Operation and entityKey

Operation	entityKey
Provision Application Instance	Application Instance Key
Modify Account	Account Key
Revoke Account	Account Key
Enable Account	Account Key
Disable Account	Account Key
Provision Entitlement	Entitlement Key
Modify Entitlement	Entitlement Instance Key
Revoke Entitlement	Entitlement Instance Key

4. Create List of RequestBeneficiaryEntityAttribute object(s), and set the attribute name and value in each object. Associate the object with the entity object by invoking `entity.setEntityData()`.This is required only if the cart item is associated with a form.

Note:

- RequestData.setTargetEntities() must not be used in this scenario.
- See ""Samples of OIMService API Usage" on page 31-14" for more details.

For all the operations that involve only the User entity, such as Create User, Modify User, Enable User, Disable User, and Delete User, the RequestData object must be populated as follows:

1. Create an instance of the RequestData object.
2. Create List of RequestEntity object(s) by setting entity type, entity key, and operation. Entity key must be set as user key for all the operations exception Create User.
3. Create List of RequestEntityAttribute object(s), and set attribute name and value in each object. This is required only for Create User and Modify User operations.

31.6.3.2 Samples of OIMService API Usage

This section provides code samples of using the OIMService API.

Example 31–6 provides the code sample for revoking an account.

Example 31–6 Revoking an Account

```
RequestData requestData = new RequestData();
Beneficiary beneficiary = new Beneficiary();
beneficiary.setBeneficiaryKey("12"); //User with key 12
beneficiary.setBeneficiaryType(Beneficiary.USER_BENEFICIARY);

RequestBeneficiaryEntity entity = new RequestBeneficiaryEntity();
entity.setEntityType("ApplicationInstance");
entity.setEntityKey(String.valueOf(accountKey));
entity.setOperation("REVOKE");

List<RequestBeneficiaryEntity> entities = new
ArrayList<RequestBeneficiaryEntity>();

entities.add(entity);
beneficiary.setTargetEntities(entities);

List<Beneficiary> beneficiaries = new ArrayList<Beneficiary>();
beneficiaries.add(beneficiary);
requestData.setBeneficiaries(beneficiaries);
OperationResult result = oimService.doOperation(requestData,
OIMService.Intent.ANY);
if( result.getRequestID() != null ) {
//Operation resulted in to request creation.
System.out.println("Request submitted with ID: " + result.getRequestID());
} else {
System.out.println("Account is revoked successfully");
}
```

Example 31–7 provides the code sample for creating a user.

Example 31–7 Creating a User

```
RequestData requestData = new RequestData("Create User");
RequestEntity ent = new RequestEntity();
ent.setRequestEntityType(OIMType.User);
ent.setOperation("CREATE");
HashMap<String, String> userData = new HashMap<String, String>();

List<RequestEntityAttribute> attrs = new ArrayList<RequestEntityAttribute>();

RequestEntityAttribute attr = new RequestEntityAttribute("Last Name", "Doe",
RequestEntityAttribute.TYPE.String);
attrs.add(attr);
attr = new RequestEntityAttribute("First Name", "John",
RequestEntityAttribute.TYPE.String);
attrs.add(attr);
attr = new RequestEntityAttribute("User Login", "jdoe",
RequestEntityAttribute.TYPE.String);
attrs.add(attr);
Long organizationKey = new Long(1);
attr = new RequestEntityAttribute("Organization", organizationKey ,
RequestEntityAttribute.TYPE.Long);
attrs.add(attr);
attr = new RequestEntityAttribute("Role", "Full-Time",
RequestEntityAttribute.TYPE.String);
attrs.add(attr);

ent.setEntityData(attrs);

List<RequestEntity> entities = new ArrayList<RequestEntity>();
entities.add(ent);
requestData.setTargetEntities(entities);
OperationResult result = oimService.doOperation(requestData,
OIMService.Intent.ANY);
if( result.getRequestID() != null ) {
//Operation resulted in to request creation.
System.out.println("Request submitted with ID: " + result.getRequestID());
} else {
System.out.println("User is created successfully");
}
```

32

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 Manage. It contains the following topics:

> **Note:** Oracle Identity Manager does not support the SPML DSML service. However, you can manually deploy the spml-dsml.ear archive file for Microsoft Active Directory password synchronization usecase only.

32.1 Introduction

This section introduces the use of SPML services using XSD profile in Oracle Identity Manage.

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 Manage makes use of the XSD profile.

Types of Interactions

The SPML specification allows interactions to be synchronous or asynchronous.

Oracle Identity Manage supports only asynchronous interactions for add, modify, delete, lookup, suspend, and resume request. For username services, all services are synchronous. Lookup is supported for user and role, which happens in synchronous manner and it can be performed using entity key, unique user/role name , LDAP GUID, and LDAP DN. Out of these, LDAP GUID and LDAP DN are applicable when LDAP synchronization is enabled. Oracle Identity Manage responds immediately with a pending status, and it is up to the requestor to get the current state by issuing a statusRequest.

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 General Considerations

Perform the following to ensure that SPML works with Oracle Identity Manager:

- Assigning SPML Admin Role to the User
- Creating Autoapproval Policies

32.2.1 Assigning SPML Admin Role to the User

Oracle Identity Manager provides an admin role for SPML. The user with this admin role is able to perform SPML requests for all usecases. The name of the SPML admin role is SPML Admin.

The SPML Admin admin role has the following permissions:

- Create, modify, and delete users via request
- Search users on all the attributes
- Enable user status via request
- Disable user status via request
- Add role memberships via request
- Delete role memberships via request
- Search roles on all the attributes
- Create, modify, and delete roles via request

The SPML Admin admin role is a global admin role published to the TOP organization. Therefore, only System Administrator or Organization Administrator of the TOP organization can assign this admin role to any user, in scope of the TOP organization.

In a new deployment of Oracle Identity Manager, the SPML Admin admin role is not assigned to any users by default. As a result, System Administrator or Organization Administrator of the TOP organization must manually assign this role to relevant users. In addition, the SPML Admin admin role can be assigned in scope of the TOP organization with include hierarchy or include sub-orgs option enabled. As a result, all permissions of the SPML Admin admin role are assigned to the user for all organizations in Oracle Identity Manager.

Note: In an upgraded deployment of Oracle Identity Manager, the existing SPML users are upgraded to assign SPML Admin admin role so that they can continue to perform SPML requests.

To assign SPML Admin admin role to a user:

1. Login to Oracle Identity Self Service. If you want to assign the SPML Admin admin role to a new user, then create the user.
2. Open the TOP organization, and click the **Admin Roles** tab.

3. Select the SPML Admin admin role, and assign the user with this role.
4. Modify the newly added membership to select the **include-suborgs** option, and then click **Apply**.

32.2.2 Creating Autoapproval Policies

Autoapproval policy rules are required so that SPML requests are auto-approved. In Oracle Identity Manager 11*g* Release 2 (11.1.2.2.0), there are no request templates. Therefore, the autoapproval policy rules must be manually created by System Administrators for each SPML user.

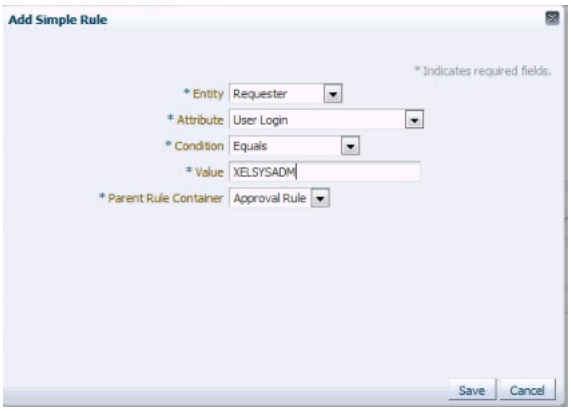
The autoapproval policies created can be based on user login, and the approval process configuration for each policy must be selected as Auto Approval. You must create autoapproval policies for each SPML operation, such as create user, modify user, create role, and so on, so that all the SPML requests raised by SPML user for these operations are approved at both request level and operational level.

Note: Only request-level approval policies are required for role create, role modify, and role delete operations.

To create the autoapproval policy rules:

1. Create new auto approval policy rules for the following:
   - Create user request level
   - Create user operation level
   - Modify user request level
   - Modify user operation level
   - Delete user request level
   - Delete user operation level
   - Enable user request level
   - Enable user operation level
   - Disable user request level
   - Disable user operation level
   - Assign roles request level
   - Assign roles operation level
   - Remove roles request level
   - Remove roles operation level
   - Create role request level
   - Modify role request level
   - Delete role request level
2. Create rules in each policy based on the user login. Ensure that user login evaluation is case-sensitive. Figure 32–1 shows a sample rule.

Figure 32–1 Sample Approval Policy Rule



Note: The autoapproval policies created for a SPML user based on the user login cause all requests raised by the user to be auto-approved irrespective of the request being raised via SPML, UI, or any other flow.

32.3 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 autogenerated in Oracle Identity Manage if the appropriate plug-ins are installed.
- For creating a user with a given password, provide the user password in Base64 encoded format within SPML create user payload. For example:

```
<pso:password>
<pso:value>V2VsY29tZTc=</pso:value>
</pso:password>
```

  You can use any standard Java library to obtain the Base64 encoded value for a string. An example of such a library is Apache commons library - org.apache.commons.codec.binary.Base64. encodeBase64("*PASSWORD*".getBytes()).
- Role created by user with SPML Admin role are autopublished to the TOP organization including its suborganizations.

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	`addRequest` element as defined by [SPMLv2]. Optional, reference capability for role memberships.
Output	`addResponse` 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 `customError` SPML custom error code. Only validation errors are returned in the Response. No request ID is returned.
Examples	See the Appendix for these examples: ■ "SPML Example - Add User" on page 32-28 ■ "SPML Example – Add User with Role Assignment" on page 32-36

32.4 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
Input	`modifyRequest` element as defined by [SPMLv2]. Use modificationMode="`delete`" for deleting role membership and modificationMode="`add`" for adding role membership. Role memberships declared using Reference capability, with typeOfReference="`inheritsFrom`" and Role GUID as PSO ID.
Output	`modifyResponse` element as defined by [SPMLv2].
Processing	The `modifyRequest` operation allows modifying an existing identity or existing role. This operation checks for SPML execution mode for both identity and role. Invalid execution mode returns an `unsupportedExecutionMode` SPML error code. If the modify request does not contain identity PSO object, or contains invalid GUIDs the operation returns `malformedRequest` or `invalidIdentifier` SPML malformed request error respectively. Other runtime errors are reported using customError SPML custom error code.

Table 32–2 (Cont.) Role Membership Management with modifyRequest

Item/Feature	Description
Examples	See the Appendix for these examples: ■ "SPML Example - Assign Role Membership" on page 32-38 ■ "SPML Example – Revoke Role Membership" on page 32-38

32.5 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	`deleteRequest` element as defined by [SPMLv2].
Output	`deleteResponse` element as defined by [SPMLv2].
Processing	The deleteRequest operation allows deletion of an existing identity or existing role. This operation checks for SPML execution mode for both identity and role. Invalid execution mode returns an `unsupportedExecutionMode` SPML error code. If the delete request does not contain identity PSO object, or contains invalid GUIDs the operation returns `malformedRequest` or `invalidIdentifier` SPML malformed request error respectively. Other runtime errors are reported using `customError` SPML custom error code.
Examples	See the example "SPML Example - Delete Role" on page 32-43.

32.6 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

Table 32–4 (Cont.) Check Request Status

Item/Feature	Description
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 "SPML Example - Status Request" on page 32-43

32.7 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 Manage; the object types that we support are all Oracle Identity Manage 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].
Processing	Only the XML Schema profile is supported. Any another profile request results in a failure with the `unsupportedProfile` error code. A single, static provisioning target named `Oracle Identity Manager` is supported. 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 `listTargetsResponse` element.

32.8 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	`suspendRequest` element as defined by [SPMLv2].
Output	`suspendResponse` element as defined by [SPMLv2].

Table 32–6 (Cont.) Suspending a User with suspendRequest

Item/Feature	Description
Processing	This operation requires a valid user PSO ID and optionally an effective suspension date. If the PSO identifier is invalid, the `noSuchIdentifier` error code is returned. The suspend operation is applicable for users only. It returns `unsupportedOperation` error if the PSO object is not an identity.
Examples	See the example "SPML Example - Suspend User" on page 32-34.

32.9 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	`resumeRequest` element as defined by [SPMLv2].
Output	`resumeResponse` element as defined by [SPMLv2].
Processing	This operation requires a valid user PSO ID and optionally an effective resumption date. If the PSO identifier is invalid, the `noSuchIdentifier` error code is returned. The resume operation is applicable for users only. It returns `unsupportedOperation` error if the PSO object is not an identity.
Examples	See the example "SPML Example - Resume User" on page 32-33.

32.10 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.

Table 32–8 (Cont.) Checking if User Has Been Suspended with activeRequest

Item/Feature	Description
Examples	See the example "SPML Example - Check If User is Active" on page 32-35.

32.11 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 32-35.

32.12 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	`suggestUsernameRequest` element as defined by [SPMLv2].
Output	`suggestUsernameResponse` 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 `customError` code.
Examples	See the example "SPML Example - Suggest User Name" on page 32-34.

32.13 Lookup an Identity or Role (SPML Core Service: lookupRequest)

The lookupRequest operation enables a requestor to lookup for a user or role in the system by using any one of entity key, user/role name, LDAP GUID, LDAP DN. Out of these, LDAP GUID and LDAP DN are applicable only in an environment for which LDAP synchronization is enabled. Lookup is supported only in synchronous mode.

Requestor can also choose to filter the response by using the returnData attribute, whose default value is `everything`. The returnData attribute can have the following values:

- `returnData='identifier'`: The provider returns only the identifier of a requested object.
- `returnData='data'`: The provider returns the identifier of a requested object and all the attributes associated with that object.
- `returnData='everything'`: The provider returns the identifier of a requested object, all the attributes associated with that object, and any capability data associated with the request object. For identity lookup, capability data contains the direct or indirect roles the user has. For role lookup, capability data contains the parent roles and the direct or indirect children roles.

Any user can perform the lookup by using the SPML interface. Based on authorization, the requestor can lookup user or role. If there is no appropriate authorization privilege, then error code is returned in the SPML response. Only authorized attributes on which requestor has permission to view is returned in the response.

Table 32–11 lists the features of the lookupRequest operation.

Table 32–11 Identity/Role Lookup using lookupRequest

Item/Feature	Description
SPML Execution Mode	Synchronous
Input	lookupRequest element as defined by [SPMLv2].
Output	lookupResponse element as defined by [SPMLv2].

Table 32–11 (Cont.) Identity/Role Lookup using lookupRequest

Item/Feature	Description

Processing	This operation requires a valid user PSO ID in the following format:
	psoID ID="*ENTITY_TYPE*:*PSO_ID_TYPE*:*LOOKUP_VALUE*"
	Here:
	ENTITY_TYPE is identity or role. This value is mandatory.
	PSO_ID_TYPE is key, DN, GUID, or name. This value is optional.
	LOOKUP_VALUE is valid value for lookup. This value is mandatory.
	The following are example values:
	`psoID ID="identity:key:6"`
	`psoID ID="identity:6"`
	`psoID ID="identity:name:JohnSmith"`
	`psoID ID="identity:dn: cn=john,cn=Users,dc=us,dc=oracle,dc=com"`
	`psoID ID="identity:guid: CEF2C4F20E5BF04DE040F20A9681408D"`
	`psoID ID="role:key:6"`
	`psoID ID="role:6"`
	`psoID ID="role:name:ManagerRole"`
	`psoID ID="role:dn: cn= ManagerRole,cn=Groups,dc=us,dc=oracle,dc=com"`
	`psoID ID="role:guid: CEF2C4F20E5BF04DE040F20A9681408D"`
	If the PSO identifier does not exist, then an error code is returned. If the request is valid and if the specified user/role exists, then depending upon the returnData attribute, the result is returned in the lookupResponse.

Examples	See the example "SPML Example - Identity/Role Lookup" on page 32-46.

32.14 Reset Password (SPML Core Service: resetPasswordRequest)

The resetPasswordRequest operation enables a requestor to reset the password for a user.

Table 32–12 lists the features of the resetPasswordRequest operation.

Table 32–12 Resetting the user password with resetPasswordRequest

Item/Feature	Description
SPML Execution Mode	Synchronous
Input	`resetPasswordRequest` element as defined by [SPMLv2]. Optional notification data for controlling end user email notification.
Output	`resetPasswordRequest` element as defined by [SPMLv2].

Table 32–12 (Cont.) Resetting the user password with resetPasswordRequest

Item/Feature	Description
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: ■ SentNotification: Boolean flag to determine whether or not to send notification. ■ SendNotificationTo: Comma separated email address. Processing errors are reported with SPML `customError` code.
Examples	See the following examples: ■ "SPML Example - Reset Password" on page 32-49 ■ "SPML Example - Reset Password with Notification" on page 32-50

32.15 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 32–13 lists the features of the lookupUsernamePolicy operation.

Table 32–13 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 "SPML Example - Lookup User Name Policy" on page 32-51.

32.16 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 32–14 lists the features of the cancelRequest operation.

Table 32–14 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].

Table 32–14 (Cont.) Cancel a Request with cancelRequest

Item/Feature	Description
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 "SPML Example - Cancel Request" on page 32-51.

32.17 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.

Table 32–15 lists the features of the batchRequest operation.

Table 32–15 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 "SPML Example - Batch Request" on page 32-52.

32.18 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

32.18.1 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.

32.18.2 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-sece
xt-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>
```

32.18.3 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*.

32.19 Operations Not Supported

Oracle Identity Manager 11*g* Release 2 (11.1.2.2.0) 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

32.20 SPML Attributes and LDAP Mappings, and Oracle Identity Manage 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:

32.20.1 Identity PSO Attributes

Table 32–16 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 32–16 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
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

Table 32–16 (Cont.) Identity PSO Attributes

SPML Attribute Name	Syntax	Description	LDAP Mapping (Oracle Internet Directory)
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 32–17.	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
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

Table 32–16 (Cont.) Identity PSO Attributes

SPML Attribute Name	Syntax	Description	LDAP Mapping (Oracle Internet Directory)
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 32–17 shows the valid values for the `employeeType` attribute:

Table 32–17 Valid Values of employeeType

Value	Meaning
Full-Time	Full-Time Employee
Part-Time	Part-Time Employee
Temp	Temp
Intern	Intern
Consultant	Consultant

Table 32–17 (Cont.) Valid Values of employeeType

Value	Meaning
Contractor	Contractor
EMP	Employee
CWK	Contingent Worker
NONW	Non Worker
OTHER	Other Employee Type

Note: Oracle Identity Manage passes only the codes shown in the Value column; the meaning of each code is shown for reference.

32.20.1.1 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>
...
```

32.20.2 Role PSO Attributes

Table 32–18 lists the role attributes supported by the SPML implementation in Oracle Identity Manager and how these attributes map to LDAP objects/attributes.

Table 32–18 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.

32.20.2.1 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.

32.20.3 Preference Attributes

Table 32–19 lists the preference attributes supported by the SPML implementation in Oracle Identity Manager:

Table 32–19 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 32–19 (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 32–19 (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 32–19 (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

32.20.4 Special Character Restrictions in Oracle Identity Manage Attributes

This section lists character restrictions applicable to Oracle Identity Manage 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

32.20.4.1 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.

32.20.4.2 Special Characters in the Password Field

There are no restrictions on the usage of special character in the password field. The only restriction on the special character in the password field is imposed by the applicable password policy of the user.

The user password must be provided in Base64 encoded format within SPML payload. For example:

```
<pso:password>
<pso:value>V2VsY29tZTc=</pso:value>
</pso:password>
```

32.20.4.3 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

32.20.4.4 Usage of Semicolon

The semicolon (;) can be used only in access policy names.

32.20.4.5 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 (>)

32.20.5 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

32.20.5.1 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>
```

32.20.5.2 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>
```

32.21 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 – Add User with Role Assignment
- SPML Example - Assign Role Membership
- 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 - Identity/Role Lookup
- SPML Example - Reset Password
- SPML Example - Reset Password with Notification
- SPML Example - Lookup User Name Policy
- SPML Example - Cancel Request
- SPML Example - Batch Request

32.21.1 SPML Example - Add User

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">
<!--Zero or more repetitions:-->
<data>
<!--You have a CHOICE of the next 3 items at this level-->
<pso:identity>
<!--Optional:-->
<pso:attributes>
<!--Here, We are trying to set OIM Organization (act_key) -->
<pso:attr name="oimOrganization">
<pso:value>3</pso:value>
</pso:attr>
<!--Here, My Attribute is a UDF, with 'My Attribute' also added in
CreateUserDataset.xml -->
<pso:attr name="My Attribute">
<pso:value>New Value</pso:value>
</pso:attr>
</pso:attributes>
<!--Optional:-->
<pso:activeEndDate>2009-06-12T00:00:00</pso:activeEndDate>
<!--Optional:-->
<pso:activeStartDate>2009-06-11T00:00:00</pso:activeStartDate>
<pso:commonName>
<pso:values>
<pso:value>CommonName</pso:value>
</pso:values>
</pso:commonName>
<!--Optional:-->
<pso:countryName>India</pso:countryName>
<!--Optional:-->
<pso:departmentNumber>
<!--1 or more repetitions:-->
<pso:value>123456</pso:value>
```

```
</pso:departmentNumber>
<!--Optional:-->
<pso:description>
<!--1 or more repetitions:-->
<pso:values>
<!--1 or more repetitions:-->
<pso:value>All Optional Fields Profile</pso:value>
</pso:values>
</pso:description>
<!--Optional:-->
<pso:displayName>
<!--1 or more repetitions:-->
<pso:value locale="en">All Optional Values</pso:value>
</pso:displayName>
<!--Optional:-->
<pso:employeeNumber>24073</pso:employeeNumber>
<pso:employeeType>
<!--1 or more repetitions:-->
<pso:values>
<!--1 or more repetitions:-->
<pso:value>Part-Time</pso:value>
</pso:values>
</pso:employeeType>
<!--Optional:-->
<pso:facsimileTelephoneNumber>
<!--1 or more repetitions:-->
<pso:number>08041085304</pso: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>2009-06-11T00:00:00</pso:hireDate>
<!--Optional:-->
<pso:homePhone>
<!--1 or more repetitions:-->
<pso:number>9999999999</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@mycompany.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: This sets LDAP Organization of the user.-->
<pso:organization>
<pso:value>Sales</pso:value>
</pso:organization>
<!--Optional: This sets LDAP Organization Unit of the user.-->
<pso:organizationUnit>
<pso:value>Marketing</pso:value>
</pso:organizationUnit>
<!--Optional:-->
<pso:pager>
<!--1 or more repetitions:-->
<pso:number>7777</pso:number>
</pso:pager>
<!--Optional: This sets user password to Welcome7. If not provided, the password
will be auto-generated.-->
<pso:password>
<!--1 or more repetitions:-->
<pso:value>V2VsY29tZTc</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 32-43.
- The displayName attribute has Multiple Language Support (MLS), and language values can be specified as "en", "fr", and so on.

32.21.2 SPML Example - Delete User

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="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"/>
```

32.21.3 SPML Example - Modify User

The Request is as follows:

```
<modifyRequest xmlns="urn:oasis:names:tc:SPML:2:0"
xmlns:pso="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" />
<psoID ID="identity:6C9B96E99FC8DC32E040E50A3D5252F5" />
<modification modificationMode="add">
<component path="/identity" namespaceURI="http://www.w3.org/TR/xpath20" />
<data>
<pso:identity>
<pso:initials>
<!--1 or more repetitions:-->
<pso:value>J S</pso:value>
</pso:initials>
</pso:identity>
</data>
</modification>
<modification modificationMode="replace">
<component path="/identity" namespaceURI="http://www.w3.org/TR/xpath20" />
```

```
<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:commonName>
<!--1 or more repetitions:-->
<pso:values>
<!--1 or more repetitions:-->
<pso:value>FR Alice Krug1</pso:value>
</pso:values>
</pso:commonName>
</pso:identity>
</data>
</modification>
<modification modificationMode="delete">
<component path="/identity" namespaceURI="http://www.w3.org/TR/xpath20" />
<data>
<pso:identity>
<pso:pager>
<!--1 or more repetitions:-->
<pso:number>333</pso:number>
</pso: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"/>
```

32.21.4 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"/>
```

32.21.5 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>
```

32.21.6 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"/>
```

32.21.7 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"/>
```

32.21.8 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" />
```

32.21.9 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 if
first name is present then <first name>.<last name>@<domain>, else <last
name>@<domain></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>
```

32.21.10 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_prc02.xml">
        <spml:data xsi:type="spml:PSOType">
           <identity
             xmlns="http://xmlns.oracle.com/idm/identity/PSO"
             xmlns:pso="http://xmlns.oracle.com/idm/identity/PSO">
            <pso:commonName>
              <pso:values>
                <pso:value>John Doe</pso:value>
              </pso:values>
            </pso:commonName>
            <pso:displayName>
              <pso:value>John Doe</pso:value>
            </pso:displayName>
            <pso:givenName>
              <pso:value>John</pso:value>
            </pso:givenName>
```

```
            <pso:mail>
               <pso:value>john.doe@acme.com</pso:value>
            </pso:mail>
            <pso:middleName/>
            <pso:organization>
              <pso:values>
                <pso:value>ACME, Inc.</pso:value>
              </pso:values>
            </pso:organization>
             <pso:password>
              <pso:value>qwert</pso:value>
             </pso:password>
            <pso:surname>
              <pso:values>
                <pso:value>Doe</pso:value>
              </pso:values>
            </pso:surname>
            <pso:username>
               <pso:value>jdoe</pso:value>
            </pso:username>
            <pso:employeeType>
              <pso:values>
               <pso:value>Full-Time</pso:value>
              </pso:values>
            </pso:employeeType>
          </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>
```

32.21.11 SPML Example - Assign Role Membership

The Request example is as follows:

Note: only those roles can be granted to users via SPML Add Role Membership that are:

- Either published to the TOP organization with hierarchy, OR
- Published to at least one member organization of the user to whom the role is to be granted via SPML

All other role grant attempts via SPML will fail authorization checks. These roles must be explicitly published to relevant organizations by using the UI or APIs to let SPML Role grant work.

```
<modifyRequest
xmlns="urn:oasis:names:tc:SPML:2:0"
xmlns:pso="http://xmlns.oracle.com/idm/identity/PSO"
executionMode="asynchronous"
locale="en"
policyURI="gant_role_01">
<psoID 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"/>
```

32.21.12 SPML Example – Revoke Role Membership

The Request is as follows:

```
<modifyRequest
xmlns="urn:oasis:names:tc:SPML:2:0"
xmlns:pso="http://xmlns.oracle.com/idm/identity/PSO"
executionMode="asynchronous"
locale="en"
policyURI="revoke_role_01">
<psoID 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"/>
```

32.21.13 SPML Example - Add Role

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_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-->
        <pso:role>
            <pso:attributes>
                <pso:attr name="Role Category Name">
                    <!-- pso:value>OIM Roles</pso:value-->
                    <pso:value>Default</pso:value>
                </pso:attr>
            </pso:attributes>
            <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: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>
```

32.21.14 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>
    <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"/>
```

32.21.15 SPML Example - Modify Role

The Request is as follows:

```
<modifyRequest xmlns="urn:oasis:names:tc:SPML:2:0"
xmlns:pso="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" />
<psoID ID="role:6C9B96E99FC8DC32E040E50A3D5252F5" />
<modification modificationMode="replace">
<component path="/role" namespaceURI="http://www.w3.org/TR/xpath20" />
<data>
<pso:role>
<pso:description>
<!--1 or more repetitions:-->
<pso:values>
<pso:value>UK Updated Administrator</pso:value>
</pso:values>
</pso:description>
</pso: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"/>
```

32.21.16 SPML Example - Add Parent to a Role

The Request is as follows:

```
<modifyRequest xmlns="urn:oasis:names:tc:SPML:2:0"
xmlns:pso="http://xmlns.oracle.com/idm/identity/PSO" executionMode="asynchronous"
locale="string" policyURI="http://www.sample.com/string/string"
requestID="string" returnData="identifier">
```

```
<psoID ID="role:26" targetID="target" />
<modification modificationMode="modify">
<component path="/role" namespaceURI="http://www.w3.org/TR/xpath20" />
<data>
<pso:role>
<pso:description>
<!--1 or more repetitions:-->
<pso:values>
<!--1 or more repetitions:-->
<pso:value>UK Updated Administrator</pso:value>
</pso:values>
</pso:description>
</pso: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"/>
```

32.21.17 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:pso="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"/>
```

32.21.18 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"/>
```

32.21.19 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>2009-12-25T00:00:00.000Z</ns2:activeEndDate>

<ns2:activeStartDate>2009-12-24T00:00:00.000Z</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>1999-12-24T00:00:00.000Z</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>
        </pso>
    </ns2:addResponse>
</ns9:StatusResponseType>
```

32.21.20 SPML Example - Identity/Role Lookup

The request is as follows:

```
<ns1:lookupRequest xmlns:ns1="urn:oasis:names:tc:SPML:2:0"
returnData="everything">
            <ns1:psoID ID="identity:key:1" />
</ns1:lookupRequest>
```

The response is as follows:

```
<ns3:lookupResponse 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">
<ns3:capabilityData capabilityURI="urn:oasis:names:tc:SPML:2:0:reference"
mustUnderstand="true">
<ns7:reference typeOfReference="memberOf"><ns7:toPsoID ID="1"/>
</ns7:reference>
</ns3:capabilityData><ns3:pso>
<ns3:data xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="ns2:ProvisioningObjectType">
<ns2:identity><ns2:attributes><ns2:attr name="usr_disabled">
<ns2:value>0</ns2:value></ns2:attr><ns2:attr name="Display Name">
<ns2:value>zh-TW=System Administrator</ns2:value><ns2:value>pt-BR=System
Administrator</ns2:value>
<ns2:value>base=System Administrator</ns2:value><ns2:value>fr=System
Administrator</ns2:value>
<ns2:value>en=System Administrator</ns2:value>
<ns2:value>zh-CN=System Administrator</ns2:value>
</ns2:attr><ns2:attr name="usr_locked">
<ns2:value>0</ns2:value>
</ns2:attr><ns2:attr name="usr_created">
<ns2:value>Mon Dec 03 03:42:21 PST 2012</ns2:value>
</ns2:attr><ns2:attr name="Full Name">
<ns2:value>base=null</ns2:value>
</ns2:attr><ns2:attr name="usr_pwd_expire_date">
<ns2:value>Tue Apr 02 03:42:21 PDT 2013</ns2:value>
</ns2:attr><ns2:attr name="Email">
```

```
<ns2:value>donotreply@mydomain.com</ns2:value>
</ns2:attr><ns2:attr name="usr_data_level">
<ns2:value>2</ns2:value></ns2:attr>
<ns2:attr name="usr_login_attempts_ctr">
<ns2:value>0</ns2:value></ns2:attr>
<ns2:attr name="Last Name">
<ns2:value>Administrator</ns2:value>
</ns2:attr><ns2:attr name="First Name">
<ns2:value>System</ns2:value>
</ns2:attr><ns2:attr name="usr_createby">
<ns2:value>1</ns2:value></ns2:attr>
<ns2:attr name="usr_updateby">
<ns2:value>1</ns2:value>
</ns2:attr><ns2:attr name="User Login">
<ns2:value>XELSYSADM</ns2:value>
</ns2:attr><ns2:attr name="Role">
<ns2:value>Full-Time</ns2:value>
</ns2:attr><ns2:attr name="usr_pwd_warn_date">
<ns2:value>Tue Mar 26 03:42:21 PDT 2013</ns2:value>
</ns2:attr><ns2:attr name="Organization Name">
<ns2:value>Xellerate Users</ns2:value></ns2:attr>
<ns2:attr name="usr_update"><ns2:value>Mon Dec 03 03:42:21 PST 2012</ns2:value>
</ns2:attr><ns2:attr name="usr_pwd_reset_attempts_ctr">
<ns2:value>0</ns2:value></ns2:attr><ns2:attr name="usr_create">
<ns2:value>Mon Dec 03 03:42:21 PST 2012</ns2:value>
</ns2:attr><ns2:attr name="Xellerate Type">
<ns2:value>End-User Administrator</ns2:value>
</ns2:attr><ns2:attr name="Common Name">
<ns2:value>System Administrator</ns2:value></ns2:attr>
<ns2:attr name="act_key">
<ns2:value>1</ns2:value>
</ns2:attr><ns2:attr name="usr_key">
<ns2:value>1</ns2:value></ns2:attr>
<ns2:attr name="Common Name Generated">
<ns2:value>0</ns2:value>
</ns2:attr><ns2:attr name="Status">
<ns2:value>Active</ns2:value>
</ns2:attr></ns2:attributes>
<ns2:commonName>
<ns2:values><ns2:value>System Administrator</ns2:value>
</ns2:values></ns2:commonName>
<ns2:displayName>
<ns2:value locale="zh-TW">System Administrator</ns2:value><ns2:value
locale="pt-BR">System Administrator</ns2:value><ns2:value locale="base">System
Administrator</ns2:value>
<ns2:value locale="fr">System Administrator</ns2:value><ns2:value
locale="en">System Administrator</ns2:value><ns2:value locale="zh-CN">System
Administrator</ns2:value>
</ns2:displayName><ns2:employeeType>
<ns2:values>
<ns2:value>Full-Time</ns2:value></ns2:values>
</ns2:employeeType>
<ns2:givenName>
<ns2:value>System</ns2:value></ns2:givenName>
<ns2:mail>
<ns2:value>donotreply@mydomain.com</ns2:value>
</ns2:mail><ns2:surname><ns2:values>
<ns2:value>Administrator</ns2:value></ns2:values>
</ns2:surname>
<ns2:userId><ns2:value>XELSYSADM</ns2:value>
```

```
</ns2:userId><ns2:userType>End-User Administrator</ns2:userType>
</ns2:identity>
</ns3:data>
</ns3:pso>
</ns3:lookupResponse>
```

Another request is as follows:

```
<ns1:lookupRequest xmlns:ns1="urn:oasis:names:tc:SPML:2:0"
returnData="everything">
            <ns1:psoID ID="role:name: FinanceRole " />
</ns1:lookupRequest>
```

The response is as follows:

```
<ns3:lookupResponse 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">
<ns3:capabilityData capabilityURI="urn:oasis:names:tc:SPML:2:0:reference"
mustUnderstand="true">
<ns7:reference typeOfReference="inheritsFrom">
<ns7:toPsoID ID="10"/>
</ns7:reference>
<ns7:reference typeOfReference="inheritsFrom">
<ns7:toPsoID ID="7"/>
</ns7:reference><ns7:reference typeOfReference="memberOf">
<ns7:toPsoID ID="8"/></ns7:reference>
</ns3:capabilityData><ns3:pso>
<ns3:data xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="ns2:ProvisioningObjectType">
<ns2:role><ns2:attributes>
<ns2:attr name="Role Display Name">
<ns2:value>FinanceRole</ns2:value>
</ns2:attr><ns2:attr name="Role Unique Name">
<ns2:value>FinanceRole</ns2:value>
</ns2:attr><ns2:attr name="Owner Login">
<ns2:value>XELSYSADM</ns2:value>
</ns2:attr><ns2:attr name="ugp_createby">
<ns2:value>1</ns2:value></ns2:attr>
<ns2:attr name="ugp_create">
<ns2:value>Wed Nov 21 23:28:42 PST 2012</ns2:value>
</ns2:attr><ns2:attr name="Role Owner Key">
<ns2:value>1</ns2:value></ns2:attr>
<ns2:attr name="Role Description">
<ns2:value>desc</ns2:value>
</ns2:attr><ns2:attr name="Role Name">
<ns2:value>FinanceRole</ns2:value>
</ns2:attr><ns2:attr name="ugp_update">
<ns2:value>Wed Nov 21 23:28:42 PST 2012</ns2:value></ns2:attr>
<ns2:attr name="Owner Email">
<ns2:value>donotreply@mydomain.com</ns2:value></ns2:attr>
<ns2:attr name="Role Namespace"><ns2:value>Default</ns2:value></ns2:attr>
<ns2:attr name="Owner Display Name">
<ns2:value>System Administrator</ns2:value>
</ns2:attr><ns2:attr name="Role Key">
```

```
<ns2:value>6</ns2:value>
</ns2:attr>
<ns2:attr name="ugp_updateby">
<ns2:value>1</ns2:value>
</ns2:attr>
<ns2:attr name="Role Category Key">
<ns2:value>2</ns2:value>
</ns2:attr><ns2:attr name="Owner Last Name">
<ns2:value>Administrator</ns2:value>
</ns2:attr><ns2:attr name="Role Email">
<ns2:value>email@email.com</ns2:value>
</ns2:attr><ns2:attr name="Owner First Name">
<ns2:value>System</ns2:value></ns2:attr>
<ns2:attr name="Role Category Name">
<ns2:value>OIM Roles</ns2:value>
</ns2:attr>
</ns2:attributes>
<ns2:commonName>
<ns2:values>
<ns2:value>FinanceRole</ns2:value>
</ns2:values></ns2:commonName>
<ns2:description>
<ns2:values>
<ns2:value>desc</ns2:value></ns2:values>
</ns2:description>
<ns2:displayName><ns2:value>FinanceRole</ns2:value>
</ns2:displayName></ns2:role>
</ns3:data>
</ns3:pso>
</ns3:lookupResponse>
```

32.21.21 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-sece
xt-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>
```

32.21.22 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-sece
xt-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,te
rrence.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>
```

32.21.23 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>
```

32.21.24 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>
```

32.21.25 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:pso="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-->
                    <pso:identity>
                        <pso:commonName>
                            <pso:values locale="en">
                                <pso:value>John Smith</pso:value>
                            </pso:values>
                        </pso:commonName>
                        <pso:countryName>Canada</pso:countryName>
                        <pso:departmentNumber>
                            <pso:value>123</pso:value>
                        </pso:departmentNumber>
                        <pso:description>
                            <pso:values>
                                <pso:value>John Smiths profile</pso:value>
                            </pso:values>
                        </pso:description>
                        <pso:displayName>
                            <pso:value>John Smith</pso:value>
                        </pso:displayName>
                        <pso:employeeNumber>333</pso:employeeNumber>
                        <pso:employeeType>
                            <pso:values>
                                <pso:value>Full-Time</pso:value>
                            </pso:values>
                        </pso:employeeType>
                        <pso:facsimileTelephoneNumber>
                            <pso:number>6506072253</pso:number>
                        </pso:facsimileTelephoneNumber>
                        <pso:generationQualifier>
                            <pso:value>II</pso:value>
                        </pso:generationQualifier>
                        <pso:givenName>
                            <pso:value>John</pso:value>
                        </pso:givenName>
                        <pso:hireDate>1999-12-24T00:00:00</pso:hireDate>
                        <pso:homePhone>
                            <pso:number>8888888888</pso:number>
                        </pso:homePhone>
                        <pso:homePostalAddress>
                            <pso:value>Baker street</pso:value>
                        </pso:homePostalAddress>
                        <pso:initials>
```

```
    <pso:value>J S</pso:value>
</pso:initials>
<pso:jpegPhoto>
    <pso:value>c3RyaW5n</pso:value>
</pso:jpegPhoto>
<pso:localityName>
    <pso:value>SFO</pso:value>
</pso:localityName>
<pso:mail>
    <pso:value>jsmith@mydomain.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@mydomain.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>
```

33

Using URLs

Other web applications may need to redirect end users and administrators to the Oracle Identity Self Service task flows for their Identity Self Service and Identity Administration requirements.

For the task flows listed in Table 33–1, Oracle Identity Self Service exposes direct URLs that can be embedded as links into other web application pages.

Table 33–1 Task Flows and Direct URLs

Task Flow Title	Direct URL (relative to https://*OIMHOST*:*PORT*)	Description
Home	/identity/faces/home?tf=home	Displays Home page task flows.
My Information	/identity/faces/home?tf=my_information	Displays User Profile, Change Password, Challenge Questions, Proxies, and Direct Reports.
My Access Roles	/identity/faces/home?tf=my_access_roles	Displays enterprise role memberships with Request Roles action.
My Access Admin Roles	/identity/faces/home?tf=my_access_admin_roles	Displays admin role memberships.
My Access Accounts	/identity/faces/home?tf=my_access_accounts	Displays assigned accounts with Request Accounts action.
My Access Entitlements	/identity/faces/home?tf=my_access_entitlements	Displays assigned entitlements with Request Entitlements action.
Approval Details	/identity/faces/home?tf=approval_details	Displays pending approval tasks.
Request Details	/identity/faces/home?tf=request_details&requestId=<Request Id>	Request ID is mandatory. This is the ID generated on submission of a request.
Organizations	/identity/faces/home?tf=organizations	Displays Organization search page
Roles	/identity/faces/home?tf=roles	Displays Role search page
Role Categories	/identity/faces/home?tf=role_categories	Displays Role Categories search page

Part IX

Notification Service

This part contains chapters that describe the notification service and callback service.

It contains the following chapters:

34

Developing Notification Events

This chapter describes how to develop notification events. It contains the following sections:

34.1 Notification Concepts

For developing custom notification for various operations in Oracle Identity Manager, the notification engine supports creation of notification events and notification templates.

An event is an operation that occurs in Oracle Identity Manager, such as user creation, request initiation, or any custom event created by the user. The events are generated as part of business operations or via generation of errors. Event definition is the metadata that describes the event. To define metadata for events, it is important to identify all event types supported by a functional component. For example, as a part of the scheduler component, metadata can be defined for scheduled job execution failed and shutting down of the scheduler. Every time a job fails or the scheduler is shut down, the events are raised and notifications associated with that event are sent.

Notification templates are associated to specific events. The templates are used for defining the format of the notification. Oracle Identity Manager provides predefined or default notification templates. In addition, you can create new notification templates.

For some events, Oracle Identity Manager sends notification by default. For example, when a user is created without username and password through UI or reconciliation, the login credentials are notified to the user and user's manager.

You can define new notification events by using notification APIs and resolver class, as described in the subsequent sections. The following are examples of custom notification requirements:

- A user is assigned to a role. The user and role owner are to be notified.
- A user is assigned with a new application instance, which is financially significant, as part of reconciliation. The application instance owner and compliance officer is to be notified for prospective rogue attempts.

34.2 Developing Custom Notification

Developing custom notification is described in the following sections:

- Building the Notification Logic
- Creating Plug-in Pack Containing the Resolver Class
- Building the Invocation Logic
- Configuring the Notification Service

34.2.1 Building the Notification Logic

Building the notification login involves defining the event metadata XML and creating the Resolver class, as described in the following sections:

- Defining Event Metadata
- Creating the Resolver Class
- Creating the plugin.xml File

34.2.1.1 Defining Event Metadata

Corresponding to each event, you must create an XML file that has the specific schema defined by the notification engine. Compliant to that schema (.xsd file), an XML file is created that defines how an event looks like. When the event is defined, you can configure a notification template for that event.

An event file must be compliant with the schema defined by the notification engine, which is NotificationEvent.xsd. The event file contains basic information about the event.

Note: The NotificationEvent.xsd file is in the iam\iam-product\features\notification\metadata directory in the MDS.

The following is a sample event XML file:

```
<?xml version="1.0" encoding="UTF-8"?>
<Events xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="../../../metadata/NotificationEvent.xsd">
  <EventType name="User Created">
    <StaticData>
      <Attribute DataType="X2-Entity" EntityName="User" Name="Granted User"/>
      <Attribute DataType="X2-Entity" EntityName="User" Name="Grantee User"/>
      <Attribute DataType="91-Entity" EntityName="User Group" Name="User Grp"/>
    </StaticData>
      <Resolver class="oracle.iam.notification.DemoResolver">
      <Param DataType="91-Entity" EntityName="Resource" Name="ResourceInfo"/>
    </Resolver>
  </EventType>
</Events>
```

The event XML file has the following elements:

- **EventType name:** The name of the event that will be available while creating notification templates for the event.

- **StaticData:** The list of static parameters. This set of parameters specifically let the user add parameters that are not data dependent. In other words, this element defines the static data to be displayed when notification template is to be configured. For instance, the user entity is not data dependent, and when resolved, has the same set of attributes for all the event instances and notification templates.
- **Param DataType:** The list of dynamic parameters. This set of parameters specifically let the user add parameters that are data dependent. For instance, the Resource entity is data dependent. Corresponding to this field, a lookup is displayed on the UI. When the user selects the resource object, the call goes to the Resolver class provided to get the fields that are shown in the tree from which user can select the attribute to be used on the template.

  Note: Available data is the list of attributes that can be embedded as a token in the template. These tokens are replaced by the value passed by the resolver class at run time. See step 7 of "Creating a Notification Template" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager* for an example of a token.

  Available data is displayed in a drop-down list while creating a notification template, as described in "Creating a Notification Template" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager*.

  Selected data is a single attribute that helps user to copy and paste the attribute name in a notification template. Selected data is the same attribute name as selected in the Available Data list.

  The dynamic entities supported for lookup are user, resource, and organization. These entity names must be specified in the Param DataType element.

  Note: The <Param DataType> element is not a mandatory element. However, when it is used, the entity names must be specified as User, Resource, or Organization.

- **Resolver class:** The Resolver class must be defined for each notification. It defines what parameters are available in the notification creation screen and how those parameters are replaced when the notification is to be sent. In other words, the resolver class resolves the data dynamically at run time and displays the attributes in the UI. See "Creating the Resolver Class" on page 34-4 for information about implementing the resolver class.

Notification service reads the custom event XML files from the META-INF directory of a plug-in.

The recommended way to use the event XML is by placing it in a plugin's META-INF directory. The structure of the custom notification event plug-in is:

- The lib/ directory
  - Notification_Resolver.jar
- The META-INF directory
  - Notification_Event.xml
- plugin.xml

See "Developing Plug-ins" on page 27-1 for detailed information about creating the plug-in JAR and deploying it by using the Plugin Registration Utility.

34.2.1.2 Creating the Resolver Class

All classes have to implement the NotificationEventResolver interface. This interface provides the following methods:

The getAvailableData Method

public List<NotificationAttribute> getAvailableData(String eventType, Map<String, Object> params);

This API returns the list of available data variables. These variables are available on the UI while creating or modifying the templates and allows the user to select the variables so that they can be the part of the messages on the template.

The eventType parameter specifies the event name for which the template is to be read.

The params parameter is the map that has the entity name and the corresponding value for which available data is to be fetched. For instance:

map.put("Resource", "laptop");

This helps you fetch the fields associated with the laptop resource or other data according to the code that you have provided in the resolver class.

Sample code:

```
/**
* this is a dummy implementation and uses hardcoded values
* Implementors need to iterate the XML as found through the event type
* params : will have all the specific values that your resolver needs
* for instance resource name = "laptop" that you may want here to be resolved
through your custon implementation
*/

List<NotificationAttribute> list = new ArrayList<NotificationAttribute>();
NotificationAttribute subatr = new NotificationAttribute();
subatr.setName("Dynamic1"); subatr.setType("91-Entity");
subatr.setEntityName("Resource"); subatr.setRequired(false);
subatr.setSearchable(true); subatr.setSubtree(lookup91EntityMetaData("resource"),
params.get(0)); list.add(subatr);
```

The main tree contains the entity information and the subtree contains all the nodes that are available on the UI. The name field from each node in the subtree is available on the UI for selection.

The getReplacedData Method

HashMap<String, String> getReplacedData(String eventType, Map<String, Object> params);

This API returns the resolved value of the variables present on the template at run time when notification is being sent.

The eventType parameter specifies the event name for which the template is to be read.

The params parameter is the map that has the base values, such as usr_key and obj_key, required by the resolver implementation to resolve the rest of the variables in the template.

Sample code:

```
HashMap<String, Object> resolvedData = new HashMap<String, Object>();
resolvedData.put("shortDate", new Date()); resolvedData.put("longDate", new
Date());
String firstName = getUserFirstname(params.get("usr_key"));
resolvedData.put("fname", firstName); resolvedData.put("lname", "lastname");
resolvedData.put("count", "1 million");
return resolvedData;
```

Example: Creating a Custom Resolver Class

Consider the example of Oracle Identity Manager sending email notification to the user who has been added as a proxy. If the requirement is to change the date format in the notification email, then create a new resolver class file, such as AddProxyResolverModified, for notification while adding a proxy. The following is the code for the AddProxyResolverModified resolver class:

```
package oracle.iam.selfservice.notification;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Date;
import java.util.logging.Level;
import java.text.ParseException;
import java.text.SimpleDateFormat;

import static oracle.iam.identity.utils.Constants.PROXY_START_DATE;
import static oracle.iam.identity.utils.Constants.PROXY_END_DATE;
import static oracle.iam.identity.utils.Constants.PROXY_ORIGINAL_USR_NAME;
import static oracle.iam.identity.utils.Constants.PROXY_ORIG_USER_LOGIN;
import static oracle.iam.identity.utils.Constants.FIRSTNAME;
import static oracle.iam.identity.utils.Constants.LASTNAME;
import oracle.iam.notification.impl.NotificationEventResolver;
import oracle.iam.notification.vo.NotificationAttribute;

public class AddProxyResolverModified implements NotificationEventResolver {
public List<NotificationAttribute> getAvailableData(String eventType, Map<String,
Object> params) throws Exception {
return null;
}

    public HashMap<String, Object> getReplacedData(String eventType, Map<String,
Object> params)throws Exception {

        SimpleDateFormat sdfSource = new SimpleDateFormat("MMMMMMMM DD,yyyy
HH:mm:ss a z");
        SimpleDateFormat sdfDestination = new SimpleDateFormat("EEE, d MMM yyyy
HH:mm:ss Z");
        Date sdate = null;
        Date edate = null;

        HashMap<String, Object> resolvedData = new HashMap<String, Object>();
        resolvedData.put("firstName",params.get(FIRSTNAME));
        resolvedData.put("lastName",params.get(LASTNAME));
        resolvedData.put("originalusername",params.get(PROXY_ORIG_USER_LOGIN));


        String proxy_startDate = (String )params.get(PROXY_START_DATE);
        System.out.println("proxy_startDate : " + proxy_startDate);
```

```
            String proxy_endDate = (String)  params.get(PROXY_END_DATE);
            System.out.println("proxy_endDate : " + proxy_endDate);

            sdate = sdfSource.parse(proxy_startDate);
            edate = sdfSource.parse(proxy_endDate);


            proxy_startDate = sdfDestination.format(sdate);
            System.out.println("proxy_startDate : " + proxy_startDate);
            proxy_endDate = sdfDestination.format(edate);
            System.out.println("proxy_endDate : " + proxy_endDate);


            resolvedData.put("proxystartdate", proxy_startDate);
            resolvedData.put("proxyenddate", proxy_endDate);
            return resolvedData;
        }
}
```

34.2.1.3 Creating the plugin.xml File

The plugin.xml file is a standard XML file used by the plug-in framework. If any feature uses plug-ins, then it exposes a plug-in point. This XML has the information of plug-in point.

For notification event and resolver plug-in, the exposed plug-in point is:

```
oracle.iam.notification.impl.NotificationEventResolver
```

Sample plugin.xml for Notification event and resolver is:

```
<?xml version="1.0" encoding="UTF-8" ?> <oimplugins
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <plugins pluginpoint="oracle.iam.notification.impl.NotificationEventResolver">
<plugin pluginclass="gov.hhs.notification.resolver.SendChallengeQuestionsResolver"
version="1.0" name="Challenge Question Resolver" />   </plugins>
</oimplugins>
```

34.2.2 Creating Plug-in Pack Containing the Resolver Class

After creating the Resolver class, you must package it into a plug-in JAR file, and deploy the JAR file by using the Plug-in Registration Utility. See "Developing Plug-ins" on page 27-1 for detailed information about creating the plug-in JAR and deploying it by using the Plugin Registration Utility.

34.2.3 Building the Invocation Logic

After building the notification logic by defining event metadata XML and creating the Resolver class, you can call the notification logic at a specific operation in Oracle Identity Manager. This is achieved by using event handlers.

The invocation logic for the notification is built as a custom event handler. The custom event handler is then configured at the right stage in the relevant operation. The custom event handler is then deployed by using the Plugin Registration Utility. See "Developing Event Handlers" on page 28-1 for details about developing event handlers.

34.2.4 Configuring the Notification Service

You have the following options for creating the infrastructure-level configuration for notification:

- **Notification Configuration in Oracle Identity Manager:** In Oracle Identity Manager, notification configurations are handled via notification providers. UMS is the default notification provider. For information about notification providers, see "Managing Notification Providers" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager*.
- **Notification Configuration in BPEL workflow:** SOA exposes notification service, which can be called in BPEL workflow for notification. For information about SOA email notification, see "Configuring SOA Email Notification" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager*.

Note: You can configure e-mail definitions for using in provisioning processes by using the Email Definition Form in the Design Console. See "Email Definition Form" on page 5-1 for details.

35

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:

35.1 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 35-8 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 35-8

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

35.1.1 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 POCreater 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.

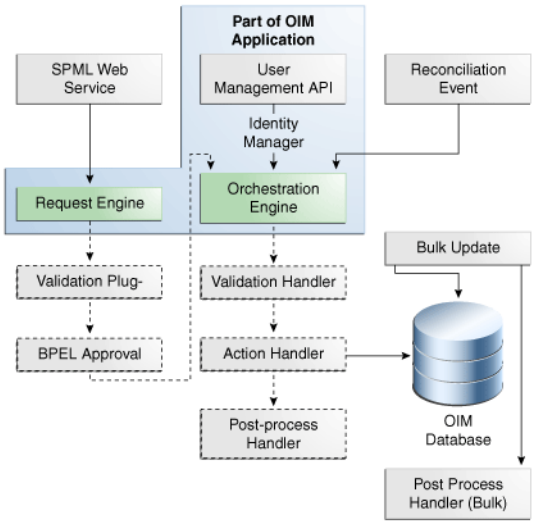
      Note: If one portion of the request is rejected, then the entire request is rejected. Therefore, in this example, although the Buyer role does not have an SoD violation, it is not provisioned to the user.

   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.

35.1.2 Understanding Event Processing

Figure 35–1 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 35–1 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.

35.1.3 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.

35.2 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 "SPML Attributes and LDAP Mappings, and Oracle Identity Manage Attributes" on page 32-16 for details.

Mapped attribute names are used in messages sent as well as in callback configuration (particularly, in the ConstraintAttribute parameter). Table 35–1 defines how Oracle Identity Manager user type attributes are represented in callbacks using SPML PSOs.

Table 35–1 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
preferredLangauage	Language
state	State
street	Street
surname	Last Name
telephoneNumber	Telephone Number
title	Title
userId	usr_key
userName	User Login

Table 35–2 defines how Oracle Identity Manager role type attributes are represented in callbacks using SPML PSOs.

Table 35–2 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.

35.3 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 35–3 summarize the user and role events for which Oracle Identity Manager makes callbacks and the information returned with the callback.

Table 35–3 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.	▪ Target Type ▪ Target GUID ▪ Operation ▪ Request ID ▪ Provisioning Service Object (PSO) with created attributes and values (except password and challenge questions) **Note:** PSO is used for create operations. Modification objects are used for modify operations. ▪ Roles assigned to the user	▪ OIM Request Status ▪ Target Type ▪ Target GUID ▪ Operation ▪ Request ID
User Modify	All sources: third party requests, the console and through a reconciliation event.	▪ Target Type ▪ Target GUID ▪ Operation ▪ Request ID ▪ Modification object* with modified attributes and values (except password and challenge questions)	▪ OIM Request Status ▪ Target Type ▪ Target GUID ▪ Operation ▪ Request ID

Table 35–3 (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.	■ Target Type ■ Target GUID ■ Operation ■ Request ID	■ 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.	■ Target Type ■ Target GUID ■ Operation ■ Request ID	■ 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.	■ Target Type ■ Target GUID ■ Operation ■ Request ID	■ 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.	■ Target Type ■ Target GUID ■ Operation ■ Request ID ■ Modification object* with GUIDs of assigned roles	■ 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.	■ Target Type ■ Target GUID ■ Operation ■ Request ID ■ Modification object* with GUIDs of assigned roles	■ 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.	■ Target Type ■ Target GUID ■ Operation ■ Request ID ■ PSO* with created attributes and values	■ OIM Request Status ■ Target Type ■ Target GUID ■ Operation ■ Request ID
Role Modify	All sources: third party requests, the console and through a reconciliation event.	■ Target Type ■ Target GUID ■ Operation ■ Request ID ■ Modification object* with modified attributes and values	■ OIM Request Status ■ Target Type ■ Target GUID ■ Operation ■ Request ID

Table 35–3 (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.	■ Target Type ■ Target GUID ■ Operation ■ Request ID	■ OIM Request Status ■ Target Type ■ Target GUID ■ Operation ■ Request ID

35.4 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
- Adding the OIM.DefaultTenantGUID System Property

35.4.1 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 35-4 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.

Note: ConstraintAttribute and ConstraintAttributeValue are not multi-valued attributes in the callback policies. Therefore, you can provide only one constraintAttribute and constraintAttributeValue attribute for each callback policy. If you specify multiple constraintAttribute and constraintAttributeValue in a single callback policy, then the callback will be triggered even if all the constraints do not match.

- **Tenant GUID:** Indicates the tenant GUID to which a callback policy is applicable in the multi-tenant (MT) mode of Oracle Identity Manager. The callback URLs for different policies in the MT mode are different, and therefore, the tenant GUID must be specified to uniquely identify the tenant to which the callback is to be sent. An example of the usage of this tag is <tenantGUID>202</tenantGUID>. This tag is not used in the non-MT mode.
- **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 35-4 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 35-4 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 35-4 for more information. The default value is LDAP GUID.

Example 35–1 is a sample configuration file.

Example 35–1 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/
   PostProcessingPluginRequestPortImplTest</stepName>
<description>Webservice url for this policy</description>
</asyncSteps>
</postProcessing>
</provisioningSteps>
</policy>
</callbackConfiguration>
```

35.4.2 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 Oracle Enterprise Manager.

   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.

35.4.3 Adding the OIM.DefaultTenantGUID System Property

In non-MT mode of Oracle Identity Manager, the OIM.DefaultTenantGUID system property must be set with a value that works as a tenant GUID for applications that expect a value for the tenant GUID. A sample value for this is 201. Oracle Identity Manager will not send any callback if this property is not set. If the application receiving the callbacks does not expect any specific tenant GUID, then provide any random value.

See Table 5-2: Nondefault System Properties in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager* for information about this system property.

35.5 Troubleshooting the Callback Service

Table 35–4 lists the troubleshooting steps that you can perform if you encounter callback service errors:

Table 35–4 Troubleshooting Callback Service

Problem	Solution
Not able to submit SPML request (Failed Authentication).	Make sure that the SPML request is submitted with a valid SPML user (member of SPML_App_Role group) with its correct credentials. If the request is submitted from client APIs, then note that compatible client policy must be applied. The following is the sample eror response displayed when a SPML request is submitted with incorrect credentials: `<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>`

Table 35–4 (Cont.) Troubleshooting Callback Service

Problem	Solution
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.	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. PolicyName matching is ignored when callbackOnly attribute is set to false. 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. All the callback Web service URLs present in that policy are triggered when the entity type and operation condition is also met.
The policy reference URI 'oracle/wss_saml_or_username_token_service_policy' is not valid.	Make sure that WSM Policy Manager is deployed and targeted to the interacting servers such as Oracle Identity Manager and SPML request starting server. In addition, make sure that WSM Policy Manager is in active mode and is ready for serving the requests.
Not sure what is SPML APPID and Oracle Identity Manager APPID, and where these APPIDs are to be created.	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. 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. 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. SPML repository or LDAP contains both SPML APPID and Oracle Identity Manager APPID. 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. 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 again LDAP by checking the Oracle Identity Manager APPID in LDAP repository. Therefore, LDAP contains Oracle Identity Manager APPID in it.

Part X

Customization Lifecycle

This part describes how to use the customization utilities provided by Oracle Identity Manager and migrate the customization from one deployment to another.

It contains the following chapters:

36

Understanding Customization Types

All customizations in Oracle Identity Manager are managed by using the following types of utilities:

- **Deployment and undeployment utilities:** The following types of utilities can be used for deployment and undeployment purposes:
  - **Database-specific utilities:** These are utilities in Oracle Identity Manager that have the capability to take the input customization and insert it in the relevant database table. For example, the Plug-in Registration Utility takes the plugin pack and inserts it into the Plugin table in Oracle Identity Manager database. See "Developing Plug-ins" on page 27-1 for information about the Plug-in Registration Utility.
  - **SOA ANT- based utilities:** There are some ANT-based utilities from SOA suite for deploying or undeploying a SOA composite or approval workflow. Refer to SOA documentation for information about SOA ANT- based utilities.
  - **OPSS policy migration tool:** This tool is used for migrating Oracle Identity Manager authorization OES policies that you have changed based on some advanced authorization customizations.
  - **Oracle Enterprise manager:** The Enterprise Manager (EM) is the Fusion Middleware diagnostic and monitoring application. Using EM, you can deploy or undeploy customizations in Meta Data Store (MDS) and SOA composites or approval workflows.

    You can use EM to insert data in to the MDS. For example, you can use the EM to insert the LDAP synchronization user attribute mapping XML into MDS. See "Deploying and Undeploying Customizations" on page 37-1 for information about exporting and importing metadata files to and from MDS.
  - **IDE:** Integrated Development Environment (IDE), such as Jdeveloper, provides ways to deploy SOA composites or approval workflows.
- **Test to Production utilities:** This refer to the utilities used to migrate the customizations from an Oracle Identity Manager deployment to another. For example, you can customize Oracle Identity Manager in a test environment, and then migrate the customizations to the production environment. This is done by using the Deployment Manager utility, as described in "Migrating Configurations and Customizations" on page 38-1.

  Another example of Test to Production utility is the Sandbox, which is used to migrate UI customizations from one deployment to another. See "Managing Sandboxes" on page 30-4 for information about the Sandbox.

Table 36–1 lists the Oracle Identity Manager artifacts that can be migrated via the deployment/undeployment and Test to Production utilities:

Table 36–1 Oracle Identity Manager Artifacts and Type of Utilities

Component	Artifacts for Deployment/Undeployment	Artifacts for Test to Production and Deployment/Undeployment
Identity Administration	Enterprise roles	Entitlements
	Role category	Catalog items
	Organizations	Scheduled tasks (as Oracle Identity Manager plug-ins)
	Access policies	SOA composites (workflows)
	Request profiles	Generic Technology Connector (GTC) providers
	Lookup code	Event handlers (as Oracle Identity Manager plug-in)
	Attestation processes	Resource bundle
	OES policies	Oracle Identity Manager plug-ins
	User defined fields (UDFs) for user, role, organization, catalog, and resource	Provisioning customizations (JAR files)
	Scheduled jobs	Notification events (as Oracle Identity Manager plug-in)
	IT resource types	Callback policy
	IT resources	UI customizations
	Application instances	Third-party JAR files
	Resource objects	Custom JAAS login modules
	Process forms	
	Reconciliation profile	
	Provisioning workflows and process task adapters	
	Data object definitions	
	Rules	
	OBR rules	
	Notification templates	
	Error codes	
	System properties	
	E-mail definitions	
	UMS-based notification (Oracle Identity Manager artifact)	
	UMS-based notification (EM artifact)	
	Password policies	
	GTC	

Table 36–1 (Cont.) Oracle Identity Manager Artifacts and Type of Utilities

Component	Artifacts for Deployment/Undeployment	Artifacts for Test to Production and Deployment/Undeployment
	Approval policies	
	Adapters	
	Web Services Security Configurations	
	Artifacts from connector on Oracle Identity Manager server	
	Artifacts in Remote Manager	
	Scripts/executables associated with connectors	
	Changes to OIMConfig.xml	
	Custom diagnostic tests added to diagnostic framework	
	Reports	
	HTTPS configurations in Oracle Identity Manager	
	Entries in CSF files	
	SSL configuration between Oracle Identity Manager and Remote Manager	
	Secure cookies	
LDAP Synchronization	OVD adapters	LDAP Container rules
	Configurations in Oracle Identity Manager, OVD, and LDAP by idmConfigTool.sh	LDAP reconciliation profiles
		LDAP attribute mappings
Web Access Management	All artifacts for LDAP synchronization	
	Configurations done by idmConfigTool.sh	
	Oracle Identity Manager WLS/WAS tier - IA Provider and other configurations	
Identity Analytics/Compliance	Oracle Identity Manager stored procedure changes	
	OIA configurations - Oracle Identity Manager JAR files, XML file changes, provisioning server connection	
Segregation of Duties (SoD)	Catalog items	SIL configuration
		SIL registration artifacts
		SIL provider - JAR
		SIL provider - XML

37

Deploying and Undeploying Customizations

This chapter contains the following topics:

- Migrating User Modifiable Metadata Files
- Migrating JARs and Resource Bundle

37.1 Migrating User Modifiable Metadata Files

The user modifiable metadata XML files can be exported to MDS, imported from MDS, and deleted from MDS by using Oracle Enterprise Manager.

This section contains the following topics:

- Exporting Metadata Files to MDS
- Importing Metadata Files from MDS
- Deleting Metadata Files from MDS
- User Modifiable Metadata Files
- Creating MDS Backup
- Exporting All MDS Data for Oracle Identity Manager

37.1.1 Exporting Metadata Files to MDS

To export metadata XML files to MDS:

1. Login to Oracle Enterprise Manager as the administrator user by navigating to the URL in the following format:

   http://*ADMINSTRATION_SERVER*/em

   Make sure that the Administrative Server and at least one Oracle Identity Manager Managed Server are running.

2. Navigate to **Identity and Access**, **oim**. Right-click and navigate to **System MBean Browser**.

3. Under Application Defined MBeans, navigate to **oracle.mds.lcm**, **Server:oim_server1**, **Application:OIMMetedata**, **MDSAppRuntime**.

4. Export metadata by using the operations. To do so:

   a. Select and open the first exportMetadata operation in the list.

b. For toLocation, provide the path to a temporary directory, in which this file is to be exported. This file will be exported to the computer on which Oracle Identity Manager is running. Therefore, make sure that the directory path you specify exist on that computer.

c. For docs, click the pencil icon, click **Add**, and in the Element box, provide the full path of the file to be exported. By clicking **Add**, you can provide the path to multiple docs. Click **OK** at the bottom after adding the metadata docs to be exported.

d. Invoke the operation.

37.1.2 Importing Metadata Files from MDS

To import metadata XML files from MDS:

1. Login to Oracle Enterprise Manager as the admin user. Make sure that the Administrative Server and at least one Oracle Identity Manager Managed Server are running.
2. Navigate to **Identity and Access**, **oim**, **oim(*VERSION*)**. Right-click and navigate to **System MBean Browser**.
3. Under Application Defined MBeans, navigate to **oracle.mds.lcm**, **Server:oim_server1**, **Application:OIMMetedata**, **MDSAppRuntime**.
4. Import metadata by using the operations. To do so:
   a. In the Operations tab, select the first importMetadata operation in the list.
   b. For fromLocation, provide the directory path of the Oracle Identity Manager host from where documents are to be imported.
   c. For docs, click the pencil icon, click **Add**, and in the Element box, provide the full path of the file to be imported. By clicking **Add**, you can provide the path to multiple docs. If no value is provided, then it imports everything under the fromLocation directory recursively.
   d. Invoke the operation.

37.1.3 Deleting Metadata Files from MDS

To delete metadata XML files from MDS.

1. Navigate to MDS runtime mbeans, as described in step 1 of "Exporting Metadata Files to MDS" on page 37-1.
2. Delete metadata by using the operations. To do so:
   a. In the Operations tab, select the first deleteMetadata operation in the list.
   b. For docs, click the pencil icon, click **Add**, and in the Element box, provide the full path of the file to be deleted. By clicking **Add**, you can provide the path to multiple docs to be deleted.
   c. Invoke the operation.

37.1.4 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, /db, or /custom. Make sure that starting path or directory name for any XML document is /custom.

```
/db/LDAPContainerRules.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-spmlws/EventHandlers.xml
/db/ssointg/EventHandlers.xml
/metadata/iam-features-catalog/EventHandlers.xml
/metadata/iam-features-provisioning/event-definition/EventHandlers.xml
/metadata/iam-features-requestprofile/event-definition/EventHandlers.xml
/metadata/iam-features-rolesod/EventHandlers.xml
```

37.1.5 Creating MDS Backup

You might need to create a backup of the MDS before performing customizations. To create a backup of the MDS by using Oracle Enterprise Manager:

1. Login to Oracle Enterprise Manager as the administrator.
2. On the landing page, click **oracle.iam.console.identity.self-service.ear(V2.0)**.
3. From the Application Deployment menu at the top, select **MDS configuration**.
4. Under Export, select the **Export metadata documents to an archive on the machine where this web browser is running** option, and then click **Export**.

   All the metadata is exported in a ZIP file.

37.1.6 Exporting All MDS Data for Oracle Identity Manager

Some configurations for Oracle Identity Manager are stored in an MDS repository rather than on a file system on the Oracle Identity Manager Server. Troubleshooting configuration issues can sometimes require exporting all MDS data in order to examine it and make corrections if required.

To export all of the Oracle Identity Manager metadata contained in the MDS repository:

1. Setup the environment as a prerequisite:
   a. To perform MDS operations, log in to the Oracle Identity Manager server host with the account used to install and run WebLogic Application Server.
   b. Set you environment variables for the Oracle Identity Manager domain by running the appropriate `setDomainEnv` script found in the *MIDDLEWAR_HOME*/user_projects/domains/*DOMAIN_NAME*/bin/ directory. The command is as shown:

      ```
      $ cd MIDDLEWARE_HOME/user_projects/domains/OIMDomain/bin
      $ .setDomainEnv.sh
      ```

   c. Create a temporary directory, such as /tmp/OIM/MDSData/, which will be used to store the resulting XML files from the database.
   d. Verify that the application server is up and running.
   e. Ensure that you know the WebLogic administrator username and the URL to the Admin Server.
2. Perform the export, as follows:
   a. In the command shell or console window, go to the *OIM_ORACLE_HOME*/common/bin/ directory.
   b. Run the `wlst.sh` command, and then run the `connect()` command, as shown:

      ```
      $ ./wlst.sh
      CLASSPATH=/opt/oracle/Middleware/wlserver_
      10.3/server/ext/jdbc/oracle/11g/ojdbc6dms.jar:...
      ...
      Your environment has been set.
      ...
      Initializing WebLogic Scripting Tool (WLST) ...

      Welcome to WebLogic Server Administration Scripting Shell

      Type help() for help on available commands
      wls:/offline> connect()
      Please enter your username [weblogic] :
      Please enter your password [welcome1] :
      Please enter your server URL [t3://localhost:port] :
      Connecting to t3://localhost:port with userid weblogic ...
      Successfully connected to Admin Server 'AdminServer' that belongs to domain
      'OIMDomain'.

      Warning: An insecure protocol was used to connect to the server. To ensure
      on-the-wire security, the SSL port or Admin port should be used instead.
      ```

   c. Provide the WebLogic administrator username and password and the URL to the Admin Server.

d. Run the `exportMetadata` command providing at least the `application`, `server`, and `toLocation` arguments, as shown:

Note: Be sure to pass the argument data in single quotes, such as:

```
server='oim_server1'
```

```
wls:/OIMDomain/serverConfig> exportMetadata(application='OIMMetadata',
server='oim_server1', toLocation='/tmp/OIM/MDSData')
```

e. A list of the files exported is displayed. At this point, you can run the `disconnect()` command followed by the `exit()` command, as shown:

```
wls:/OIMDomain/serverConfig> disconnect()
Disconnected from weblogic server: AdminServer
wls:/offline> exit()


Exiting WebLogic Scripting Tool.

$
```

f. Go to the /tmp/OIM/MDSData/ directory, and view the db/oim-config.xml file, or the db/form-metadata/FormMetaData.xml file, or any other exported MDS file.

The following is an example WLST script for exporting all MDS files:

```
connect('WEBLOGIC_USERNAME','PASSWORD','t3://localhost:PORT')
exportMetadata(application='OIMMetadata', server='oim_server1',
toLocation='/tmp/OIM/MDSData')
disconnect()
exit()
```

You can save this script in a .py file, for example /tmp/exportOIMMDS.py, which you can run to automatically produce the same results. The following is a sample .py file:

```
cd MIDDLEWARE_HOME/user_projects/domains/OIMDomain/bin
. setDomainEnv.sh
mkdir -p /tmp/OIM/MDSData
cd $OIM_ORACLE_HOME/common/bin
./wlst.sh /tmp/exportOIMMDS.py
```

37.2 Migrating JARs and Resource Bundle

When migrating from test to production environment, all the connector artifacts must be migrated to the respective database tables, which can be done by using the following utilities to migrate JAR files and resource bundle:

- 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 wlfullclient.jar is generated before running these utilities.
- Set *APP_SERVER*, *OIM_ORACLE_HOME*, *JAVA_HOME*, *MW_HOME*, *WL_HOME*, and *DOMAIN_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]
  ```

  For information about configuring the utilities to upload/download JAR files and resource bundle over SSL, see "Configuring SSL for Oracle Identity Manager Utilities" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager*.

  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_JAVA_TASK_JAR] [-ScheduleTask LOCATION_OF_SCHEDULED_TASK_
  JAR] [-ThirdParty LOCATION_OF_THIRD_PARTY_JAR] [-ICFBundle
  LOCATION_OF_ICF_BUNDLE_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
  ```

- After running the UploadResourceBundles.sh script, you must purge Oracle Identity Manager cache or restart Oracle Identity Manager to load the new objects from MDS into Oracle Identity Manager cache. If Oracle Identity Manager cache is not refreshed, then Oracle Identity Manager reads the contents from the local resource bundle files.

37.2.1 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 Xellerate admin username :]ADMISTRATOR_LOGIN
[Enter the admin password :]ADMINISTRATOR_PASSWORD
[[Enter serverURL (Ex. t3://oimhostname:oimportno for weblogic or
corbaloc:iiop:localhost:2801 for websphere)]:]t3://xyz.com:14000
[[Enter context (Ex. weblogic.jndi.WLInitialContextFactory for weblogic or
com.ibm.websphere.naming.WsnInitialContextFactory for
websphere)]:]weblogic.jndi.WLInitialContextFactory
Enter the jar type
 1.JavaTasks
 2.ScheduleTask
 3.ThirdParty
 4.ICFBundle
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.

37.2.2 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 Xellerate admin username :]ADMINISTRATOR_LOGIN
[Enter the admin password :]ADMINISTRATOR_PASSWORD
[[Enter serverURL (Ex. t3://oimhostname:oimport for weblogic or
corbaloc:iiop:localhost:2801 for websphere)]:]t3://localhost:14000
[[Enter context (i.e.: weblogic.jndi.WLInitialContextFactory for weblogic or
com.ibm.websphere.naming.WsnInitialContextFactory for
websphere)]:]weblogic.jndi.WLInitialContextFactory
Enter the jar type
1.JavaTasks
2.ScheduleTask
3.ThirdParty
4.ICFBundle
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.

37.2.3 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 Xellerate admin username :]ADMINISTRATOR_LOGIN
[Enter the admin password :]ADMINISTRATOR_PASSWORD
[[Enter serverURL (Ex. t3://oimhostname:oimport for weblogic or
corbaloc:iiop:localhost:2801 for websphere)]:]t3://localhost:14000
[[Enter context (i.e.: weblogic.jndi.WLInitialContextFactory for weblogic or
com.ibm.websphere.naming.WsnInitialContextFactory for
websphere)]:]weblogic.jndi.WLInitialContextFactory
Enter the jar type
1.JavaTasks
2.ScheduleTask
3.ThirdParty
4.ICFBundle
1
Enter the name of jar to be deleted from DB :
example.jar
Do u want to delete more jars [y/n] :n
```

37.2.4 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 Xellerate admin username :]ADMINISTRATOR_LOGIN
[Enter the admin password :]ADMINISTRATOR_PASSWORD
[[Enter serverURL (Ex. t3://oimhostname:oimportno for weblogic or
corbaloc:iiop:localhost:2801 for websphere)]:]t3://localhost:14000
[[Enter context (i.e.: weblogic.jndi.WLInitialContextFactory for weblogic or
com.ibm.websphere.naming.WsnInitialContextFactory for
websphere)]:]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
```

37.2.5 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 Xellerate admin username :]ADMINISTRATOR_LOGIN
[Enter the admin password :]ADMINISTRATOR_PASSWORD
[[Enter serverURL (Ex. t3://oimhostname:oimportno for weblogic or
corbaloc:iiop:localhost:2801 for websphere)]:]t3://localhost:14000
[[Enter context (i.e.: weblogic.jndi.WLInitialContextFactory for weblogic or
com.ibm.websphere.naming.WsnInitialContextFactory for
```

```
websphere)]:]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
```

37.2.6 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 Xellerate admin username :]ADMINISTRATOR_LOGIN
[Enter the admin password :]ADMINISTRATOR_PASSWORD
[[Enter serverURL (Ex. t3://oimhostname:oimportno for weblogic or
corbaloc:iiop:localhost:2801 for websphere)]:]t3://localhost:14000
[[Enter context (i.e.: weblogic.jndi.WLInitialContextFactory for weblogic or
com.ibm.websphere.naming.WsnInitialContextFactory for
websphere)]:]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
```

38

Migrating Configurations and Customizations

Configurations and customizations in Oracle Identity Manager can be migrated from one deployment to another deployment. For example, you might want to migrate the configurations and customizations from a test environment to a production environment. This is referred to as Test to Production (T2P).

T2P can be performed in the following ways:

- **Incremental T2P:** In this type of T2P, you use the Deployment Manager tool for exporting and importing Oracle Identity Manager configurations and customizations. The Deployment Manager lets you export the objects that constitute the Oracle Identity Manager configuration. See "Using the Deployment Manager" on page 38-1 for more information.
- **Full T2P:** Fusion Middleware Framework-based movement scripts are used for this type of T2P. These scripts are used to move all the properties of an environment to another environment without the environment-specific attributes, which can be reconfigured. See "Moving from a Test to a New Production Environment Using Movement Scripts" on page 38-16 for more information.

  Note: Movement scripts support only Oracle WebLogic Application Server, and full T2P of Oracle Identity Manager on IBM WebSphere Application Server is not supported.

- **Policy migration:** You can migrate the policies from one Oracle Identity Manager deployment to another, for example, from a test environment to production environment, by using a command-line utility provided by Oracle Platform Security. See "Migrating the Policies" on page 38-22 for more information.

This chapter contains the following topics:

38.1 Using the Deployment Manager

The Deployment Manager is a tool for exporting and importing Oracle Identity Manager configurations and customizations. The Deployment Manager lets you export the objects that constitute the Oracle Identity Manager configuration. Usually, you use the Deployment Manager to migrate a configuration from one deployment to

another, for example, from a test to a production deployment, or to create a backup of your system.

Important:

- To use Deployment Manager, JRE 1.4.2 or a higher version must be installed on any computer that is running the Oracle Identity System Administration.
- To use the Deployment Manager with the latest Java 7 update 51, update the security information in the Java Console. To do so, open the Java Console, click the **Security** tab, and add the site name. The site name is in the following format:

  http://*HOST*:*PORT*/xlWebApp/DeploymentManager/loadDU.do

You can save some or all of the objects in your configuration. This lets you develop and test your configurations in a test environment, and then import the tested objects into your production environment. You can export and import an object and all of its dependent and related objects at the same time. Alternatively, you can export and import each object individually.

The Deployment Manager allows you to retrieve configuration information and binary data from the source system, store the information in an XML file, and then import the information from the XML file to the target system. The binary data includes plug-ins, JARs, and custom resource bundles. The Deployment Manager allows you to import data from the Oracle Identity Manager database, Meta Data Store (MDS) repository, or API repository. As a result, you can import all types of objects from these repositories, such as system properties, jobs, and scheduled tasks, which are not in the same repository. For example, you can import the scheduled tasks that are in the MDS repository instead of the database.

An object exported from one type of repository is imported to the same type of repository. For example, if a scheduled task is exported from the MDS repository, then the scheduled task is imported to the same repository, which is MDS, in the target system.

Note: In addition to the Deployment Manager, you can use the sandbox feature to migrate configurations and customizations from one deployment to another. See "Managing Sandboxes" on page 30-4 for information about working with sandbox.

This section includes the following topics:

38.1.1 Features of the Deployment Manager

The Deployment Manager helps you to migrate Oracle Identity Manager deployments from one server environment to another, such as from a testing environment to a staging environment, or from a staging environment to a production environment.

The Deployment Manager enables you to:

- Update individual components of a deployment in different test environments
- Identify objects associated with components to be exported, so that those resources can be included
- Provide information about exported files
- Add comments

The Deployment Manager handles the following types of information:

- Application instances
- Catalog definitions
- Plug-ins
- JAR files
- Custom resource bundles
- Roles
- Organizations
- Access policies
- Attestation processes
- User metadata
- Role metadata
- Organization metadata
- Scheduled tasks
- Scheduled jobs
- IT resources
- Resource objects
- Lookup definitions
- Process forms
- Provisioning workflows and process task adapters
- Data object definitions
- Rules
- Notification templates
- Generic Technology Connectors (GTC)
- GTC providers
- Error codes
- System properties
- E-mail definitions

- Password policies
- IT resource definition
- Request datasets
- Approval policies
- Event handlers
- Prepopulation adapters
- Process definitions
- Risk configuration
- Certification definition

> **Note:** On the source, certification definition that is being exported might contain references to specific users, roles, application instances, entitlements, or organizations. These specific references are scrubbed while exporting the certification definition and then importing it on the target setup. On the target, the definition must be opened from the certification definition management UI and updated for selection of these entities on the target. The definitions cannot be used unless they are updated and will result in errors if used as it is. Any definition that is generic and do not contain specific references can be used as it is after importing.

- Certification configuration

The following are limitations of the Deployment Manager:

- **Merge Utility**: The Deployment Manager is not a merge utility.

  It cannot handle modifications done in both production and test environments. It replaces the object in the target system with that in the XML file.

- **Version Control Utility**: The Deployment Manager does not track versions of imported files, and does not provide rollback functionality.

  You can only use it as a means to move data between environments.

38.1.2 Exporting Deployments

You can export objects from your Oracle Identity Manager system and save them in an XML file. The Deployment Manager has an Export Wizard that lets you create your export file. Add objects by type, one type at a time, for example, roles, then forms, then processes, and so on.

> **Note:** Application instances are exported and imported without the datasets. Datasets are migrated as a part of UI customization.

If you select an object that has child objects or dependencies, you have the option to add them or not. After adding objects of one type, you can go back and add other objects to your XML files. When you have all the objects you want, the Deployment Manager saves them all at once in a single XML file.

Note: When user-defined fields are associated with a specific resource object, during the export process one of the following events can occur:

- If the user-defined fields contain values (entered information), then the Deployment Manager will consider them to be dependencies.
- If the user-defined fields contain no values (the fields are blank), then the Deployment Manager will not consider them to be dependencies.

To export a deployment:

1. Login to Oracle Identity System Administration.
2. In the left pane, under System Management, click **Export**. The Deployment Manager opens and the Search Objects page of the Export Wizard is displayed.

   Note: To open the Deployment Manager by using Mozilla Firefox Web browser, an additional authentication dialog box might be displayed. Providing authentication in this dialog box allows access to the Deployment Manager. To avoid this additional authentication:

   1. In Mozilla Firefox Web browser, 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 Deployment Manager is opened by using Microsoft Internet Explorer, Google Chrome, and Apple Safari Web browsers.

3. On the Search Objects page, select an object type from the menu, and enter search criteria. If you leave the criteria field blank, an asterisk (*) is displayed automatically to find all the objects of the selected type.

   All the objects supported by Deployment Manager for migration are available for exporting. See "Features of the Deployment Manager" on page 38-3 for the list of objects supported by Deployment Manager for migration.

4. Click **Search** to find objects of the selected type.

   To select an object, select the option of the object.

5. Click **Select Children**.

   The Select Children page is displayed with the selected objects and all of their child objects.

6. Select the child objects that you want to export.

   To select or remove an item, select the appropriate option.

   Click **Back** to go to the Search Objects page.

7. Click **Select Dependencies**.

The Select Dependencies page is displayed with any objects required by the selected objects.

8. Select the dependent objects that you want to export.

   To select or remove an item, select the option of the item.

   Click **Back** to go to the Select Children page.

9. Click **Confirmation**.

   The Confirmation page is displayed.

10. Ensure that all the required items are selected, then click **Add for Export**.

    After you click **Add for Export**, you can still add more items to this export file.

    Select **Add More** and click **OK** to go to Search Objects Page to add more objects for export.

11. Use the wizard to add more items, or finish and exit the wizard. Select the appropriate option and click **OK**.

    If you select **Add more**, repeat Steps 3 through 10. Otherwise, the Export page is displayed.

    The Export page displays your current selections for export. Your selections have icons next to them that indicate what types of objects are selected. The Summary information pane shows the objects you are exporting. The Unselected Dependencies pane displays the list of dependent or child objects that you did not select for export.

12. Make any adjustments to your export file as follows:

    - Click **Reset** to clear the form.
    - Click **Legend** to see icon definitions.
    - Click **Add Objects** to restart the wizard and add more items to your export file.

    To remove an object from the Current Selections list:

    - Right-click the object to remove and select **Remove** from the shortcut menu. If the object has child objects, then select **Remove including children** from the shortcut menu to remove the child objects all at the same time.
    - Click **Remove** to confirm. If the object is a child or dependency of a selected item, then it is added to the Unselected Children or Unselected Dependencies list.

    To add an object back to the Current Selections list from the Unselected Children or Unselected Dependencies list,

    a. Right-click the object, and select **Add**.

    b. Click **Confirmation**.

       The Confirmation page is displayed.

    c. Click **Add for Export**.

13. Click **Export**.

    The Add Description dialog box is displayed.

14. Enter a description for the file.

    This description is displayed when the file is imported.

15. Click **Export**.

    The Save As dialog box is displayed.

16. Enter a file name.

    You can browse to find a location.

17. Click **Save**.

    The Export Success dialog box is displayed.

18. Click **Close**.

38.1.3 Importing Deployments

Objects that were exported into an XML file by using the Deployment Manager can be imported into Oracle Identity Manager by using the Deployment Manager. You can import all or part of the XML file, and you can import multiple XML files at once. The Deployment Manager ensures that the dependencies for any objects you are importing are available, either in the import or in your system. During an import, you can substitute an object you are importing for one in your system. For example, you can substitute a group specified in the XML file for a group in your system.

Note:

- If a user belongs to a group to which the Import menu item has been assigned, then that user must also have the necessary permissions for the objects that the user wants to import. Without these object-specific permissions, the Import operation fails. The user must be a Deployment Manager Administrator to be able to see Deployment Manager menu items on the UI based on menu permissioning model.
- When more than 1000 resources, process definitions, parent forms, child forms, access policies, roles, and rules are imported by using the Deployment Manager, the size of the EIF table increases. The data can be truncated from this table by running a simple SQL query such as Delete from EIF.

To import an XML file:

Note: Before importing data that contains references to menu items, you must first create the menu items in the target system.

1. Login to Oracle Identity System Administration.
2. In the left pane, under System Management, click **Import**. The Deployment Manager opens.

   If another import from any other session is in progress, then a dialog box is displayed stating that the Deployment Manager import utility is currently used by another user. If there is a lock, then contact other developers because the utility can only be used by one user at a time. After all other users have released the lock, click **Get Lock** to start the import process.

Note: To open the Deployment Manager by using Mozilla Firefox Web browser, an additional authentication dialog box might be displayed. Providing authentication in this dialog box allows access to the Deployment Manager. 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 Deployment Manager is opened by using Microsoft Internet Explorer, Google Chrome, and Apple Safari Web browsers.

3. Select a file.

   The Import dialog box is displayed.

4. Click **Open**.

   The File Preview page is displayed.

5. Click **Add File**.

   The Substitutions page is displayed

6. To substitute a name, click the **New Name** field adjacent to the item you want to replace, and enter the name.

   You can substitute only items that exist in the target system.

7. Click **Next**. If you are exporting an IT resource instance, then the Provide IT Resource Instance Data page is displayed. Otherwise, you are redirected to the Confirmation page.

8. Modify the values in the current resource instance and click **Next**, or click **Skip** to skip the current resource instance, or click **New Instance** to create a new resource instance.

   The Confirmation page is displayed.

9. Confirm that the information displayed on the Confirmation page is correct.

   To go back and make changes, click **Back**, or click **View Selections**.

   The Deployment Manager Import page displays your current selections.

   The Import page also displays icons next to your current selections. The icons indicate what types of objects are selected. The icons on the right indicate the status of your selections. The file names of any selected files, summary information about the objects you are importing, and substitution information are displayed on the left side of the page. On the right, the **Objects Removed from Import** list displays any objects in the XML file that will not be imported.

10. Make any of the following adjustments:

    - Click **Reset** to clear the form.
    - Click **Legend** to see icon definitions.

- To remove an object from the Current Selections list, right-click the object, select **Remove** from the shortcut menu, and then click **Remove** to confirm that you want to remove the object.

  If the object has child objects, then select **Remove including children** from the shortcut menu to remove all the child objects at the same time. The item is added to the Objects Removed From Import list.

- To add an item back to the Current Selections list, right-click the list, and click **Add**.

  If the object has child objects, then select **Add including children** from the shortcut menu to add all the child objects at the same time.

- To make substitutions, click **Add Substitutions**.
- To add objects from another XML file, click **Add File** and repeat Steps 3 through 9.
- Click **Show Information** to see information about your imported information.

  The Information page is displayed.

  To see more information, select the **Show Info Level Messages** option, and then click **Show Messages**. Click **Close** to close the Information page.

11. To import the current selections, click **Import**.

    A confirmation dialog box is displayed.

12. Click **Import**.

    The Import Success dialog box is displayed.

13. Click **OK**.

    The objects are imported into Oracle Identity Manager.

38.1.4 Best Practices Related to Using the Deployment Manager

The following are some of the suggested practices and pitfalls to avoid while by using Deployment Manager:

- Export System Objects Only When Necessary
- Export Related Groups of Objects
- Group Definition Data and Operational Data Separately
- Use Logical Naming Conventions for Versions of a Form
- Export Root to Preserve a Complete Organizational Hierarchy
- Provide Clear Export Descriptions
- Check All Warnings Before Importing
- Check Dependencies Before Exporting Data
- Match Scheduled Task Parameters
- Deployment Manager Actions on Reimported Scheduled Tasks
- Compile Adapters and Enable Scheduled Tasks
- Export Entity Adapters Separately
- Check Permissions for Roles

- Back Up the Database
- Import Data When the System Is Quiet
- Migrating Custom Data Objects
- Remove Data Object Fields Before Importing Event Handlers as Dependencies

38.1.4.1 Export System Objects Only When Necessary

You should export or import system objects, for example, Request, Xellerate User, and System Administrator, only when it is absolutely necessary. Exporting system objects from the testing and staging environments into production can cause problems. If possible, exclude system objects when exporting or importing data.

You may want to export or import system objects when, for example, you define trusted source reconciliation on Xellerate User resource objects.

Caution: The Deployment Manager keeps track of imported components and structures, but not of completed imports. After an import is completed, you cannot roll it back to a previous version. A new import is required.

38.1.4.2 Export Related Groups of Objects

Oracle recommends that you use the Deployment Manager to export sets of related objects. A unit of export should be a collection of logical items that you want to group together.

Avoid exporting everything in the database in one operation, or exporting items one at a time. For example, suppose that you manage an integration between Oracle Identity Manager and a target system that includes processes, resource objects, adapters, IT resource type definitions, IT resource definitions, scheduled tasks, and so on. For this environment, you should create groups of related objects before exporting.

For example, if you use the same e-mail definitions in multiple integrations, you should export the e-mail definitions as one unit, and the integrations as a different unit. This enables you to import changes to e-mail definitions independently of target system integration changes. Or, if multiple resources use the same IT resource type definition, you can export and import the type definition separately from other data.

You can import one or more sets of exported data at a time. For example, you can import a resource object definition, an e-mail definition, and an IT resource type definition in a single operation.

38.1.4.3 Group Definition Data and Operational Data Separately

You must group and export definition data and operational data separately.

You configure definition data in the testing and staging environment. Definition data includes resource objects, processes, and rules.

You typically configure operational data in the production environment. Operational data includes groups and group permissions. The testing and staging servers usually do not include this data.

By grouping data according to where it is changed, you know what data goes to testing and staging, and what goes to production. For example, if approval processes are changed in production, you should group approval processes and export them with other operational data.

38.1.4.4 Use Logical Naming Conventions for Versions of a Form

You often revise forms multiple times before exporting them. Avoid generic names, for example, "v23," to differentiate among versions of a form. Create meaningful names, for example, "Before Production" or "After Production Verification." Do not use special characters, including double quotation marks, in version names.

38.1.4.5 Export Root to Preserve a Complete Organizational Hierarchy

When you export a leaf or an organization in an organizational hierarchy, only one dependency level is exported. To export a complete organizational hierarchy, you must export the root of the hierarchy.

38.1.4.6 Provide Clear Export Descriptions

The Deployment Manager records some information automatically, for example, the date of the export, who performed the export, and the source database. You must also provide a meaningful description of the content of the export, for example, "resource definition after xxx attributes added in reconciliation." This informs the importer of the file of the contents of the data being imported.

38.1.4.7 Check All Warnings Before Importing

When importing information to the production environment, check all the warnings before completing the import operation. Treat each warning seriously.

38.1.4.8 Check Dependencies Before Exporting Data

The wizard in the top right pane shows resources that must be available in the target system.

Consider the following types of dependencies:

- If the resources are already available in the target system, they do not need to be exported.
- If the resources are new (not in the target system), they must be exported.
- If the target system does not include the resources, such as lookups, IT resource definitions, or others that are reused, then record the data and export it in a separate file so it can be imported if necessary.

> **Note:** When you export a resource, groups with Data Object permissions on that form are not exported with the resource.

38.1.4.9 Match Scheduled Task Parameters

Scheduled tasks depend on certain parameters to run properly. You can import scheduled task parameters to the production server. Table 38–1 shows the rules for determining how to import scheduled tasks. Note that parameters may be available for tasks that no longer reside on the target system.

Table 38–1 Parameter Import Rules

Parameter Exists in Target System	Parameter Exists in the XML File	Action Taken
Yes	No	Remove the parameter from the target system.

Table 38–1 (Cont.) Parameter Import Rules

Parameter Exists in Target System	Parameter Exists in the XML File	Action Taken
No	Yes	Add the parameter and current value from the XML file.
Yes	Yes	Use the more recent value of the parameter.

38.1.4.10 Deployment Manager Actions on Reimported Scheduled Tasks

A scheduled task is one of the objects that you can import by using the Deployment Manager. Typically, you import a scheduled task into your Oracle Identity Manager environment and later change the values of the scheduled attributes to meet your production requirements. However, if you import the same scheduled task a second time into the same Oracle Identity Manager server, the Deployment Manager does not overwrite the attribute values in the database. Instead, the Deployment Manager compares the attribute value of the reimported XML file to any corresponding attribute values in the database.

The following table summarizes the actions performed by the Deployment Manager during a scheduled task reimport:

Does the Scheduled Task have attribute values in the XML file being imported?	Are there any corresponding attribute values in the database?	Deployment Manager Action
Yes	No	Store attribute values in the database
No	Yes	Delete existing attribute values in the database
Yes	Yes (Newer attribute values indicated by time stamp)	No change in the database
Yes (New attribute values indicated by time stamp)	Yes	Update the database with the new attribute values

38.1.4.11 Compile Adapters and Enable Scheduled Tasks

After an import operation, the adapters are set to recompile and the scheduled tasks are disabled. After importing the classes and adjusting the task attributes, manually recompile the adapters and enable the scheduled tasks.

38.1.4.12 Export Entity Adapters Separately

Entity adapters are modified to bring just the entity adapter, not its usage. If you want to export the usage of an entity adapter, you must separately export each use with a data object by exporting the data object. If you export a data object, all the adapters and event handlers attached to the object along with the permissions on the object are exported. You must pay special attention when exporting data objects. For example, to export a form, you should also add the data object corresponding to the form. This ensures that the associated entity adapters can use the form.

38.1.4.13 Check Permissions for Roles

When you export roles, the role permissions on different data objects are also exported. However, when you import data, any permissions for missing data objects are ignored. If the role is exported as a way of exporting role permission setup, then

check the warnings carefully to ensure that permission requirements are met. For example, if a role has permissions for objects A, B, and C, but the target system only has objects A and B, the permissions for object C are ignored. If object C is added later, the role permissions for C must be added manually, or the role must be imported again.

When you export role that have permissions for viewing certain reports, ensure that the reports exist in the target environment. If the reports are missing, then consider removing the permissions before exporting the role.

38.1.4.14 Back Up the Database

Before you import data into a production environment, back up the database. This enables you to restore the data if anything goes wrong with the import. Backing up the database is always a good precaution before making significant changes.

> **Note:** When you import forms and user-defined fields, you add entries to the database. These database entries cannot be rolled back or deleted. Before each import operation, ensure that the correct form version is active.

38.1.4.15 Import Data When the System Is Quiet

You cannot complete an import operation in a single transaction because it includes schema changes. These changes affect currently running transactions on the system. To limit the effect of an import operation, temporarily disable the Web application for general use and perform the operation when the system has the least activity, for example, overnight.

38.1.4.16 Migrating Custom Data Objects

The SDK table contains metadata definitions for user-defined data objects. When you import data from an XML file into the SDK table, the values in the SDK_SCHEMA column might be modified with the schema name of the source system where the XML file was created. For this reason, after you import data from an XML file into the SDK table, you must check the schema name in the SDK_SCHEMA column, and if necessary, manually change it to the schema name on the target system where the Oracle Identity Manager database is running. To update the schema name in the SDK_SCHEMA column, run a SQL query similar to the following with SQL*Plus on Oracle Database installations or with SQL Query Analyzer on Microsoft SQL Server installations:

```
UPDATE SDK SET SDK_SCHEMA='target system schema name'
```

If you do not update the schema name in the SDK_SCHEMA column, an error similar to the following might be generated when you import other XML files that modify user-defined field (UDF) definitions:

```
CREATE SEQUENCE UGP_SEQ
java.sql.SQLException: ORA-00955: name is already used by an existing object
```

38.1.4.17 Remove Data Object Fields Before Importing Event Handlers as Dependencies

The Deployment Manager does not import event handlers that include data object fields if the event handlers are imported as dependencies. For this reason, you must remove the data object fields from any event handlers that you want to import as dependencies with the Deployment Manager.

38.1.5 Troubleshooting the Deployment Manager

This section contains the following topics:

- Troubleshooting Deployment Manager Issues
- Enabling Logging for the Deployment Manager

38.1.5.1 Troubleshooting Deployment Manager Issues

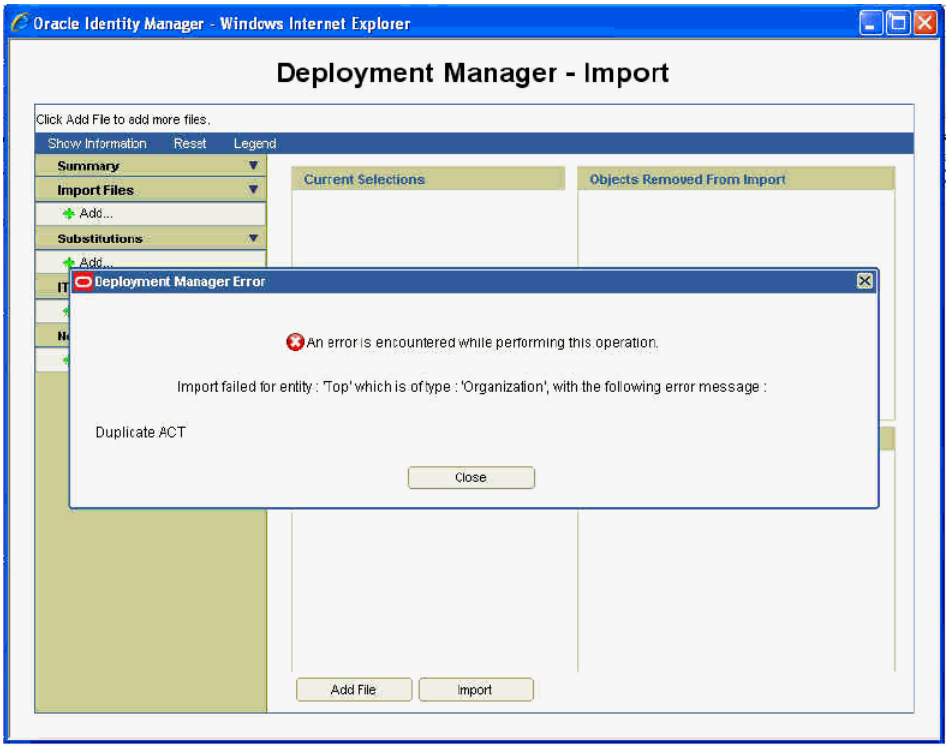
While importing data by using the Deployment Manager, the following information is displayed on the UI for an import failure:

- The entity for which the import failed
- The type of the entity for which the import failed
- The specific error message from the exception object

This information is also printed in logs along with the exception trace.

Figure 38–1 shows a sample error message that is displayed when Deployment Manager import fails.

Figure 38–1 Deployment Manager Import Failure



This helps the user in identifying which entity is causing the failure and why, and the user can try removing that particular entity and importing again if it is not necessary to be imported on the target system. This also helps the support team and developers in identifying the issue if it happens.

Table 38–2 lists the troubleshooting steps that you can perform if you encounter a failure:

Table 38–2 Troubleshooting Deployment Manager

Problem	Solution
In Oracle Identity Manager 11*g* Release 2 (11.1.2.2.0), scheduled job has a dependency on scheduled task. Therefore, scheduled task must be imported prior to scheduled job. As a result, if a XML file has scheduled job entries prior to scheduled task entries, then importing the XML file using Deployment Manager fails with the following error message: `[exec] Caused By: oracle.iam.scheduler.exception.SchedulerExceptio n: Invalid ScheduleTask definition [exec] com.thortech.xl.ddm.exception.DDMException`	Open the XML file and move all scheduled task entries above the scheduled job entries.
Deployment Manager export fails for any object. User is prompted with Export Failed dialog box, and no exception is found in the server log. When you look at the JRE console, you can see the following: java.security.AccessControlException: access denied (java.io.FilePermission *PATH_AND_NAME_OF_THE_FILE*)	Perform the following steps: 1. Modify your java.policy in the *JRE_HOME*/lib/security/ directory. 2. Replace the existing policy file content with the following: `grant{ permission java.security.AllPermission; };` 3. Restart the browser to load the policy again. You can now export the data.
The following error occurs while importing an XML file: `Caused by: oracle.iam.reconciliation.exception.ConfigExcept ion: Profile :Xellerate User InvalidAttributes :`	Perform any one of the following: ■ Remove the attribute on which the error is generated from the XML, and then try importing. ■ Create the missing UDF or other attributes by using configuration service, and then retry the import. ■ Export the UDF shown as missing dependency. Import this UDF first before importing the current XML.
Importing approval policy might result in the following error: `weblogic.kernel.Default (self-tuning)'] [userId: xelsysadm] [ecid: f9e72ab2a292a346:-188377b2:12f96ae9676:-8000-000 0000000000047,0] [APP: oim#11.1.1.3.0] Exception thrown {0}[[ oracle.iam.platform.entitymgr.ProviderException: USER_NOT_FOUND`	An approval policy rule is invalid if it points to an entity (user or organization) that does not exist in Oracle Identity Manager. These invalid approval rules must be corrected to point to a valid entity (user or organization) before the import.

38.1.5.2 Enabling Logging for the Deployment Manager

To enable logging for the Deployment Manager:

1. Add a new logger for the Deployment Manager by editing the logging.xml file, which is located in the following directory path:

   DOMAIN_NAME/config/fmwconfig/servers/*SERVER_NAME*/

For instance, to enable Notification-level logging for Deployment Manager, add the following logger inside the <loggers> section:

```
<logger name='XELLERATE.DDM' level='NOTIFICATION:1' />
```

2. Change the log level defined in the relevent <log_handler>.

See Also: "Configuring Logging" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager* for information about logging level and log handlers in Oracle Identity Manager

38.1.6 Monitoring Deployment Manager Performance Using DMS

Oracle Dynamic Monitoring Service (DMS) can be used to view performance metrics. The following DMS metrics are relevant for monitoring Deployment Manager performance:

- **DeploymentManager:** Metrics that show the time taken by different method calls in Import/Export Operation Beans.
- **DeploymentManager_APIManager:** Metrics that show the time taken by different method calls for APIHandler in Import/Export.

For more information about DMS, see "Understanding the Oracle Dynamic Monitoring Service" in the *Oracle Fusion Middleware Performance and Tuning Guide*.

38.2 Moving from a Test to a New Production Environment Using Movement Scripts

Oracle Identity Manager is a part of the Fusion Middleware environment. To move Oracle Identity Manager from test to production, you use the movement scripts. These scripts copy the Oracle Identity Manager binaries, artifacts, and configurations, and configures production Oracle Identity Manager with new end-points. The movement scripts interact with Oracle Identity Manager artifacts at the test and production environments and updates the production environment to make Oracle Identity Manager functional on the production environment. For detailed information about using the movement scripts, see "Moving from a Test to a Production Environment" in the *Oracle Fusion Middleware Administrator's Guide*. For the complete procedure for moving Oracle Identity Manager components, see "Moving Identity Management Components to a New Target Environment" in the *Oracle Fusion Middleware Administrator's Guide*.

Note: Before proceeding with migrating a source Oracle Identity Manager setup to a target setup, you can refer to "Limitations in Moving from Test to Production" in the *Oracle Fusion Middleware Release Notes* for information about the limitations and known issues related to moving from test to production. In addition, see "Troubleshooting Movement From Test to Production Environment Using Movement Scripts" on page 38-19 for information about the issues that you might encounter while migrating a source Oracle Identity Manager setup and the possible solutions.

To migrate a source Oracle Identity Manager setup to a target setup:

1. Migrate Oracle Identity Manager database schema data from source to target DB host by using the expdp/impdp (Data Pump Utility), as described in "Task 4

Perform Prerequisite Task for Oracle Identity Manager" under section "Moving Identity Management to a New Target Environment" in the *Oracle Fusion Middleware Administrator's Guide*.

2. Create the target setup by using the FMW T2P utilities. To do so:

   a. Run the following commands from the *ORACLE_COMMON_HOME*/bin/ directory.

Note:

- On Microsoft Windows, run the commands with .cmd extension, such as copyBinary.cmd and pasteBinary.cmd. For example, the copyBinary script is *ORACLE_COMMON_HOME*/bin/copyBinary.sh for UNIX and *ORACLE_COMMON_HOME*/bin/copyBinary.cmd for Microsoft Windows.
- Some arguments might be invalid for Windows operating system. For example, the -ipl *PATH_TO_ORACLE_INVENTORY_POINTER* argument does not work in Windows.
- This document provides the syntax for running the copyBinary, copyConfig, extractMovePlan, and pasteBinary scripts. For detailed information about these scripts, parameters, and example usages, see "Using the Movement Scripts" in the *Oracle Fusion Middleware Administrator's Guide*.

```
./copyBinary.sh -javaHome PATH_TO_JDK -al ARCHIVE_LOCATION -smw SOURCE_MW_
HOME -silent false -idw true -ipl PATH_TO_ORACLE_INVENTORY_POINTER -silent
true -ldl PATH_TO_LOG_DIRECTORY

./copyConfig.sh -javaHome PATH_TO_JDK -archiveLoc ARCHIVE_LOCATION
-sourceDomainLoc SOURCE_DOMAIN_LOCATION -sourceMWHomeLoc MIDDLEWARE_HOME_
LOCATION -domainHostName DOMAIN_HOST_NAME -domainPortNum DOMAIN_PORT_NUMBER
-domainAdminUserName DOMAIN_ADMIN_USERNAME -domainAdminPasswordFile DOMAIN_
ADMIN_PASSWORD_FILE -silent true -ldl PATH_TO_LOG_DIRECTORY

./extractMovePlan.sh -javaHome PATH_TO_JDK -archiveLoc ARCHIVE_LOCATION
-planDirLoc MOVE_PLAN_DIRECTORY
```

In between running the extractMovePlan and pasteConfig scripts, update the moveplan with the new values for configuring the target. See "Modifying Move Plans" in the *Oracle Fusion Middleware Administrator's Guide* for information about common moveplan modifications. See the moveplan property descriptions in "Table 20-22 Move Plan Properties for Oracle Identity Manager" in the *Oracle Fusion Middleware Administrator's Guide*.

Note:

- While editing the moveplan, provide the listen address of the target in the Oracle Identity Manager Managed Server details.
- The datasource JDBC URL coming from source to the moveplan can either be in SID format, which is "jdbc:oracle:thin:@*HOST*:*PORT*:*SID*", or in service name format, which is "jdbc:oracle:thin:*HOST*:*PORT*/*SERVICE_NAME*". But you must always provide the JDBC URL in the datasource details in the service name format.

b. On the target host, create a new directory and copy pasteBinary.sh from the *SOURCE_MACHINE*/Middleware/oracle_common/bin/ directory. In addition, copy the cloningclient.jar file from the *SOURCE_MACHINE*/Middleware/oracle_common/jlib/ directory to the target host. Make sure that these two files are in the same location, for example /scratch/aime1/scripts. Then, run the following command:

```
./pasteBinary.sh -javaHome PATH_TO_JDK -al ARCHIVE_LOCATION -tmw TARGET_MW_
HOME -silent false -idw true -esp true -ipl PATH_TO_ORACLE_INVENTORY_
POINTER -ldl PATH_TO_LOG_DIRECTORY -silent true
```

c. Go to the *TARGET_MIDDLEWARE_HOME*/bin/ directory, and run the following command:

```
./pasteConfig.sh -javaHome PATH_TO_JDK -archiveLoc ARCHIVE_LOCATION
-targetDomainLoc TARGET_DOMAIN_PATH -targetMWHomeLoc TARGET_MIDDLEWARE_
HOME_PATH -movePlanLoc MOVE_PLAN_PATH -domainAdminPasswordFile DOMAIN_
ADMIN_PASSWORD_FILE -silent true -ldl PATH_TO_LOG_DIRECTORY
```

Note:

- You might need to change the permissions on the *TARGET_MIDDLEWARE_HOME* and the target directory on which the JAR has been placed.
- Provide consistent directory paths for each of the parameters. For example, if you are using absolute path for *MIDDLEWARE_HOME*, then specify this path in the same way at all places.

3. Verify or modify the following configurations after full T2P migration:

- In the xlclient.cmd file, update the JDK path if the JDK library that was configured with the Design Console on the source is no longer accessible on the target.

  In the config/xlconfig.xml file, update the Application JNDI URL to point to the target application URL instead of source application URL.
- The IT Resource configurations are not part of the moveplan in the T2P procedure. After completing the T2P steps and starting the servers on the target setup, you can configure the IT Resource parameters as per the production setup. In Oracle Identity System Administration, under Configuration, click IT Resource. On the Manage IT Resource page, click the edit icon for the IT resource that you want to modify.

- Some entities, such as users and provisioned accounts, are not migrated from source to target during the T2P procedure, as they are considered transactional data. Therefore, user personalization settings such as sort order, saved searches, and layout changes will not be found on the target setup.
- Some users, such as role owners, are referenced in many places in Oracle Identity Manager. After full T2P migration, references to such users are replaced with reference to *SYSTEM_ADMINISTRATOR_USERNAME*, the Oracle Identity Manager system administrator.
- Customization done on the LDAP Sync configuration, such as Role Category Container Rules has to be configured on the T2P environment post migration.

38.2.1 Troubleshooting Movement From Test to Production Environment Using Movement Scripts

Table 38–3 lists the troubleshooting step that you can perform if you encounter issues related to movement from a test to a new production environment by using movement scripts.

Table 38–3 Troubleshooting Movement From Test to Production Environment Using Movement Scripts

Problem	Solution
After migrating from an Oracle Identity Manager clustered deployment to another, SOA Server is running but you are not able to access soa-infra. This is because the coherence settings are pointing to source.	Change the coherence settings accordingly by referring to "Specifying the Host Name Used by Oracle Coherence" in the *Oracle Fusion Middleware Enterprise Deployment Guide for Oracle Identity Management*, and then restart the SOA Server.

The following section is logged in the cloning error logs:
`NOTIFICATION: PManager instance is created without multitenancy support as JVM flag "oracle.multitenant.enabled" is not set to enable multitenancy support.`
`Sep 24, 2013 10:26:55 PM oracle.security.jps.internal.config.xml.XmlConfigurationFactory initDefaultConfiguration`
`SEVERE: java.io.FileNotFoundException: ./config/jps-config.xml (No such file or directory)`
`Sep 24, 2013 10:26:55 PM oracle.mds NOTIFICATION: Auditing is disabled for component MDS.`
`Sep 24, 2013 10:26:55 PM oracle.mds NOTIFICATION: PManager instance is created without multitenancy support as JVM flag "oracle.multitenant.enabled" is not set to enable multitenancy support.`
`Sep 24, 2013 10:26:55 PM oracle.security.jps.internal.config.xml.XmlConfigurationFactory initDefaultConfiguration`
`SEVERE: java.io.FileNotFoundException: ./config/jps-config.xml (No such file or directory)`
`Sep 24, 2013 10:26:55 PM oracle.mds NOTIFICATION: Auditing is disabled for component MDS.`
`Sep 24, 2013 10:26:55 PM oracle.mds`
	This section of the cloning error logs is benign and can be safely ignored.

Table 38–3 (Cont.) Troubleshooting Movement From Test to Production Environment Using Movement

Problem	Solution
After migrating Oracle Identity Manager to a new environment, database connection-related errors are thrown when you try one or more of the following operations: ■ Create a user ■ Search and open a user ■ Provision an application instance to a user	Set the following tuning parameters as shown: `JAVA_ OPTIONS="-Djbo.ampool.doampooling=true -Djbo.ampool.minavailablesize=1 -Djbo.ampool.maxavailablesize=120 -Djbo.recyclethreshold=60 -Djbo.ampool.timetolive=-1 -Djbo.load.components.lazily=true -Djbo.doconnectionpooling=true -Djbo.txn.disconnect_level=1 -Djbo.connectfailover=false -Djbo.max.cursors=5 -Doracle.jdbc.implicitStatementCacheSize=5 -Doracle.jdbc.maxCachedBufferSize=19 ${JAVA_OPTIONS}"` For information about these tuning parameters, see "Application Module Pooling" in the *Oracle Fusion Middleware Performance and Tuning Guide*.

38.3 Migrating the Policies

You can migrate the policies from one Oracle Identity Manager deployment to another, for example, from a test environment to production environment, using the migrateSecurityStore WLST command utility provided by Oracle Platform Security. For example, to migrate all Oracle Identity Manager approval policies from the source to target, run the following migration command in overwrite mode from the WLST prompt:

```
migrateSecurityStore(type="appPolicies",configFile="<configuration file path>",
src="DBsourceContext", dst="DBdestinationContext", srcApp="OIM", dstApp="OIM",
overWrite="true")
```

For information about the various parameters used with the migrateSecurityStore command and the usage options for the utility, see the 'migrateSecurityStore' section in "Infrastructure Security Custom WLST Commands" in the *Oracle Fusion Middleware WebLogic Scripting Tool Command Reference* for 11*g* Release 2 (11.1.2).

For information about constructing the configuration file, see "Migrating Policies Manually" in the *Oracle Fusion Middleware Application Security Guide* for 11*g* Release 1 (11.1.1).

Note: While creating the configuration file for migration of Oracle Identity Manager policies, make sure that the following properties are set for the source system as well as the target system, as shown:

```
<property value="cn=mySourceDomain"
name="oracle.security.jps.farm.name"/>
<property value="cn=mySourceRootName"
name="oracle.security.jps.ldap.root.name"/>
```

You can take these values from the jps-config.xml file in your domain at the following directory path:

DOMAIN_HOME/config/fmwconfig/

Search for the property names in the jps-config.xml in your domain home and copy the value to this configuration file.

For the source system values, search in the domain home of the source system, and similarly for the target system.

38.3.1 Troubleshooting Migration of Policies

Table 38–4 lists the troubleshooting step that you can perform if you encounter issues related to migration of policies.

Table 38–4 Troubleshooting Migration of Policies

Problem	Solution
The following error is displayed: `oracle.security.jps.service.policystore.PolicyStoreIncompatibleVersionException: JPS-06100: Policy Store version 11.0 and Oracle Platform Security Services Version 11.1.1.6.0 are not compatible`	Not being able to connect or find the policy store can be the possible causes of this error message. To troubleshoot the problem, check the configuration file for the database connectivity details, root name, and farm name for both source and target.

Part XI

Reports and Audit

This part describes about audit engine and how to configure reports in Oracle Identity Manager.

It contains the following chapters:

39

Configuring Reports

This chapter describes Oracle Identity Manager Reports and contains the following topics:

39.1 What is Oracle Identity Manager Reports?

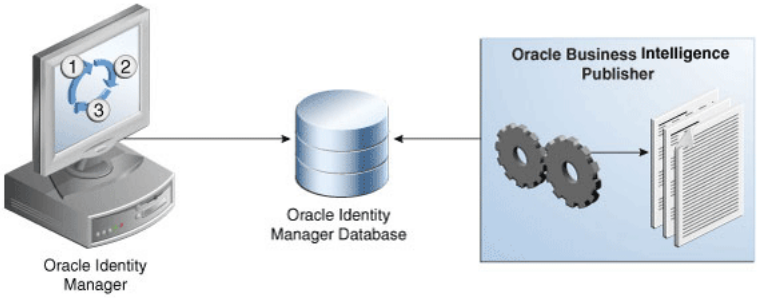
Oracle Identity Manager Reports enables you to use Oracle BI Publisher as the reporting solution for Oracle Identity Management products.

> **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 39–1, 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 39–1 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.

39.2 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.

39.3 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.

39.4 Deploying Oracle Identity Manager Reports

This section explains how to deploy Oracle Identity Manager Reports 11*g* Release 2 (11.1.2.2.0) and contains the following topics:

- Creating the Metadata Repository
- Installing BI Publisher 11g (11.1.1.7.1)

Note: To use reports on Oracle Identity Manager 11*g* Release 2 (11.1.2.2.0), you must install Oracle BI Publisher 11*g* (11.1.1.7.1). To install Oracle BI Publisher 11*g* (11.1.1.7.1), first install Oracle BI Publisher 11*g* (11.1.1.7.0), and then apply the patch for Oracle BI Publisher 11*g* (11.1.1.7.1) by using OPATCH. The patch number for Oracle BI Publisher 11*g* (11.1.1.7.1) is 16556157, which can be downloaded at the following URL:

https://support.oracle.com

You can download Oracle BI Publisher 11g (11.1.1.7.0) by navigating to the following URL:

http://www.oracle.com/technetwork/middleware/bi-enterprise-edition/downloads/bi-downloads-1923016.html

See *Oracle Fusion Middleware Installation Guide for Oracle Business Intelligence* for information about installing BI Publisher 11g (11.1.1.7.0).

39.4.1 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 11*g* (11.1.1.7.1). 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-1923016.html

For installing BI Publisher 11*g* (11.1.1.7.1), the following metadata repositories are required:

- Matadata Store (MDS)
- Business Intelligence Platform (BI Platform)

To create the repository in your database by using the RCU utility:

1. Log in to the database as SYSDBA.

   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.

39.4.2 Installing BI Publisher 11*g* (11.1.1.7.1)

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 11*g* (11.1.1.7.1).

If you do not have Oracle WebLogic Server installed, then OBIEE 11*g* installs the WebLogic Server by default, and creates the bi domain named bifoundation_domain under the user_projects/domains directory. See "Deploying Oracle Identity Manager Reports on OBIEE Environment" on page 39-7 for information about deploying Oracle Identity Manager reports on OBIEE environment.

Installing BI Publisher 11*g* (11.1.1.7.1) is described in the following sections:

- Starting the Oracle Business Intelligence Wizard
- Installing BI Publisher 11g (11.1.1.7.1) When Oracle WebLogic Server is Not Installed
- Installing BI Publisher 11g (11.1.1.7.1) When Oracle WebLogic Server is Installed

- Deploying Oracle Identity Manager Reports on OBIEE Environment

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 (11.1.1.7.1) 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 Installations 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.

   Note: When you are installing BI Publisher 11*g* (11.1.1.7.1), the wizard in Configure Components page displays the following options:

   - Business Intelligence Enterprise Edition
   - Business Intelligence Publisher
   - Real Time Decisions
   - Essbase Suite

   If you are using the report function for Oracle Identity Manager 11*g*, then ensure that you select the **Business Intelligence Publisher** option only, and not any other option.

   ---
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 39-3.
11. Complete the remaining steps of the wizard by clicking **Next**.

Installing BI Publisher 11g (11.1.1.7.1) When Oracle WebLogic Server is Installed

When BI Publisher 11g (11.1.1.7.1) 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.

Note: When you are installing BI Publisher 11*g* (11.1.1.7.1), the wizard in Configure Components page displays the following options:

- Business Intelligence Enterprise Edition
- Business Intelligence Publisher
- Real Time Decisions
- Essbase Suite

If you are using the report function for Oracle Identity Manager 11*g*, then ensure that you select the **Business Intelligence Publisher** option only, and not any other option.

---

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 39-3.
9. Complete the remaining steps of the wizard by clicking **Next**.

Note: If you intend to use the Software Only Install type as part of your strategy to later extend an existing Oracle WebLogic Server domain, then ensure the following:

- The domain is empty, that is, no other software products exist in the domain.
- The domain physically exists on the same computer where you plan to install and configure Oracle Business Intelligence. Extending domains remotely is not supported.

For more information, see "Installing Oracle Business Intelligence" at the following URL:

http://docs.oracle.com/cd/E23943_01/bi.1111/e10539/c4_installing.htm#BIEIG366

Deploying Oracle Identity Manager Reports on OBIEE Environment

To deploy Oracle Identity Manager reports on OBIEE environment:

1. Copy the oim_product_BIP11gReports_11_1_2_2_0.zip file to the BI Publisher directory.
2. Extract the zip file and check for the Oracle Identity Manager and Translations directories created.
3. To sync the catalog with /analytics:
   a. Login to BI Publisher. so that Oracle Identity Manager reports are displayed in the catalog folder on BIP.
   b. Click **Administration**.
   c. In System Maintenance, click `Server Configuration`.
   d. In Catalog, select the `Oracle BIEE Catalog` catalog type, and click **Apply**.
   e. Restart the BI Server.

39.5 Configuring Oracle Identity Manager Reports

This section describes configuring BI Publisher 11*g* (11.1.1.7.1) in the following topics:

- Configuring Security on BI Publisher 11g (11.1.1.7.1)
- Configuring Data Sources for Running Oracle Identity Manager Reports

39.5.1 Configuring Security on BI Publisher 11*g* (11.1.1.7.1)

To configure security in BI Publisher 11*g* (11.1.1.7.1):

1. 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.
2. Select **Administration, Security Centre, Security Configuration,** and **Local Superuser.**
3. Enable Local Superuser and register an authorized superuser. For example, `xelsysadm`.
4. Restart the BI Publisher Server (bi_server1).
5. Ensure that you are able to log in to the BI Publisher as a registered superuser. For example, xelsysadm.
6. Copy the oim_product_BIP11gReports_11_1_2_2_0.zip file to the BI Publisher directory.
7. Extract the zip file and check for the Oracle Identity Manager and Translations directories created.
8. Navigate to Home and Catalog Folders in the BI Publisher. You can see the Oracle Identity Manager folder under the Shared Folders option.
9. Select **Administration, Data Sources,** and **JDBC Configuration.**

10. In the JDBC Configuration location, create two data sources for OIM 11*g* using the following details:
    - Data Source Name: `OIM JDBC`
      - Driver Type : `Oracle 11g`
      - Database Driver Class : `oracle.jdbc.OracleDriver`
      - Connection String : `jdbc:oracle:thin:@oimhost:1521:orcl`
      - Username : `DEV_OIM`
      - Password: <*PASSWORD*>
    - Data Source Name: `BPEL JDBC`
      - Driver Type : `Oracle 11g`
      - Database Driver Class : `oracle.jdbc.OracleDriver`
      - Connection String : `jdbc:oracle:thin:@oimhost:1521:orcl`
      - Username : `DEV_SOAINFRA`
      - Password: <*PASSWORD*>

39.5.2 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

39.5.2.1 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 10*g* or Oracle 11*g* 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.

39.5.2.2 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 10*g* or Oracle 11*g* 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 database 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.

39.6 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 11*g* reports.

39.6.1 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 39-7 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 Configuring Security on BI Publisher 11g (11.1.1.7.1) 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.

39.6.2 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 Configuring Security on BI Publisher 11g (11.1.1.7.1) 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.

39.6.3 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 39-9 for more information about creating a BPEL data source.
2. Log in to Oracle BI Publisher. See step 3 in Configuring Security on BI Publisher 11g (11.1.1.7.1) 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 **Open** for the report you want to generate.
5. Select an output format for the report, and click **Apply**.

   The report is generated based on the BPEL-based JDBC data source.

40

Understanding Auditing

User profile audits cover changes to user profile attributes, user membership, resource provisioning, access policies, and resource forms.

The audit engine collects auditing information in Oracle Identity Manager. Whenever a profile is modified, the audit engine captures the changes (the delta) and updates (or generates, if missing) the snapshots of the user and role profiles and stores these snapshots and deltas in XML format. The audit engine also contains post-processors, which, based on the generated XML, populate the reporting tables with relevant data. To maintain high performance, by default the audit engine performs these tasks in an asynchronous and offline manner by using the underlying Java Messaging Service (JMS) provided by the application server.

This chapter discusses the following topics:

40.1 Audit Levels

As mentioned earlier in this chapter, when you install Oracle Identity Manager user profile auditing is enabled by default and the auditing level is set to Resource Form. If you change the auditing level, then you must run the GenerateSnapshot.sh script (on UNIX) or the GenerateSnapshot.bat script (on Microsoft Windows). This script is in the *IDM_HOME*/server/bin directory. The script examines all users in Oracle Identity Manager database and generates new snapshots based on the new auditing level.

Note: Before running the GenerateSnapshot script, you must set the following environment variables:

- APP_SERVER: Set the value to weblogic.
- OIM_ORACLE_HOME: Set it to the directory on which Oracle Identity Manager is installed.
- JAVA_HOME: Set it to the directory path of the Java Runtime directory for the Oracle Identity Manager server.
- WL_HOME: Set it to the directory on which Oracle WebLogic Server is installed.
- MW_HOME: Set it to the directory on which Oracle Fusion Middleware is installed.
- DOMAIN_HOME: Set it to the Oracle Identity Manager domain.

When you run the GenerateSnapshot script, you are prompted to enter the following:

```
[Enter Xellerate admin username :]SYSTEM_ADMINISTRATOR_USERNAME
[Enter password for xelsysadm :]SYSTEM_ADMINISTRATOR_PASSWORD
[Threads to use [ 8 ]]
[Enter serverURL :[t3://OIM_HOST:OIM_PORT]
[Enter context Factory :[ weblogic.jndi.WLInitialContextFactory]
```

Note: If you change the auditing level, then you must run the GenerateSnapshot script before allowing users to access the system.

You can configure the "level of detail for auditing" aspect of the auditing engine and specify the audit level as the value of the XL.UserProfileAuditDataCollection system property in the Advanced Administration.

See Also: "System Properties in Oracle Identity Manager" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager* for information about this system property

The supported audit levels are:

- **Process Task:** Audits the entire user profile snapshot together with the resource lifecycle process.
- **Resource Form:** Audits user record, role membership, resource provisioned, and any form data associated to the resource.
- **Resource:** Audits the user record, role membership, and resource provisioning.
- **Membership:** Only audits the user record and role membership.
- **Core:** Only audits the user record.
- **None:** No audit is stored.

Note: When you specify a particular audit level, all audit levels that are at a lower priority level are automatically enabled. For example, if you specify the Membership audit level, then the Core audit level is automatically enabled.

Audit level specifications are case-sensitive. When you specify an audit level, ensure that you do not change the case (uppercase and lowercase) of the audit level.

40.2 Tables Used for Storing Information About Auditors

Information about auditors is stored in the following tables of the database:

- **AUD:** This table stores metadata about all the auditors defined in Oracle Identity Manager.
- **aud_jms:** This table stores data to be consumed by the audit engine and eventually by the auditors. It is an operational and intermediate staging table.

  The key in this table is sent to the JMS. Oracle Identity Manager uses this table to control the order of the changes when multiple changes are made to the same user. You can use the Issue Audit Messages Task scheduled task to automate the reissue of messages that are not processed. For more information about this scheduled task, see "Managing the Scheduler" in *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager*.

40.3 Issuing Audit Messages

Oracle Identity Manager provides a scheduled task named Issue Audit Messages Task. This scheduled task retrieves audit message details from the aud_jms table and sends a single JMS message for a particular identifier and auditor entry in the aud_jms table. An MDB processes the corresponding audit message.

The following is the attribute of this task:

Max Records

Use the Max Records attribute to specify the maximum number of audit messages to be processed for a specified scheduled task run. The default value of this attribute is 400.

If there is a backlog of audit messages in the aud_jms table, then you can increase the value of the Max Records attribute. The value that you set depends on how many messages the JMS engine can process during the default scheduled task execution interval. This, in turn, depends on the performance of the application server and database. Before increasing the Max Records value, you must determine how much time is taken to process the number of audit messages in the JMS destination (oimAuditQueue) by, for example, using the administrative console of the application server. If the time taken is less than the scheduled task interval, then you can make a corresponding increase in the value of the Max Records attribute.

Part XII

Appendixes

This part contains the following appendixes:

- Appendix A, "General Customization Concepts"
- Appendix B, "The FacesUtils Class"
- Appendix C, "Username Reservation and Common Name Generation"

A

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 appendix describes the following topics:

- Rule Elements, Variables, Data Types, and System Properties
- Service Accounts
- Design Console Actions

A.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 A–1 lists the rule elements that can be used to create Oracle Identity Manager rules, by using the Rule Designer form.

Table A–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

Table A–1 (Cont.) Rule Elements to Create Oracle Identity Manager Rules

Type	Sub-Type	Attribute Source	Variable
			Start Date
General	NA	User Profile Data	User Type
			Identity Status
			User Login
			Design Console Access
			All fields in the User form under System Entities in Oracle Identity System Administration
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
			All fields in the User form under System Entities in Oracle Identity System Administration
Process Determination	**Organization Provisioning**	Object Information	Object Name
			Object Type
		Request Target Information	Organization Customer Type
			Organization Name
			Organization Status
			Parent Organization
			All fields in the Organization form under System Entities in Oracle Identity System Administration

Table A–1 (Cont.) Rule Elements to Create Oracle Identity Manager Rules

Type	Sub-Type	Attribute Source	Variable
Process Determination	**User Provisioning**	Requester Information; Request Target Information	All fields in the User form under System Entities in Oracle Identity System Administration
		Object Information	Object Name
			Object Type
		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
			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	All the UDFs defined in the User form under System Entities in Oracle Identity System Administration
Pre-Populate	**Organization Provisioning; User Provisioning**	Requester Information	Display Name
			Email
			End Date
			First Name
			Identity
			Last Name

Table A–1 (Cont.) Rule Elements to Create Oracle Identity Manager Rules

Type	Sub-Type	Attribute Source	Variable
			Manager Full Name
			Manager
			Middle Name
			Organization Name
			User Type
			Start Date
			Identity Status
			User Role Name
			User Login
			Design Console Access
			All fields in the User form under System Entities in Oracle Identity System Administration
		Request Information	Request Creation Date
			Request ID
			Request Object Action
			Request Priority
			Requestor
		Object Information	Object Name
			Object Type
	Organization Provisioning	Request Target Information	Organization Customer Type
			Organization Name
			Organization Status
			Parent Organization
			Any field that is displayed in the Organization form under System Entities in Oracle Identity System Administration
	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

Table A–1 (Cont.) Rule Elements to Create Oracle Identity Manager Rules

Type	Sub-Type	Attribute Source	Variable
			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
			All fields in the User form under System Entities in Oracle Identity System Administration

You can use the Email Definition form to create templates for e-mail notifications to be sent to the users. Table A–2 lists the variables that can be used to create e-mail templates by using the Email Definition form.

Table A–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

Table A–2 (Cont.) Variables to Create Templates

Type	Target	Location Type	Contact Type	Variable
				Xellerate 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	NA	NA	Object Name
				Object Target Type
				Object Type
	Process Information	NA	NA	Object Name
				Process Name
				Process Type
				Process Response
				Process Response Description
	Object Data Information	NA	NA	Any field that is defined 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

Table A–2 (Cont.) Variables to Create Templates

Type	Target	Location Type	Contact Type	Variable
				User End Date
				User Group Name
				User Login
				User Manager
				User Start Date
				Xellerate Type
				Any field that is displayed in the **User Defined Fields** region of the **User Profile** tab of the **Users** form

Table A–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 A–3 Properties Associated with Data Types for Creating Oracle Identity Manager Forms

Data Type	Data Property	Description
Text Field	Required	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 thi field. **Note**: The default value for this data property is false.
	Visible Field	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. **Note**: The default value for this data property is true.
Lookup Field	Auto Complete	By entering "true" in the corresponding **Property Value** field, Oracle Identit Manager filters the lookup field. A user can then add characters to the look field before double-clicking it. By doing so, only those Lookup values which match these characters are displayed in the Lookup window. 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, Ne 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" in the associated **Property Value** field. The default property value for the **Auto Complete** property is false.
	Column Captions	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 ea column heading into the **Property Value** field, separating them with comma for example, **Organization Name**, **Organization Status**.
Lookup Field	Column Names	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. If the Lookup window has multiple columns, then enter each database column into the **Property Value** field, separating them with commas.

Table A–3 (Cont.) Properties Associated with Data Types for Creating Oracle Identity Manager Forms

Data Type	Data Property	Description
	Column Widths	In the corresponding **Property Value** field, enter the width of the column that is displayed in the Lookup window. 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**.
	Lookup Column Name	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.

	Lookup Query	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.
		To correctly display the data returned from a query, you must add a `lookupfield.header` property to the xlWebAdmin_locale.properties file. For example, consider the following SQL query: select usr_status from usr. To view the data returned from the query, you must add the following entry to the xlWebAdmin_ locale.properties files:
		`lookupfield.header.users.status=User Status`
		If the xlWebAdmin_locale.properties file does not contain a `lookupfield.header` property for your specified query, then the Identity System Administration displays a lookup window after you click the corresponding lookup icon.
		The syntax for a `lookupfield.header` property is as follows:
		`lookupfield.header.`*`column_code`*`=display value`
		The *column_code* portion of the entry must be lowercase and any space must be replaced by the underscore character (_).
		By default, the following entries for lookup field column headers are already available in the system resource bundle:
		`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`

Table A–3 (Cont.) Properties Associated with Data Types for Creating Oracle Identity Manager Forms

Data Type	Data Property	Description
Lookup Field	Lookup Code	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. **Important**: The Lookup Code data property can be used in lieu of the **Colun 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 propert **Tip**: An easy way to enter a lookup code is by starting the Lookup Definitio form, querying for the desired code, copying this code to the Clipboard, and pasting it into the Lookup Code field. **Note**: The classification type of the lookup definition code must be of Looku Type (the Lookup Type radio button on the Lookup Definition form must b selected).
	Required	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. **Note**: The default value for this data property is false.
	Visible Field	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. **Note**: The default value for this data property is true.
Text Area	Number of Rows	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" int the **Property Value** field.
	Required	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 thi field. **Note**: The default value for this data property is false.
	Visible Field	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. **Note**: The default value for this data property is true.
IT Resource Lookup Field	Type	If you select this data property, then a box is displayed in the **Property Valu** field. From this box, select the type of Server for the IT Resource. **Important**: This property is required.
	Required	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" i this field. **Note**: The default value for this data property is false.
	Visible Field	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. **Note**: The default value for this data property is true.
Date and Time Window	Required	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 fie **Note**: To populate this text field, double-click it, and select a date and time from the Date & Time window that is displayed. **Note**: The default value for this data property is false.

Table A–3 (Cont.) Properties Associated with Data Types for Creating Oracle Identity Manager Forms

Data Type	Data Property	Description
	Visible Field	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. **Note**: The default value for this data property is true.
Password Field	Required	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. **Note**: The default value for this data property is false.
	Visible Field	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. **Note**: The default value for this data property is true.
Lookup Field	Lookup Code	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.

	Lookup Query	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.
		To correctly display the data returned from a query, you must add a `lookupfield.header` property to the xlWebAdmin_locale.properties file. For example, consider the following SQL query: select usr_status from usr. To view the data returned from the query, you must add the following entry to the xlWebAdmin_ locale.properties files:
		`lookupfield.header.users.status=User Status`
		If the xlWebAdmin_locale.properties file does not contain a `lookupfield.header` property for your specified query, then the Identity System Administration displays a lookup window after you click the corresponding lookup icon.
		The syntax for a `lookupfield.header` property is as follows:
		`lookupfield.header.`*`column_code`*`=display value`
		The *column_code* portion of the entry must be lowercase and any space must be replaced by the underscore character (_).
		By default, the following entries for lookup field column headers are already available in the system resource bundle:
		`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`

Table A–3 (Cont.) Properties Associated with Data Types for Creating Oracle Identity Manager Forms

Data Type	Data Property	Description
	Column Names	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. If the Lookup window has multiple columns, then enter each database column into the **Property Value** field, separating them with commas.
Radio Button	Required	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. **Note**: The default value for this data property is false.
	Visible Field	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. **Note**: The default value for this data property is true.
Check Box	Required	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. **Note**: The default value for this data property is false.
	Visible Field	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. **Note**: The default value for this data property is true.
Combo Box	Lookup Code	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. **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. **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. **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).
	Required	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. **Note**: The default value for this data property is false.
	Visible Field	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. **Note**: The default value for this data property is true.
Text Field (Display Only)	Visible Field	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. **Note**: The default value for this data property is true.

A.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 A–4 describes the service account management tasks and their corresponding APIs.

Table A–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.	`tcUserOperationsIntf.changeFromServiceAccount` `tcUserOperationsIntf.changeToServiceAccount`
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.	`tcUserOperationsIntf.moveServiceAccount`

A.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.

A.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.

A.3 Design Console Actions

Table A–5 lists the Oracle Identity Manager actions, and the conditions and results of these actions.

Table A–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 *field label* `Updated` naming convention (for example, HomeDirectory Updated).	The condition for this task is met (the process task begins with the *field label* `Updated` 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.

Table A–5 (Cont.) Oracle identity Manager Actions, Conditions, and Results

Action	Condition	Result
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.

B

The FacesUtils Class

The FacesUtils class is used in the customization use cases shown in "Using Managed Beans" on page 30-46. This class contains various helper methods for re-rendering components, evaluating EL expressions, and accessing attributes through binding.

Example B–1 provides the code snippet of the FacesUtils class with implementation of some of the methods:

Note: If you add the code from Example B–1 in any ViewControllerProject source code, then some packages, such as the following, might not be found:

```
import oracle.adf.model.BindingContext;
import oracle.adf.model.binding.DCBindingContainer;
import oracle.adf.model.binding.DCControlBinding;
import oracle.binding.AttributeBinding;
import oracle.binding.ControlBinding;
```

You must add ADF Model Runtime to the class path to resolve the errors related to importing these packages. To add ADF Model Runtime to project class path:

1. Right-click the project, and select **Project Properties**.
2. On the left navigation bar, select **Libraries and Classpath**.
3. Click **Add Library**.
4. Under Extension, select **ADF Model Runtime**.
5. Click **OK**.
6. Click **OK**.

Example B–1 Sample FacesUtils Class

```
package oracle.iam.ui.sample.common.view.utils;

import java.io.IOException;

import java.util.Map;
import java.util.ResourceBundle;

import javax.el.ELContext;
import javax.el.ExpressionFactory;
import javax.el.MethodExpression;
import javax.el.ValueExpression;
```

```
import javax.faces.application.Application;
import javax.faces.application.FacesMessage;
import javax.faces.component.UIComponent;
import javax.faces.context.FacesContext;

import oracle.adf.model.BindingContext;
import oracle.adf.model.binding.DCBindingContainer;
import oracle.adf.model.binding.DCControlBinding;
import oracle.adf.view.rich.context.AdfFacesContext;

import oracle.binding.AttributeBinding;
import oracle.binding.ControlBinding;

import oracle.iam.ui.platform.utils.TaskFlowUtils;

import oracle.jbo.uicli.binding.JUCtrlActionBinding;
import oracle.jbo.uicli.binding.JUCtrlListBinding;
import oracle.jbo.uicli.binding.JUEventBinding;


public class FacesUtils {

    private FacesUtils() {
        // do not instantiate
        throw new AssertionError();
    }

    /*
     * Re-render the component.
     */
    public static void partialRender(UIComponent component) {
        if (component != null) {
            AdfFacesContext.getCurrentInstance().addPartialTarget(component);
        }
    }

    /*
     * Sets attribute value through attribute binding.
     */
    public static void setAttributeBindingValue(String attributeName,
                                                Object value) {
        AttributeBinding binding = getAttributeBinding(attributeName);
        if (binding != null) {
            binding.setInputValue(value);
        } else {
            throw new IllegalArgumentException("Binding " + attributeName +
                                               " does not exist.");
        }
    }

    /*
     * Gets attribute value using attribute binding.
     */
    public static <T> T getAttributeBindingValue(String attributeName,
                                                 Class<T> clazz) {
        AttributeBinding binding = getAttributeBinding(attributeName);
        if (binding != null) {
            return (T)binding.getInputValue();
        } else {
```

```
            throw new IllegalArgumentException("Binding " + attributeName +
                                               " does not exist.");
        }
    }

    /*
     * Gets attribute value using list binding.
     */
    public static <T> T getListBindingValue(String attributeName,
                                            Class<T> clazz) {
        ControlBinding ctrlBinding = getControlBinding(attributeName);
        if (ctrlBinding instanceof JUCtrlListBinding) {
            JUCtrlListBinding listBinding = (JUCtrlListBinding)ctrlBinding;
            return (T)listBinding.getAttributeValue();
        } else {
            throw new IllegalArgumentException("Binding " + attributeName +
                                               " is not list binding.");
        }
    }

    public static ControlBinding getControlBinding(String name) {
        ControlBinding crtlBinding = getBindings().getControlBinding(name);
        if (crtlBinding == null) {
            throw new IllegalArgumentException("Control Binding '" + name +
                                               "' not found");
        }
        return crtlBinding;
    }

    public static AttributeBinding getAttributeBinding(String name) {
        ControlBinding ctrlBinding = getControlBinding(name);
        AttributeBinding attributeBinding = null;
        if (ctrlBinding != null) {
            if (ctrlBinding instanceof AttributeBinding) {
                attributeBinding = (AttributeBinding)ctrlBinding;
            }
        }
        return attributeBinding;
    }

    public static DCBindingContainer getBindings() {
        FacesContext fc = FacesContext.getCurrentInstance();
        ExpressionFactory exprfactory =
            fc.getApplication().getExpressionFactory();
        ELContext elctx = fc.getELContext();

        ValueExpression valueExpression =
            exprfactory.createValueExpression(elctx, "#{bindings}",
                                              Object.class);

        DCBindingContainer dcbinding =
            (DCBindingContainer)valueExpression.getValue(elctx);

        return dcbinding;
    }

    /*
     * Evaluates EL expression and returns value.
     */
    public static <T> T getValueFromELExpression(String expression,
```

```
                                                Class<T> clazz) {
        FacesContext facesContext = FacesContext.getCurrentInstance();
        Application app = facesContext.getApplication();
        ExpressionFactory elFactory = app.getExpressionFactory();
        ELContext elContext = facesContext.getELContext();
        ValueExpression valueExp =
            elFactory.createValueExpression(elContext, expression, clazz);
        return (T)valueExp.getValue(elContext);
    }

    /*
     * Gets MethodExpression based on the EL expression. MethodExpression can then
be used to invoke the method.
     */
    public static MethodExpression getMethodExpressionFromEL(String expression,
                                                             Class<?> returnType,
                                                             Class[] paramTypes) {
        FacesContext facesContext = FacesContext.getCurrentInstance();
        Application app = facesContext.getApplication();
        ExpressionFactory elFactory = app.getExpressionFactory();
        ELContext elContext = facesContext.getELContext();
        MethodExpression methodExp =
            elFactory.createMethodExpression(elContext, expression, returnType,
                                             paramTypes);
        return methodExp;
    }

    public static ELContext getELContext() {
        return FacesContext.getCurrentInstance().getELContext();
    }

    /*
     * Shows FacesMessage. The message will not be bound to any component.
     */
    public static void showFacesMessage(FacesMessage fm) {
        FacesContext.getCurrentInstance().addMessage(null, fm);
    }

    /*
     * Launch bounded taskFlow based on provided parameters.
     */
    public static void launchTaskFlow(String id, String taskFlowId,
                                      String name, String icon,
                                      String description, String helpTopicId,
                                      boolean inDialog,
                                      Map<String, Object> params) {
        // create JSON payload for the contextual event
        String jsonPayLoad =
            TaskFlowUtils.createContextualEventPayLoad(id, taskFlowId,
                                                       name, icon, description,
                                                       helpTopicId, inDialog,
                                                       params);

        // create and enqueue contextual event
        DCBindingContainer bc =

(DCBindingContainer)BindingContext.getCurrent().getCurrentBindingsEntry();
        DCControlBinding ctrlBinding = bc.findCtrlBinding(TaskFlowUtils.RAISE_
TASK_FLOW_LAUNCH_EVENT);
        // support both bindings - using eventBinding as well as methodAction
```

```
        if (ctrlBinding instanceof JUEventBinding) {
            JUEventBinding eventProducer = (JUEventBinding) ctrlBinding;
            bc.getEventDispatcher().queueEvent(eventProducer, jsonPayLoad);
        } else if (ctrlBinding instanceof JUCtrlActionBinding) {
            JUCtrlActionBinding actionBinding = (JUCtrlActionBinding) ctrlBinding;
            bc.getEventDispatcher().queueEvent(actionBinding.getEventProducer(),
jsonPayLoad);
        } else {
            throw new IllegalArgumentException("Incorrect binding for " +
TaskFlowUtils.RAISE_TASK_FLOW_LAUNCH_EVENT);
        }
        bc.getEventDispatcher().processContextualEvents();
    }

    /*
     * Redirect to a provided url.
     */
    public static void redirect(String url) {
        try {
            FacesContext fctx = FacesContext.getCurrentInstance();
            fctx.getExternalContext().redirect(url);
            fctx.responseComplete();
        } catch (IOException ex) {
            throw new RuntimeException(ex);
        }
    }

}
```

C

Username Reservation and Common Name Generation

This appendix contains the following topics:

C.1 Username Reservation

When the request for user creation is submitted, the following scenarios are possible:

- While the request is pending, another create user request is submitted with the same username. If the second request is approved and the user is created, then the first request, when approved, fails because the username already exists in Oracle Identity Manager.
- While the request is pending, another user with the same username is directly created in the LDAP identity store. When the create user request is approved, it fails while provisioning the user entity to LDAP because an entry already exists in LDAP with the same username.

To avoid these problems, you can reserve the username in both Oracle Identity Manager and LDAP while the create user request is pending for approval. If a request is created to create a user with the same username, then an error message is displayed and the create user request is not created.

For reserving the username:

- The USER ATTRIBUTE RESERVATION ENABLED system property must be set to TRUE for the functionality to be enabled. For information about searching and modifying system properties, see "Managing System Properties" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager*.
- Reservation in LDAP is done only if reservation functionality is enabled, and LDAP is in sync with Oracle Identity Manager database. For information about synchronization between Oracle identity Manager and LDAP identity store, see "Integration Between LDAP Identity Store and Oracle Identity Manager" in *Oracle Fusion Middleware Developer's Guide for Oracle Identity Manager.*

Note:

- If LDAP provider is not configured, then the reservation is done only in Oracle Identity Manager.
- When LDAP synchronization and user attribute reservation features are enabled, it is recommended to enable UID uniqueness in the directory server. Without this, user reservation in the directory does not work properly because while the user is reserved in the reservation container, the user with the same user ID can be created in the user container. This results is user creation failure when Oracle Identity Manager tries to move the user from the reservation container to the user container.

If user attribute reservation is enabled, the reservation happens in two phases:

In the first phase, an entry is created in Oracle Identity Manager database and a user is created in reservation container. This entry in Oracle Identity Manager database is removed after successful creation of user, rejection by approver, or request failure.

In the second phase, in LDAP, on successful creation, the user is moved to the reservation container. In other cases such as rejection by approver or request failure, the user is removed from the reservation container.

After the request-level and operation-level approvals are obtained for the create user request, the username is no longer reserved in the username container in LDAP. The username is moved to the container in which the existing users are stored. The user is also created in Oracle Identity Manager.

This section consists of the following topics:

C.1.1 Enabling and Disabling Username Reservation

The username reservation functionality is enabled by default in Oracle Identity Manager. This is done by keeping the value of the USER ATTRIBUTE RESERVATION ENABLED system property to TRUE. You can verify the value of this system property in the System Configuration section of the Oracle Identity Manager System Administration Console.

To disable username reservation:

1. Log in to Oracle Identity System Administration.
2. In the left pane, under System Management, click **System Configuration.** The Advanced Administration opens in a new window.
3. In the left pane, click the search icon to search for all existing system properties. A list of system properties are displayed in the search results table.
4. Click **User Attribute Reservation Enabled**. The System Property Detail page for the selected system property is displayed, as shown in Figure C–1:

Figure C–1 The System Property Detail Page

5. In the Value field, enter **False**.
6. Click **Save**. The username reservation functionality is disabled.

C.1.2 Configuring the Username Policy

Username Policy is a plugin implementation for username operations such as username generation and username validation. You can change the default policies from the System Configuration section in Oracle Identity System Administration.

In case of a Create User usecase, the plugins are invoked only if the user login is not provided. In such a case, the plugin to be invoked is picked up from the system property, "Default policy for username generation". The custom user name policy is honored in all the following use cases:

- Create Admin User
- Create User Request
- Reconciliation
- Bulk Load

Table C–1 lists the predefined username policies provided by Oracle Identity Manager. In this table, the dollar ($) sign in the username generation indicates random alphabet:

Table C–1 Predefined Username Policies

Policy Name	Expected Information	Username Generated
oracle.iam.identity.usermgmt.impl.plugins.EmailIdPolicy	E-mail	E-mail value is used as the auto-generated user name
oracle.iam.identity.usermgmt.impl.plugins.LastNameFirstInitialLocalePolicy	First name, last name, and locale	last name + first initial_locale, last name + middle initial + first initial_locale, last name + $ + first initial_locale (all possibilities of single random alphabets), last name + $$ + first initial_locale
oracle.iam.identity.usermgmt.impl.plugins.FirstInitialLastNameLocalePolicy	Firstname, Lastname, Locale	first initial + lastname_locale, first initial + middle initial + first name_locale, first initial + $ + lastname_locale, first initial + $$ + lastname_locale

Table C–1 (Cont.) Predefined Username Policies

Policy Name	Expected Information	Username Generated
oracle.iam.identity.usermgmt.impl.plugins.LastNameFirstInitialPolicy	Firstname, Lastname	lastname+firstInitial, lastname+middleinitial+firstInitial, lastname+$+firstInitial (all possibilities of single random alphabets) , lastname+$$+firstInitial
oracle.iam.identity.usermgmt.impl.plugins.FirstInitialLastNamePolicy	Firstname, Lastname	firstInitial+lastname, firstInitial+middleInitial+firstname, firstInitial+$+lastname, firstInitial+$$+lastname
oracle.iam.identity.usermgmt.impl.plugins.LastNameFirstNamePolicy	Firstname, Lastname	lastname.firstname, lastname.middleinitial.firstname, lastname.$.firstname (all possibilities of single random alphabets) , lastname.$$.firstname
oracle.iam.identity.usermgmt.impl.plugins.FirstNameLastNamePolicy	Firstname, Lastname	firstname.lastname, firstname.middleinitial.lastname, firstname.$.lastname (all possibilities of single random alphabets) , firstname.$$.lastname
oracle.iam.identity.usermgmt.impl.plugins.DefaultComboPolicy	Any one of the following: - Email - Firstname, Last Name - Last name.	If e-mail is provided, then username is generated based on the e-mail. If e-mail is not available, then it generates username based on firstname and lastname by appending a user domain to it. If first name is not available, then it generates the username based of the last name only by appending a user domain to it. The user domain is configured as the Default user name domain system property, and the default value is @oracle.com
oracle.iam.identity.usermgmt.impl.plugins.LastNamePolicy,	Lastname	lastname, middle initial + lastname , $ + lastname, $$ + lastname
oracle.iam.identity.usermgmt.impl.plugins.LastNameLocalePolicy	Lastname, Locale	lastname_locale, middle initial + lastname_locale , $ + lastname_locale, $$ + lastname_locale
oracle.iam.identity.usermgmt.impl.plugins.FirstNameLastNamePolicyForAD	Firstname, Lastname	firstname+lastname, substring of firstname+lastname+$, substring of firstname+ substring of lastname+$
oracle.iam.identity.usermgmt.impl.plugins.LastNameFirstNamePolicyForAD	Lastname, Firstname	lastname+firstname, lastname+substring of firstname+$, substring of lastname+ substring of firstname+$

The policy implementations generate the username, check for its availability, and if the username is not available, then generate other username based on the policy in the order mentioned in Table C–1, and repeat the procedure. The dollar ($) sign in the username generation indicates random alphabet. If any of the expected information is missing, then the policies generate errors.

Values must be provided for all the parameters of the username generation format. If any of the parameters are not provided, then Oracle Identity Manager generates an error. For example, If the firstname.lastname policy is configured and the firstname is

not provided, then the error would be "An error occurred while generating the Username. Please provide firstname as expected by the firstname.lastname policy".

The username generation is exposed as public APIs in User Manager. Oracle Identity Manager provides an utility class for accessing the functionality of generating user names. The class that contains utility methods is as shown:

```
oracle.iam.identity.usermgmt.api.UserManager
```

The UserManager class exposes the following public API for username generation and validation:

```
//Method that will generate username based on default policy

    public String generateUserNameFromDefaultPolicy(Map<String,  Object> attrMap)
 throws UserNameGenerationException,  UserManagerException;

//Method that will generate username based on policy

    public String generateUserNameFromPolicy(String policyId,  Map<String, Object>
attrMap) throws UserNameGenerationException,  UserManagerException;

//Method that will check whether username is valid against default policy

    public boolean isUserNameValidForDefaultPolicy(String userName,  Map<String,
Object> attrMap) throws UserManagerException;

//Method that will check whether username is valid against given policy

    public boolean isUserNameValidForPolicy(String userName, String  policyId,
Map<String, Object> attrMap) throws  UserManagerException;

//Method to return all policies (including customer written)

        public List<Map<String, String>> getAllUserNamePolicies(Locale locale)

//Method that will return policy description in given locale

    public String getPolicyDescription(String policyID, Locale locale)
```

Table C–2 lists the constants defined in oracle.iam.identity.usermgmt.utils.UserNameGenerationUtil to represent the policy ID of the default username policies:

Table C–2 Constants Representing Policy IDs

Policy Name	Constant
EmailIDPolicy	EMAIL_ID_POLICY
LastNameFirstInitialLocalePolicy	FIRSTNAME_LASTNAME_POLICY
FirstInitialLastNameLocalePolicy	LASTNAME_FIRSTNAME_POLICY
LastNameFirstInitialPolicy	FIRSTINITIAL_LASTNAME_POLICY
FirstInitialLastNamePolicy	LASTNAME_FIRSTINITIAL_POLICY
LastNameFirstNamePolicy	FIRSTINITIAL_LASTNAME_LOCALE_POLICY
FirstNameLastNamePolicy	LASTNAME_FIRSTINITIAL_LOCALE_POLICY
DefaultComboPolicy	DEFAULT_COMBO_POLICY

Table C–2 (Cont.) Constants Representing Policy IDs

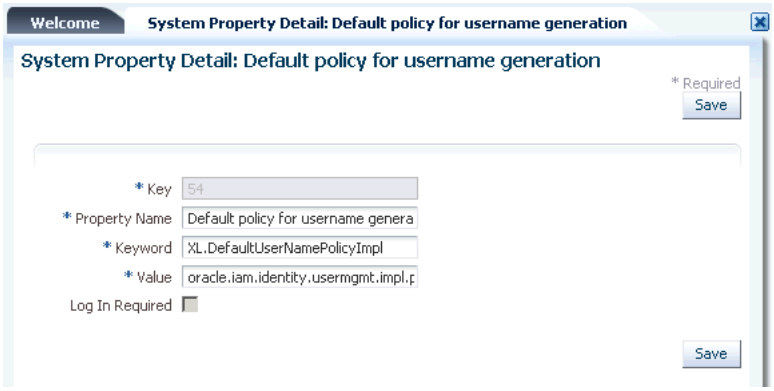
Policy Name	Constant
LastNamePolicy	LASTNAME_POLICY
LastNameLocalePolicy	LASTNAME_LOCALE_POLICY
FirstNameLastNamePolicyForAD	FIRSTNAME_LASTNAME_POLICY_FOR_AD
LastNameFirstNamePolicyForAD	LASTNAME_FIRSTNAME_POLICY_FOR_AD

When called to generate username, the policy classes expect the attribute values to be set in a map by using the key constants defined in the oracle.iam.identity.utils class.Constants. This means that a proper parameter value must be passed to call the method by using the appropriate constant defined for it, for example, the FirstName parameter has a constant defined for it.

The default username policy can be configured by using the Oracle Identity System Administration. To do so:

1. Navigate to the System Configuration section.
2. Search for all the system properties.
3. Click **Default policy for username generation**. The System Property Detail page for the selected property is displayed, as shown in Figure C–2:

Figure C–2 The Default Username Policy Configuration



The XL.DefaultUserNameImpl system property is provided for picking up the default policy implementation. By default, it points to the default username policy, which is oracle.iam.identity.usermgmt.impl.plugins.DefaultComboPolicy displayed in the Value field.

4. In the Value field, enter **oracle.iam.identity.usermgmt.impl.plugins.*POLICY***. Here, *POLICY* is one of the policy implementations.

Note: All the plug-ins must be registered with Oracle Identity Manager by using the /identity/metadata/plugin.xml file. A sample plugin.xml file is as shown:

```
<plugins
pluginpoint="oracle.iam.identity.usermgmt.api.UserNamePolicy">
<plugin
pluginclass="oracle.iam.identity.usermgmt.impl.plugins.LastNameFirs
tNamePolicy"
version="1.0" name="LastNameFirstNamePolicy"/>
</plugins>
```

5. Click **Save**.

C.1.3 Writing Custom User Name Policy

You can write your own policies by adding new plug-ins and changing the default policies from the System Configuration section in Oracle Identity System Administration.

See Also: "Developing Plug-ins" on page 27-1 for information about the plug-in framework

The UserManager exposes APIs for username operations. The APIs take the user data as input and return a generated username. The APIs make a call to plug-ins that return the username. This allows you to replace the default policies with custom plug-ins with your implementation for username operations.

Note:

- For user name generation and validation, public APIs are exposed in UserManager.
- While creating the policy, ensure that the attributes used in generating the username are defined in the request data set.

You can write your own username policies by implementing the plug-in interface, as shown:

```
package oracle.iam.identity.usermgmt.api;

public interface UserNameGenerationPolicy extends
 oracle.iam.identity.usermgmt.api.UserNamePolicy {
public String getUserName(Map<String, Object> reqData) throws
UserNameGenerationException;
public boolean isGivenUserNameValid(String userName, Map<String, Object> reqData);

//methods inherited from old user name policy interface
//oracle.iam.identity.usermgmt.api.UserNamePolicy
public String getUserNameFromPolicy(HashMap<String, String> reqData) throws
UserNameGenerationException;
public boolean isUserNameValid(String userName, HashMap<String, String> reqData);
public String getDescription(Locale locale);

}
```

This plug-in point is exposed as a kernel plug-in that takes request data as input and returns the username. Each plug-in expects some information and generates username based on that information provided.

Note: Oracle Identity Manager provides an abstract implementation of the oracle.iam.identity.usermgmt.api.UserNameGenerationPolicy interface as the oracle.iam.identity.usermgmt.api.AbstractUserNameGenerationPolicy class name. Therefore, you need not implement the following two methods:

```
public String getUserNameFromPolicy(HashMap<String, String>
reqData) throws UserNameGenerationException;

public boolean isUserNameValid(String userName, HashMap<String,
String> reqData);
```

Create the plug-in ZIP with lib (containing the JAR) and plugin.xml, and then register it by performing the procedure in section "Registering and Unregistering Plug-ins By Using the Plugin Registration Utility" on page 27-8. The following is a sample plugin.xml file:

```
<?xml version="1.0" encoding="UTF-8"?>
<oimplugins>
<plugins pluginpoint="oracle.iam.identity.usermgmt.api.UserNamePolicy">
<plugin
pluginclass="oracle.iam.identity.usermgmt.impl.plugins.CustomDepartmentNumberEmplo
yeeNumberPolicy" version="1.0" name="CustomDepartmentNumberEmployeeNumberPolicy"/>
</plugins>
</oimplugins>
```

The following are the guidelines on while writing custom user name policies:

- Policies should implement the new interface oracle.iam.identity.usermgmt.api.UserNameGenerationPolicy.
- Custom user name policies must be re-entrant. This means that the custom code in the policy should return the same user login if approver has updated an attribute that does not contribute in generating the user login.

For sample implementation please refer below:

```
package oracle.iam.identity.usermgmt.impl.plugins;

import java.util.Locale;
import java.util.Map;

import oracle.iam.identity.exception.UserNameGenerationException;
import oracle.iam.identity.usermgmt.api.AbstractUserNameGenerationPolicy;
import oracle.iam.identity.usermgmt.api.UserManagerConstants;
import oracle.iam.identity.usermgmt.api.UserNameGenerationPolicy;

public class CustomDepartmentNumberEmployeeNumberPolicy extends
AbstractUserNameGenerationPolicy implements UserNameGenerationPolicy {


          private String departmentNumberKey =
UserManagerConstants.AttributeName.DEPARTMENT_NUMBER.getId();

          private String employeeNumberKey =
```

```
UserManagerConstants.AttributeName.EMPLOYEE_NUMBER.getId();

          @Override
          public String getUserName(Map<String, Object> reqData)
                                    throws UserNameGenerationException {

          String departmentnumber = reqData.get(departmentNumberKey) == null ?
null : reqData.get(departmentNumberKey).toString();

          String employeeNumber = reqData.get(employeeNumberKey) == null ? null :
reqData.get(employeeNumberKey).toString();

          // Required in case of approver edit. If approver has not modified any
attribute which contributes in user name generation , then return same old user
login

          //Check if user data is not changed using checkForSameUserLogin method
present in AbstractUserNameGenerationPolicy, then return same user login

          //OR use Map<String, Object> existingData = (Map<String, Object>)
reqData.get(oracle.iam.identity.usermgmt.api.UserManagerConstants.EXISTING_DATA )
to implement your own comparison logic

          // If existingData is NULL, it means generate a new user login. If it is
not NULL, then it means policy is invoked during approver edit.

          // If it is NOT NULL, Compare value of participating attributes from
existingData and reqData. If same, return same user login as present in
existingData ; otherwise generate a new user login.

          String oldUserLogin = checkForSameUserLogin(reqData , new
String[]{departmentNumberKey , employeeNumberKey});
               if(oldUserLogin!=null)
                    return oldUserLogin;

               // TODO: DO basic validations. Also, Ensure newly generated user
name is unique and not reserved. You may use utility methods in
oracle.iam.identity.usermgmt.utils.UserNamePolicyUtil for preforming validations.
               return departmentnumber + "-" + employeeNumber;
          }

          @Override
          public boolean isGivenUserNameValid(String userName, Map<String, Object>
reqData) {
               // TODO : custom implementation
               return true;
}

@Override
public String getDescription(Locale locale) {
               return "User Name Generation Policy using department number and
employee number";
     }

}
```

C.1.4 Releasing the Username

The username is released in the following scenarios:

- When the request is approved, and the user is successfully created in Oracle Identity Manager and provisioned to LDAP, and the username from the reserved table is removed. The reserved username is removed after successful user creation after the approvals. The reserved entry in LDAP is removed and the actual user is created.
- If the request is rejected, then the reserved entry of username in LDAP and Oracle Identity Manager is removed.
- If the request fails while or before creating a user in Oracle Identity Manager or LDAP, then the reserved username is deleted.

C.1.5 Configuring Username Generation to Support Microsoft Active Directory

In Oracle Identity Manager deployment with LDAP synchronization is enabled, where Microsoft Active Directory (AD) is the data store, the User Login attribute in Oracle Identity Manager is mapped to the uid attribute in LDAP, which in turn is mapped to the sAMAccountName attribute. The sAMAccountName attribute is used as login for all AD-based applications. There is a limitation on the maximum length supported for value contained in the sAMAccountName attribute in AD. It cannot exceed 20 characters.

Oracle Identity Manager accepts user name as an input at the time of user creation and it can be more than 20 characters. Because AD does not support user name of more than 20 characters, Oracle Identity Manager can be configured to generate the user name, which consists of less than 20 characters.

When AD is used as data store, you can configure the autogeneration of user name by setting the value of the XL.DefaultUserNamePolicyImpl system property to any one of the following:

- **FirstNameLastNamePolicyForAD:** Generates the user login by prefixing a substring from the first name to that of the last name
- **LastNameFirstNamePolicyForAD:** Generates the user login by prefixing a substring from last name to that of the first name

See "Administering System Properties" for information about the XL.DefaultUserNamePolicyImpl system property and setting values of system properties.

Note: If AD is the data store, then any one of the FirstNameLastNamePolicyForAD or LastNameFirstNamePolicyForAD policies must be used. Any other user name generation policy will fail to generate the user name.

C.2 Common Name Generation

The generation of the Common Name user attribute value in Oracle Identity Manager is described in the following sections:

- Common Name Generation for Create User Operation
- Common Name Generation for Modify User Operation

C.2.1 Common Name Generation for Create User Operation

In an LDAP-enabled deployment of Oracle Identity Manager, Fusion applications such as Human Capability Management (HCM) does not pass the common name via SPML

request. Given that the common name is a mandatory attribute in LDAP and Oracle Identity Manager is setup to use it as the RDN, Oracle Identity Manager must generate a unique common name.

Based on the description on Common Name, it is the user's display name consisting of first name and last name. Therefore, Oracle Identity Manager generates the Common Name with the help of a common name generation policy that specifies the Common Name in the "firstname lastname" format.

To configure common name generation in Oracle Identity Manager, set the value of the XL.DefaultCommonNamePolicyImpl system property to oracle.iam.identity.usermgmt.impl.plugins.FirstNameLastNamePolicy. For information about the XL.DefaultCommonNamePolicyImpl system property and setting the value of a system property, see "Managing System Properties" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager*.

The following are the details of the FirstNameLastNamePolicy:

- Expected information: Firstname, Lastname
- Common Name generated: firstname.lastname, firstname.$.lastname (all possibilities of single random alphabets), firstname.$$.lastname and so on until a unique common name is generated

> **Note:** The common name must be reserved until the user is created by the request so that multiple requests generated simultaneously having same first and last names do not generate the same common name.

C.2.2 Common Name Generation for Modify User Operation

When the user profile is modified, one or more attributes can change. HCM cannot filter out and send only the modified data to Oracle Identity Manager because it does not have the old user attributes and cannot determine which ones are modified. Therefore, all attributes including the common name (CN) are passed to Oracle Identity Manager by the SPML request. Because the CN changed, Oracle Identity Manager attempts a modify operation (modrdn) in the directory resulting in DN change. Because of this unintended DN change, the group membership DN becomes stale resulting in the user loosing membership in that group. This subsequently results in authorization failure. This happens when referential integrity is turned off in the LDAP server, and therefore, the referenced groups are not updated when the RDN of the user changes. Therefore, referential integrity must be turned on in the target LDAP server. Otherwise, the group memberships become stale. The referential integrity issue is also applicable to roles. Groups are also members of other groups and any RDN changes must be reflected as well.

You can turn on the referential integrity by setting the value of the XL.IsReferentialIntegrityEnabled system property to TRUE. For information about this system property, see "Managing System Properties" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager*.

Table C–3 lists the possible scenarios when RDN is modified:

Table C–3 RDN Modification Scenarios

Referential Integrity in LDAP	XL.IsReferentialIntegrity Enabled	Result of Modify Operation (modrdn)
Disabled	FALSE	Oracle Identity Manager generates an error and operation fails.
Disabled	TRUE	Modify operation passes from Oracle Identity Manager and RDN is changed in LDAP. However, the group references are not updated and are stale. This configuration is not recommended.
Enabled	FALSE	Oracle Identity Manager generates an error and modify operation fails. This property must be set to TRUE in Oracle Identity Manager because referential integrity is enabled in LDAP.
Enabled	TRUE	Modify operation passes and RDN is updated. In addition, the references for the DN are updated in LDAP.
Multiple directories with roles and users stored in separate directories. Referential integrity property is not relevant here.	FALSE	Modify operation fails from Oracle Identity Manager. This is not supported by LDAP. Therefore, FALSE is the recommended value in Oracle Identity Manager for the property.
Multiple directories with roles and users stored in separate directories. Referential integrity property is not relevant here.	TRUE	Modify operation passes and RDN is modified. However, because LDAP does not support referential integrity in multiple directories, the group references are stale and must be manually updated.

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X