

Oracle® Identity Manager

Audit Report Developer's Guide

Release 9.1.0.1

E14045-01

December 2008

Copyright © 2008, Oracle and/or its affiliates. All rights reserved.

Primary Author: Lyju Vadassery

Contributing Author: Debapriya Datta

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

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

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

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

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

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

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

Contents

Preface	xi
Audience.....	xi
Documentation Accessibility	xi
Related Documents	xii
Documentation Updates	xii
Conventions	xii
 1 Introduction to Oracle Identity Manager Auditing	
Auditing Design Components.....	1-1
Profile Auditing.....	1-2
Standard and Customized Reports	1-2
Secondary Data Source Reporting	1-2
 2 Audit Engine	
Audit Engine	2-1
Audit Levels.....	2-1
Post-Processors	2-2
Creating Custom Post-Processors.....	2-2
Tables Used for Storing Information About Auditors.....	2-3
Issuing Audit Messages	2-3
 3 User Profile Auditing	
Data Collected for Audits	3-1
Capture and Archiving of User Profile Audit Data	3-1
XML Representation of Snapshots and Changes to Snapshots.....	3-2
XML Representation of User Profile Snapshots	3-2
XML Representation of Changes to User Profile Snapshots	3-5
Storage of Snapshots.....	3-7
Trigger for Taking Snapshots	3-8
Post-Processor Used for User Profile Auditing	3-8
Tables Used for User Profile Auditing.....	3-8

4 Group Profile Auditing

Data Collected for Audits	4-1
Capture and Archiving of Group Profile Audit Data	4-1
XML Representation of Snapshots and Changes to Snapshots	4-2
XML Representation of Group Profile Snapshots	4-2
XML Representation of Changes to Group Profile Snapshots	4-3
Storage of Snapshots	4-6
Trigger for Taking Snapshots	4-7
Tables Used for Group Profile Auditing	4-7

5 Oracle Identity Manager Reporting

Reporting Features	5-1
Data Layer	5-2
XML Metadata	5-3
API Layer	5-3
List of Reports	5-3
Exception Reports	5-4
Using the UPA Form Data Upgrade Utility	5-6
How to Create a Report	5-8
Writing the Stored Procedure	5-8
Generic Parameters	5-8
Specific Parameters	5-9
Other Stored Procedure Notes	5-10
Example of a Stored Procedure Signature	5-10
Creating the Report XML Metadata	5-11
StoredProcedure Element	5-11
ReturnColumns tag	5-14
Creating an REP Entry and Providing Access to the Report	5-15
Creating a New Entry in the REP Table	5-15
Loading XML Metadata	5-16
Providing a User Group with Access to a Report	5-16
Modifying Property Files for Translating Label Names, Report Names, and Report Descriptions	5-16
Adding Properties for Translating Label Names	5-16
Adding Properties for Translating Report Names and Descriptions	5-18
Working with Third-Party Reporting Tools	5-19
Date Ranges in Historical Reports	5-19
Limitations	5-19

6 Secondary Datasource Reporting

Writing User Profile Audits to a Secondary Datasource	6-1
Steps to Set Up a Secondary Data Source	6-4
Using JBoss Application Server with a Secondary Data Source	6-4
Cluster Configuration for JBoss Application Server	6-5
Using Oracle WebLogic Server with a Secondary Data Source	6-5
Cluster Configuration for Oracle WebLogic Server	6-7

Using IBM WebSphere Application Server with a Secondary Data Source.....	6-7
Cluster Configuration for IBM WebSphere Application Server	6-8
Using Oracle Application Server with a Secondary Data Source.....	6-8
Cluster Configuration for Oracle Application Server.....	6-10

A Sample Code for a Custom Post-Processor

Index

List of Examples

3-1	XML Representation of a User Profile Snapshot	3-3
3-2	XML Representation of Changes to a Sample User Profile Snapshot	3-6
4-1	XML Representation of a Group Profile Snapshot.....	4-2
4-2	XML Representation of Changes to a Sample Group Profile Snapshot.....	4-5
5-1	Signature of the User Resource Access Stored Procedure for Oracle Database	5-10

List of Tables

3-1	Values of the reason and reasonKey Attributes for User Profile Auditing.....	3-6
3-2	Definition of the UPA Table	3-7
4-1	Values of the reason and reasonKey Attributes for Group Profile Auditing	4-4
4-2	Definition of the GPA Table	4-6
5-1	List of Operational Reports	5-3
5-2	List of Historical Reports	5-4
5-3	Variables of the UPA Form Data Upgrade Utility	5-6
6-1	Tables and Constraints Used in Historical Reports	6-2

Preface

This guide introduces you to the process of generating historical and operational reports related to the audit features of Oracle Identity Manager.

Audience

This guide is intended for Oracle Identity Manager administrators and users. It is assumed that you are familiar with the Oracle Identity Manager system and documentation (specifically, the information given in *Oracle Identity Manager Administrative and User Console Guide*).

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible to all users, including users that are disabled. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at <http://www.oracle.com/accessibility/>.

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

TTY Access to Oracle Support Services

To reach AT&T Customer Assistants, dial 711 or 1.800.855.2880. An AT&T Customer Assistant will relay information between the customer and Oracle Support Services at 1.800.223.1711. Complete instructions for using the AT&T relay services are available at <http://www.consumer.att.com/relay/tty/standard2.html>. After the AT&T Customer Assistant contacts Oracle Support Services, an Oracle Support

Services engineer will handle technical issues and provide customer support according to the Oracle service request process.

Related Documents

For more information, see the following documents in the Oracle Identity Manager documentation set for release 9.1.0.1:

- *Oracle Identity Manager Readme*
- *Oracle Identity Manager Installation and Configuration Guide for JBoss Application Server*
- *Oracle Identity Manager Installation and Configuration Guide for Oracle WebLogic Server*
- *Oracle Identity Manager Administrative and User Console Customization Guide*

Note: The remaining documents in the Oracle Identity Manager release 9.1.0 documentation library also apply to Oracle Identity Manager release 9.1.0.1.

Documentation Updates

Oracle is committed to delivering the best and most recent information available. For information about updates to the Oracle Identity Manager documentation set, visit Oracle Technology Network at

<http://www.oracle.com/technology/documentation/index.html>

Conventions

The following text conventions are used in this document:

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

Introduction to Oracle Identity Manager Auditing

Oracle Identity Manager provides a powerful audit engine to collect extensive data for audit and compliance purposes. It also provides a flexible reporting engine to run reports on that data. The customer can use the Audit and Report functionality together to capture, archive, and view entity and transactional data for compliance monitoring and IT-centric processes and forensic auditing. Therefore, with the audit and compliance modules, Oracle Identity Manager provides profile auditing, reporting, and attestation features. You can capture, transport, store, retrieve, and remove historical data over its life cycle. Security is maintained at every stage of the data life cycle.

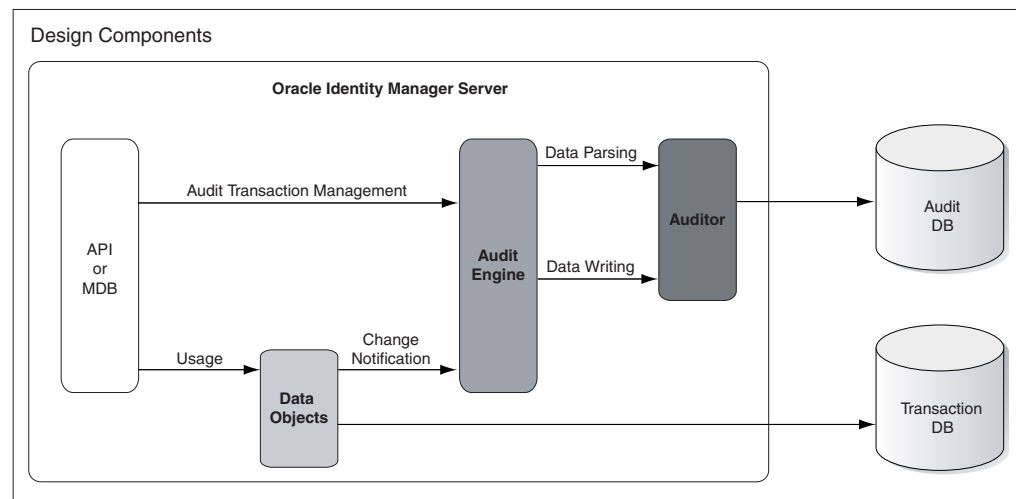
This guide discusses the profile auditing and reporting features of Oracle Identity Manager. See *Oracle Identity Manager Administrative and User Console Guide* for attestation details.

This chapter discusses the following topics:

- [Auditing Design Components](#)
- [Profile Auditing](#)
- [Standard and Customized Reports](#)
- [Secondary Data Source Reporting](#)

1.1 Auditing Design Components

[Figure 1–1](#) shows the design components of the Oracle Identity Manager auditing process.

Figure 1–1 Design Components of the Auditing Process

Any action that a user performs in Oracle Identity Manager translates into an Application Programming Interface (API) call or into a Message Driven Bean (MDB) picking up a message to process an action.

One action can cause multiple changes. All changes are combined into an **audit transaction**. Each API method that can modify data objects calls the `startTransaction` method in the audit engine at the beginning of the API call and the `endTransaction` method at the end of the API call. This defines boundaries for the audit transaction. The audit engine generates a transaction ID to identify the changes made in the transaction.

1.2 Profile Auditing

Oracle Identity Manager provides auditing and historical archiving of profile information. It takes a snapshot of a profile, stores the snapshot in an audit table in the database, and updates the snapshot each time the profile data changes.

Note: In the context of profile auditing, the term **snapshot** means a copy taken of the entire profile data at any instant when the data is modified.

1.3 Standard and Customized Reports

Oracle Identity Manager provides standard reports for viewing archived data. You can also create customized reports.

1.4 Secondary Data Source Reporting

When you first install Oracle Identity Manager, it uses a primary data source for creating reports. To reduce the load on the primary data source, you can configure a secondary data source for reporting. To use a secondary database, you must configure the replication of data between transactional data and the reporting database.

Audit Engine

User profile audits cover changes to user profile attributes, user membership, resource provisioning, access policies, and resource forms.

This chapter discusses the following topics:

- [Audit Engine](#)
- [Tables Used for Storing Information About Auditors](#)
- [Issuing Audit Messages](#)

2.1 Audit Engine

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 group 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 section discusses the following topics:

- [Audit Levels](#)
- [Post-Processors](#)
- [Creating Custom Post-Processors](#)

2.1.1 Audit Levels

As mentioned earlier in this chapter, when you install Oracle Identity Manager with the Audit and Compliance module, 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 Linux) or the `GenerateSnapshot.bat` script (on Microsoft Windows). This script is in the `OIM_HOME/xellerate/bin` directory. The script examines all users in the Oracle Identity Manager database and generates new snapshots based on the new auditing level.

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 following aspects of the auditing engine:

- Level of detail for auditing
- Active triggers
- Behavior of the audit engine
- Data captured in the audit snapshot

You can specify the audit level as the value of the `XL.UserProfileAuditDataCollection` system configuration property in the Design Console.

See Also: The "System Configuration Form" section of *Oracle Identity Manager Design Console Guide*

The following are the supported audit levels:

1. **Process Task:** Audits the entire user profile snapshot with the resource life cycle process
2. **Resource Form:** Audits user record, group membership, provisioned resources, and any form data associated with the resource and process
3. **Resource:** Audits the user record, group membership, and provisioned resources
4. **Membership:** Only audits the user record and user group membership
5. **Core:** Only audits the user record
6. **None:** Audit data is not generated or 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.

2.1.2 Post-Processors

The `AUD` table stores the audit metadata XML. The audit engine uses this metadata to create the snapshot XML and the snapshot changes XML. The metadata XML provides, among other things, information about the table that stores the snapshot XML and snapshot changes XML and the post-processors. After the audit engine generates the snapshot, these post-processors process the snapshot data and change XML information, and store this information in the reporting tables.

There are two types of post-processors, the internal auditor type and custom type. Internal auditor post-processors are defined in the auditor XML metadata.

2.1.3 Creating Custom Post-Processors

You can create custom post-processors to extend the functionality of the existing post-processors so that additional information can be stored in the reporting tables. Custom post-processors are not defined in the XML metadata.

To create a custom post-processor:

See Also:

- [Appendix A, "Sample Code for a Custom Post-Processor"](#)
- The "Lookup Definition Form" section of *Oracle Identity Manager Design Console Guide*

1. Open the Lookup Definition form of the Design Console.
2. Create a class that extends the `CustomAuditDataProcessor` class and implements the `processAuditData` method.
3. Create a lookup definition in the Oracle Identity Manager Design Console as follows:
 - a. In the **Code** field, enter `Audit.AuditorName.CustomProcessors`. Here, *AuditorName* is the name of the auditor that the post-processor uses.

For example, for a user profile audit, the Code field value can be `Audit.UserProfile.CustomProcessors`.
 - b. Select the **Lookup Type** option.
 - c. Enter the group name related to the auditor, in this case, the `UPA Processors Group`.
 - d. Add the lookup code information.

This is the fully qualified classpath to the class you create. The code key and Decode should be the classpath.
4. After the class is created and the lookup information is set up, place the class in a JAR file in the `OIM_HOME/xellerate/JavaTasks` directory.

2.2 Tables Used for Storing Information About Auditors

Information about auditors is stored in the following tables of the database:

- `AUD`: This table stores information about all the auditors defined in Oracle Identity Manager.
- `aud_jms`: This table stores information about the changes made for an auditor.

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. This scheduled task is discussed in the ["Issuing Audit Messages"](#) section on page 2-3.

2.3 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 (`xlAuditQueue`) 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.

User Profile Auditing

User profile audits cover changes to user profile attributes, user membership, resource provisioning, access policies, and resource forms.

This chapter discusses the following topics:

- [Data Collected for Audits](#)
- [Post-Processor Used for User Profile Auditing](#)
- [Tables Used for User Profile Auditing](#)

3.1 Data Collected for Audits

By default, user profile auditing is enabled and the auditing level is set to `Resource Form` when you install Oracle Identity Manager with the Audit and Compliance module. This auditing level specifies the minimum level required for attestation of form data.

You configure the audit level in the System Properties page of the Design Console by using the `XL.UserProfileAuditDataCollection` keyword.

See Also:

The "[Audit Levels](#)" section on page 2-1 for more information about audit levels

The "System Properties" section in *Oracle Identity Manager Design Console Guide* for information about the `XL.UserProfileAuditDataCollection` keyword

This section discusses the following topics:

- [Capture and Archiving of User Profile Audit Data](#)
- [XML Representation of Snapshots and Changes to Snapshots](#)
- [Storage of Snapshots](#)
- [Trigger for Taking Snapshots](#)

3.1.1 Capture and Archiving of User Profile Audit Data

Each time a user profile changes, Oracle Identity Manager takes a snapshot of the user profile and stores the snapshot in an audit table in the database.

A snapshot is also generated when there is a change in a user profile that must be audited, even if an initial snapshot is missing. The current snapshot is treated as the initial snapshot.

The following are the components of a user profile and the tables that store these components:

- User Record: USR table, including all User Defined Fields (UDFs)
- User Group Membership: USG, UGP, and RUL tables
- User Policy Profile: UPP and UPD tables

Note: When you change a group name by using the Administrative and User Console, the User Profile Audit (UPA) tables in the database are not updated with the change until the next snapshot of the user.

- User Resource Profile: This component can be divided into the following subcomponents:
 - User Resource Instance: OIU, OBI , OST, and OBJ tables
 - Resource Lifecycle (Provisioning) Process: ORC, PKG, TOS, STA, OSI, SCH , MIL tables
 - Resource State (Process) Form: All tables that have names starting with UD_* (including child tables)

3.1.2 XML Representation of Snapshots and Changes to Snapshots

Oracle Identity Manager stores snapshots and changes to snapshots as XML in the UPA table. The following sections describe the XML representation of snapshots and changes to snapshots of user profiles:

- [XML Representation of User Profile Snapshots](#)
- [XML Representation of Changes to User Profile Snapshots](#)

3.1.2.1 XML Representation of User Profile Snapshots

The following elements constitute the XML representation of a user profile snapshot:

- `UserProfileSnapshot`

This is the topmost element in the XML representation. This element contains a user key and a version for each XML entry.

The remaining elements in this list are child elements of the `UserProfileSnapshot` element.
- `UserInfo`

This element contains general information about the user profile.
- `GroupMembership`

This element contains information about group membership of the user.
- `PolicyProfile`

This element contains information about the policy that allowed the provisioning of a specific resource to the user.
- `ResourceProfile`

This element contains the following elements:

- ResourceInstance: This element contains information about each resource that is provisioned to the user.
- ProcessData: This element contains information about the process form data stored in the UDFs.
- ObjectData: This element contains information about the object form data stored in the UDFs.

Example 3–1 is the XML representation of a sample user profile snapshot.

Example 3–1 XML Representation of a User Profile Snapshot

```
<?xml version="1.0" encoding="UTF-8"?>
- <UserProfileSnapshot key="202" version="1.0">
- <UserInfo>
  <Attribute name="Users.First Name">Testing02First</Attribute>
  <Attribute name="Users.Role">Full-Time</Attribute>
  <Attribute name="Users.Disable User">0</Attribute>
  <Attribute name="Users.Email">john.doe@acmetech.com</Attribute>
  <Attribute name="Users.Status">Active</Attribute>
  <Attribute name="Users.Update Date">2007-01-05 17:12:25.181</Attribute>
  <Attribute name="Users.User ID">TESTING02USER9</Attribute>
  <Attribute name="Users.Xellerate Type">End-User</Attribute>
  <Attribute name="Users.Last Name">Testing02Last</Attribute>
  <Attribute name="Users.Provisioned Date">2007-01-05 17:11:56.868</Attribute>
  <Attribute encrypted="true" name="Users.Password"
    password="true">8YxO3YSKDXJLmcsKeZhUSw == </Attribute>
  <Attribute name="Users.Creation Date">2007-01-05 17:11:56.868</Attribute>
  <Attribute name="Users.Lock User">0</Attribute>
  <Attribute key="1" name="Users.Updated By Login">XELSYSADM</Attribute>
  <Attribute name="Users.Password Reset Attempts Counter">0</Attribute>
  <Attribute key="1" name="Organizations.Organization Name">Xellerate Users
</Attribute>
  <Attribute name="Users.Login Attempts Counter">0</Attribute>
  <Attribute key="1" name="Users.Created By Login">XELSYSADM</Attribute>
</UserInfo>
- <GroupMembership>
- <Group key="3">
  <Attribute name="Groups-Users.Creation Date">2007-01-05 17:12:30.299
  </Attribute>
  <Attribute name="Groups-Users.Update Date">2007-01-05 17:12:30.299
  </Attribute>
  <Attribute name="Groups-Users.Membership Status">Active</Attribute>
  <Attribute key="1" name="Groups-Users.Updated By Login">XELSYSADM
  </Attribute>
  <Attribute name="Groups-Users.Membership Type">Direct</Attribute>
  <Attribute key="3" name="Groups.Group Name">ALL USERS</Attribute>
  <Attribute key="1" name="Groups-Users.Created By Login">XELSYSADM
  </Attribute>
</Group>
</GroupMembership>
- <PolicyProfile>
- <Policy key="1">
  <Attribute name="UPD_ALLOW_LIST">Res2</Attribute>
  <Attribute name="Access Policies.Key">1</Attribute>
  <Attribute name="Access Policies.Name">AP2</Attribute>
</Policy>
</PolicyProfile>
- <ResourceProfile>
```

```
- <ResourceInstance key="57">
  <Attribute name="Users-Object Instance For User.Creation Date">2007-01-05
    17:12:36.599 </Attribute>
  <Attribute key="45" name="Objects.Object Status.Status">Enabled</Attribute>
  <Attribute key="1" name="Access Policies.Name">AP2</Attribute>
  <Attribute key="6" name="Objects.Name">Res2</Attribute>
  <Attribute name="Users-Object Instance For User.Provisioned By Method">
    Access Policy</Attribute>
  <Attribute key="1"
    name="Users-Object Instance For User.Provisioned By Login">
    XELSYSADM</Attribute>
  <Attribute name="Users-Object Instance For User.Provisioned By ID">1
  </Attribute>
  <Attribute key="AP2" name="Access Policies.Key">1</Attribute>
</ObjectData>
- <Parent key="7">
- <FormInfo>
  <Attribute key="7" name="Structure Utility.Table Name">UD_PRC_PP</Attribute>
  <Attribute key="0" name="Structure Utility.Structure Utility Version Label.Version
  Label">Initial Version</Attribute>
</FormInfo>
- <Data key="162">
  <Attribute name="UD_PRC_PP_A">xxxxxxxxxx</Attribute>
</Data>
</Parent>
- <Children>
- <Child key="10">
- <FormInfo>
  <Attribute key="10" name="Structure Utility.Table Name">UD_PRC_CF</Attribute>
  <Attribute key="0" name="Structure Utility.Structure Utility Version Label.Version
  Label">Initial Version</Attribute>
</FormInfo>
- <Data key="162">
  <Attribute name="UD_PRC_CF_B">yyyyyyyyyy</Attribute>
</Data>
</Child>
</Children>
</ObjectData>
- <ProcessData>
- <Parent key="8">
- <FormInfo>
  <Attribute key="8" name="Structure Utility.Table Name">UD_RES2_PP
  </Attribute>
  <Attribute key="0" name="Structure Utility.Structure Utility Version
  Label.Version Label">Initial Version</Attribute>
</FormInfo>
- <Data key="54">
  <Attribute name="UD_RES2_PP_B">some_value1</Attribute>
  <Attribute name="UD_RES2_PP_A">some_value2</Attribute>
  <Attribute key="1" name="Access Policies.Name">AP2</Attribute>
</Data>
</Parent>
- <Children>
- <Child key="9">
- <FormInfo>
  <Attribute key="9" name="Structure Utility.Table Name">UD_RES2_CP
  </Attribute>
  <Attribute key="0" name="Structure Utility.Structure Utility Version
  Label.Version Label">Initial Version</Attribute>
</FormInfo>
```

```

-      <Data key="63">
        <Attribute name="UD_RES2_CP_C">Entry1C</Attribute>
        <Attribute name="UD_RES2_CP_D">Entry1D</Attribute>
        <Attribute key="1" name="Access Policies.Name">AP2</Attribute>
      </Data>
    </Child>
  </Children>
</ProcessData>
</ResourceInstance>
- <ResourceInstance key="74">
  <Attribute name="Users-Object Instance For User.Creation Date">2007-01-05
    17:22:37.597</Attribute>
  <Attribute key="33" name="Objects.Object Status.Status">Provisioning
</Attribute>
  <Attribute key="5" name="Objects.Name">Res1</Attribute>
  <Attribute name="Users-Object Instance For User.Provisioned By Method">
    Direct Provision</Attribute>
  <Attribute key="1" name="Users-Object Instance For User.Provisioned By
    Login">XELSYSADM</Attribute>
  <Attribute name="Users-Object Instance For User.Provisioned By ID">
    XELSYSADM</Attribute>
</ResourceInstance>
</ResourceProfile>
</UserProfileSnapshot>

```

3.1.2.2 XML Representation of Changes to User Profile Snapshots

Changes to the snapshot are stored in XML format. This XML information describes all changes that affect user profile attributes for a given transaction and the reason those changes were made.

The topmost element in this XML representation is **Changes**. Each change made during a particular transaction is described in a **Change** element. There may be multiple **Change** tags inside a **Changes** element. The following are attributes of the **Change** element:

- **reason**
This attribute holds the reason for the change in the user profile data.
- **reasonKey**
This attribute holds the key of the entity or the process that brought about the change in the user profile data.
- **where**
This attribute holds the location of the change.
- **action**
This attribute specifies whether the change is because of an insert, update, or a delete. The values are `insert`, `update`, and `delete`, respectively.
- **order**
This attribute specifies the order of the **Change** element in the **Delta** if there are more than one **Change** element.

Table 3–1 lists all possible values of the **reason** and **reasonKey** attributes.

Table 3–1 Values of the reason and reasonKey Attributes for User Profile Auditing

reason Attribute Value	reasonKey Attribute Value	Description
Reconciliation	Key of the reconciliation event (RCE_KEY value)	Change carried out through reconciliation
Access Policy	Key of the access policy (POL_KEY value)	Change carried out through a change in access policy
Request	Key of the request (REQ_KEY value)	Change carried out through a request
Direct Provision	Key of the user who performs direct provisioning (USR_KEY value)	Change carried out through direct provisioning
Manual	Key of the user who manually performs the change (USR_KEY value)	Change carried out manually by a user
Auto Group Membership	Key of the Auto Group Membership rule (RUL_KEY value)	Change carried out because of an update to the Auto Group Membership rule
Adapter	Key of the adapter (ADP_KEY value)	Change carried out when an adapter was run
API	Key of the user who performs the action that uses the API (USR_KEY value)	Change carried out through an API
Data Object	Key of the user who performs the action that carries out the data object change (USR_KEY value)	Change carried out at the data object level
Offline Processing	Key of the user who performs the offline processing action (USR_KEY value)	Change carried out during offline processing
Event Handler	Key of the event handler (EVT_KEY value)	Change carried out by the event handler
Attestation	Key of the attestation request (ATR_KEY value)	Change carried out through attestation
Unknown	0	Change that is not covered by any of the reason attribute values listed in this table
Regeneration	0	0 will be the value whenever the delta is created because of the execution of the GenerateSnapshot script. The value of changeReasonKey will always be 0 in this case.

[Example 3–2](#) is the XML representation of changes to a sample user profile snapshot.

Example 3–2 XML Representation of Changes to a Sample User Profile Snapshot

```
<?xml version="1.0" encoding="UTF-8"?>
- <Changes>
-   <Change action="insert" order="1" reason="Manual" reasonKey="1"
      where="/UserProfileSnapshot/ResourceProfile/ResourceInstance[@key='74']">
-     <Attribute name="Users-Object Instance For User.Creation Date">
-       <OldValue />
-       <NewValue>2007-01-05 17:22:37.597</NewValue>
-     </Attribute>
-     <Attribute name="Objects.Object Status.Status">
-       <OldValue key="" />
-       <NewValue key="35">Ready</NewValue>
-     </Attribute>
-     <Attribute name="Objects.Name">
-       <OldValue key="" />
-       <NewValue key="5">Res1</NewValue>
```



```

    </Attribute>
-   <Attribute name="Users-Object Instance For User.Provisioned By Method">
        <OldValue />
        <NewValue>Direct Provision</NewValue>
    </Attribute>
-   <Attribute name="Users-Object Instance For User.Provisioned By Login">
        <OldValue key=" " />
        <NewValue key="1">XELSYSADM</NewValue>
    </Attribute>
-   <Attribute name="Users-Object Instance For User.Provisioned By ID">
        <OldValue />
        <NewValue>XELSYSADM</NewValue>
    </Attribute>
</Change>
-   <Change action="update" order="2" reason="Manual" reasonKey="1"
    where="/UserProfileSnapshot/ResourceProfile/ResourceInstance[@key='74']">
-   <Attribute name="Objects.Object Status.Status">
        <OldValue key="35">Ready</OldValue>
        <NewValue key="33">Provisioning</NewValue>
    </Attribute>
</Change>
</Changes>

```

Information in this XML form is first stored in the UPA table and then stored in normalized form in the UPA_USR, UPA_FIELDS, UPA_RESOURCE, UPA_GRP_MEMBERSHIP, UPA_UD_FORMS, and UPA_UD_FORMFIELDS tables. Normalizing this data across multiple tables facilitates the retrieval of information for reporting purposes.

3.1.3 Storage of Snapshots

When Oracle Identity Manager takes a snapshot of a user profile, it stores the snapshot in the UPA table. The structure of the UPA table is described in [Table 3–2](#).

Table 3–2 Definition of the UPA Table

Column	Data Type	Description
UPA_KEY	NUMBER (19,0)	Key for the audit record
USR_KEY	NUMBER (19,0)	Key for the user whose snapshot is recorded in this entry
EFF_FROM_DATE	TIMESTAMP (6)	Date and time at which the snapshot entry became effective
EFF_TO_DATE	TIMESTAMP (6)	Date and time at which the snapshot entry was no longer effective In other words, this is the date and time at which the next snapshot entry was created. For the entry representing the latest user profile, the To Date column value is set to NULL.
SNAPSHOT	CLOB	XML representation of the snapshot
DELTAS	CLOB	XML representation of old and new values corresponding to a change made to the snapshot
SRC	VARCHAR2 (4000)	User ID of the user responsible for the change, and the API used to carry out the change
SIGNATURE	CLOB	Can be used by customers to store a digital signature for the snapshot (for nonrepudiation purposes)

3.1.4 Trigger for Taking Snapshots

When any data element in a user profile changes, Oracle Identity Manager creates a snapshot.

The following events trigger the creation of a user profile snapshot:

- Modification of any kind to the user record (for example, through reconciliation and direct provisioning)
- Group membership change for the user
- Changes in the policies that apply to the user
- Provisioning a resource to the user
- Deprovisioning of a resource for the user
- Any provisioning-related event for a provisioned resource:
 - Resource status change
 - Addition of provisioning tasks to the provisioning process
 - Updates to provisioning tasks in the provisioning process, for example, status changes, escalations, and so on
 - Creation of or updates to Process Form data
 - Creation of or updates to Object Form data

3.2 Post-Processor Used for User Profile Auditing

The user profile auditor has an internal post-processor that normalizes the snapshot XML into the reporting tables: UPA_USR, UPA_FIELDS, UPA_GRP_MEMBERSHIP, UPA_RESOURCE, UPA_UD_FORMS, and UPA_UD_FORMFIELDS. These tables are used by the reporting module to generate the appropriate reports.

3.3 Tables Used for User Profile Auditing

User profile audits use the following tables in the database:

The UPA table is the main table and stores all the snapshots and changes made to the user profiles. The audit engine reads data from the UPA table and normalizes it across the following reporting tables:

- UPA_USR: This table stores user profile information.
- UPA_FIELDS: This table stores user profile information in a vertical format.

This table has more information than the UPA_USR table. For instance, UD fields are stored in this table as well as other fields that are not available in UPA_USR.
- UPA_GRP_MEMBERSHIP: This table contains group membership for all the users in the system.

The information includes when a user was added and removed from a group.
- UPA_RESOURCE: The information in this table includes provisioned resources and changes in status for each of the resources.

This table does not include any form table information.
- UPA_UD_FORMS: Along with the UPA_UD_FORMFIELDS table, this table contains information about changes to the resource and process forms. It contains

information about the corresponding tables that are being changed. The actual field changes are stored in the UPA_UD_FORMFIELDS table.

- UPA_UD_FORMFIELDS : This table stores the names of form fields that are changed and the old and new values of the changed form fields. Whenever a form field is changed, a new row is inserted in this table to reflect the change.

Note: The UPA_UD_FORMS and UPA_UD_FORMFIELDS tables will be populated only if the `XL.EnableExceptionReports` system configuration property is set to `TRUE`. For more information about this property, see "[Exception Reports](#)" on page 5-4.

Group Profile Auditing

Group profile audits cover changes to group profile attributes, group administrators, and direct subgroups.

This chapter discusses the following topics:

- [Data Collected for Audits](#)
- [Tables Used for Group Profile Auditing](#)

4.1 Data Collected for Audits

Unlike user auditing, an independent audit level is not defined for group profile auditing. Instead, the audit levels defined for user profile auditing are used for group profile auditing. Group profile auditing takes place only if the audit level defined for user profile audit level is `Membership` or a level higher than that. By default, user profile auditing is enabled and the audit level is set to `Resource Form` when you install Oracle Identity Manager. As a result, group profile auditing is also enabled by default because the default audit level for user profile audit is `Resource Form`, which is higher than `Membership`.

This section discusses the following topics:

- [Capture and Archiving of Group Profile Audit Data](#)
- [XML Representation of Snapshots and Changes to Snapshots](#)
- [Storage of Snapshots](#)
- [Trigger for Taking Snapshots](#)

4.1.1 Capture and Archiving of Group Profile Audit Data

Each time a group profile changes, Oracle Identity Manager takes a snapshot of the group profile and stores the snapshot in an audit table in the database.

Oracle Identity Manager generates a snapshot when an audit is created for a group, even if an initial snapshot is missing. The current snapshot is treated as the initial snapshot.

The following are the components of a group profile and the tables that constitute these components:

- User Group Record: UGP table, including all UDFs for groups
- User group administrators: GPP table
- Subgroup information: GPG table

4.1.2 XML Representation of Snapshots and Changes to Snapshots

Oracle Identity Manager stores group snapshot data as XML in the Group Profile Audit (GPA) tables. The following sections describe the XML representation of snapshots and changes to snapshots:

- [XML Representation of Group Profile Snapshots](#)
- [XML Representation of Changes to Group Profile Snapshots](#)

4.1.2.1 XML Representation of Group Profile Snapshots

The following elements constitute the XML representation of a group profile snapshot:

- **GroupSnapshot**
This is the topmost element in the XML representation. This element contains a group key and a version for each XML entry. For a particular group profile, the value of the group key is fixed and the version number assigned to the snapshot is incremented by 1 for each new snapshot created for the group profile.
The remaining elements in this list are child elements of the GroupSnapshot element.
- **GroupInfo**
This element contains general group profile information.
- **GroupAdmin**
This element contains information about group administrators.
- **Subgroups**
This element contains information about subgroups.

[Example 4–1](#) is the XML representation of a sample group profile snapshot.

Example 4–1 XML Representation of a Group Profile Snapshot

```
<?xml version="1.0" encoding="UTF-8" ?>
- <GroupSnapshot key="311" version="1.0">
-   <GroupInfo>
-       <Attribute name="Groups.Creation Date">2007-04-12 17:27:17.231</Attribute>
-       <Attribute key="311" name="Groups.Group
Name">TESTGROUP100</Attribute>
-       <Attribute name="Groups.Update Date">2007-04-12 17:27:17.231</Attribute>
-       <Attribute key="1" name="UGP_UPDATEBY_LOGIN">XELSYSADM</Attribute>
-       <Attribute key="1" name="UGP_CREATEBY_LOGIN">XELSYSADM</Attribute>
-       <Attribute name="Groups.Group Status">Active</Attribute>
-   </GroupInfo>
-   <GroupAdmin>
-   <Group key="1">
-       <Attribute name="Groups-Group Ownership.Write">1</Attribute>
-       <Attribute name="Groups-Group Ownership.Creation Date">2007-04-12
17:27:17.356</Attribute>
-       <Attribute name="Groups.Key">311</Attribute>
-       <Attribute name="Groups-Group Ownership.Delete">1</Attribute>
-       <Attribute name="Groups-Group Ownership.Update Date">2007-04-12
17:27:17.356</Attribute>
-       <Attribute key="1" name="GPP_CREATEBY_LOGIN">XELSYSADM</Attribute>
-       <Attribute key="1" name="Groups.Group Name">SYSTEM
ADMINISTRATORS</Attribute>
-       <Attribute key="1" name="GPP_UPDATEBY_LOGIN">XELSYSADM</Attribute>
-   </Group>
```

```

- <Group key="312">
  <Attribute key="1" name="GPP_CREATEBY_LOGIN">XELSYSADM</Attribute>
  <Attribute key="312" name="Groups.Group Name">ADMINGROUP1</Attribute>
  <Attribute key="1" name="GPP_UPDATEBY_LOGIN">XELSYSADM</Attribute>
  <Attribute name="Groups-Group Ownership.Write">1</Attribute>
  <Attribute name="Groups-Group Ownership.Delete">1</Attribute>
</Group>
- <Group key="313">
  <Attribute key="1" name="GPP_CREATEBY_LOGIN">XELSYSADM</Attribute>
  <Attribute key="313" name="Groups.Group Name">ADMINGROUP2</Attribute>
  <Attribute key="1" name="GPP_UPDATEBY_LOGIN">XELSYSADM</Attribute>
  <Attribute name="Groups-Group Ownership.Write">1</Attribute>
  <Attribute name="Groups-Group Ownership.Delete">0</Attribute>
</Group>
</GroupAdmin>
- <Subgroups>
- <Group key="314">
  <Attribute name="Groups-User Sub Groups.Creation Date">2007-04-12
17:34:56.746</Attribute>
  <Attribute name="Groups-User Sub Groups.Update Date">2007-04-12
17:34:56.746</Attribute>
  <Attribute key="1" name="GPG_UPDATEBY_LOGIN">XELSYSADM</Attribute>
  <Attribute key="1" name="GPG_CREATEBY_LOGIN">XELSYSADM</Attribute>
  <Attribute key="314" name="Groups.Group Name">SUBGROUP100</Attribute>
</Group>
- <Group key="315">
  <Attribute name="Groups-User Sub Groups.Creation Date">2007-04-12
17:34:56.746</Attribute>
  <Attribute name="Groups-User Sub Groups.Update Date">2007-04-12
17:34:56.746</Attribute>
  <Attribute key="1" name="GPG_UPDATEBY_LOGIN">XELSYSADM</Attribute>
  <Attribute key="1" name="GPG_CREATEBY_LOGIN">XELSYSADM</Attribute>
  <Attribute key="315" name="Groups.Group Name">SUBGROUP101</Attribute>
</Group>
- <Group key="316">
  <Attribute name="Groups-User Sub Groups.Creation Date">2007-04-12
17:34:56.746</Attribute>
  <Attribute name="Groups-User Sub Groups.Update Date">2007-04-12
17:34:56.746</Attribute>
  <Attribute key="1" name="GPG_UPDATEBY_LOGIN">XELSYSADM</Attribute>
  <Attribute key="1" name="GPG_CREATEBY_LOGIN">XELSYSADM</Attribute>
  <Attribute key="316" name="Groups.Group Name">SUBGROUP102</Attribute>
</Group>
</Subgroups>
</GroupSnapshot>

```

4.1.2.2 XML Representation of Changes to Group Profile Snapshots

Changes to the snapshot are stored in XML format in the DELTAS column of the GPA table. This XML information describes all changes that affect group profile attributes for a given transaction and the reason those changes were made.

The topmost element in this XML representation is Changes. Each change made during a particular transaction is described in a Change element. There may be multiple Change elements inside a Changes element. The following are attributes of the Change element:

- reason

This attribute holds the reason for the change in the user profile data.

- `reasonKey`
This attribute holds the key of the entity or the process that brought about the change in the user profile data.
- `where`
This attribute holds the location of the change.
- `action`
This attribute specifies whether the change is because of an insert, update, or a delete. The values are `insert`, `update`, and `delete`, respectively.
- `order`
This attribute specifies the order of the Change element in the Delta if there are more than one Change element.

Table 4–1 lists all possible values of the `reason` and `reasonKey` attributes.

Table 4–1 Values of the `reason` and `reasonKey` Attributes for Group Profile Auditing

reason Attribute Value	reasonKey Attribute Value	Description
Reconciliation	Key of the reconciliation event (RCE_KEY value)	Change carried out through reconciliation
Access Policy	Key of the access policy (POL_KEY value)	Change carried out through a change in access policy
Request	Key of the request (REQ_KEY value)	Change carried out through a request
Direct Provision	Key of the user who performs direct provisioning (USR_KEY value)	Change carried out through direct provisioning
Manual	Key of the user who manually performs the change (USR_KEY value)	Change carried out manually by a user
Auto Group Membership	Key of the Auto Group Membership rule (RUL_KEY value)	Change carried out because of an update to the Auto Group Membership rule
Adapter	Key of the adapter (ADP_KEY value)	Change carried out when an adapter was run
API	Key of the user who performs the action that uses the API (USR_KEY value)	Change carried out through an API
Data Object	Key of the user who performs the action that carries out the data object change (USR_KEY value)	Change carried out at the data object level
Offline Processing	Key of the user who performs the offline processing action (USR_KEY value)	Change carried out during offline processing
Event Handler	Key of the event handler (EVT_KEY value)	Change carried out by the event handler
Attestation	Key of the attestation request (ATR_KEY value)	Change carried out through attestation
Unknown	0	Change that is not covered by any of the reason attribute values listed in this table
Regeneration	0	0 will be the value whenever the delta is created because of the execution of the <code>GenerateSnapshot</code> script. The value of <code>changeReasonKey</code> will always be 0 in this case.

[Example 4-2](#) is the XML representation of changes to a sample group profile snapshot.

Example 4-2 XML Representation of Changes to a Sample Group Profile Snapshot

```
<?xml version="1.0" encoding="UTF-8" ?>
- <Changes>
- <Change action="insert" order="1" reason="Manual" reasonKey="1"
where="/GroupSnapshot/Subgroups/Group[@key='314']">
- <Attribute name="GPG_CREATEBY_LOGIN">
  <OldValue key="" />
  <NewValue key="1">XELSYSADM</NewValue>
</Attribute>
- <Attribute name="GPG_UPDATEBY_LOGIN">
  <OldValue key="" />
  <NewValue key="1">XELSYSADM</NewValue>
</Attribute>
- <Attribute name="Groups-User Sub Groups.Creation Date">
  <OldValue />
  <NewValue>2007-04-12 17:34:56.746</NewValue>
</Attribute>
- <Attribute name="Groups.Key">
  <OldValue />
  <NewValue>311</NewValue>
</Attribute>
- <Attribute name="Groups-User Sub Groups.Update Date">
  <OldValue />
  <NewValue>2007-04-12 17:34:56.746</NewValue>
</Attribute>
- <Attribute name="Groups.Group Name">
  <OldValue key="" />
  <NewValue key="314">SUBGROUP100</NewValue>
</Attribute>
</Change>
- <Change action="insert" order="2" reason="Manual" reasonKey="1"
where="/GroupSnapshot/Subgroups/Group[@key='314']">
- <Attribute name="Groups-User Sub Groups.Creation Date">
  <OldValue />
  <NewValue>2007-04-12 17:34:56.809</NewValue>
</Attribute>
- <Attribute name="Groups.Key">
  <OldValue />
  <NewValue>311</NewValue>
</Attribute>
- <Attribute name="Groups-User Sub Groups.Update Date">
  <OldValue />
  <NewValue>2007-04-12 17:34:56.809</NewValue>
</Attribute>
</Change>
- <Change action="insert" order="3" reason="Manual" reasonKey="1"
where="/GroupSnapshot/Subgroups/Group[@key='315']">
- <Attribute name="GPG_UPDATEBY_LOGIN">
  <OldValue key="" />
  <NewValue key="1">XELSYSADM</NewValue>
</Attribute>
- <Attribute name="GPG_CREATEBY_LOGIN">
  <OldValue key="" />
  <NewValue key="1">XELSYSADM</NewValue>
</Attribute>
- <Attribute name="Groups.Group Name">
  <OldValue key="" />
```

```

        <NewValue key="315">SUBGROUP101</NewValue>
      </Attribute>
    </Change>
  - <Change action="insert" order="4" reason="Manual" reasonKey="1"
where="/GroupSnapshot/Subgroups/Group[@key='314']">
  - <Attribute name="Groups-User Sub Groups.Creation Date">
    <OldValue />
    <NewValue>2007-04-12 17:34:56.871</NewValue>
  </Attribute>
  - <Attribute name="Groups.Key">
    <OldValue />
    <NewValue>311</NewValue>
  </Attribute>
  - <Attribute name="Groups-User Sub Groups.Update Date">
    <OldValue />
    <NewValue>2007-04-12 17:34:56.871</NewValue>
  </Attribute>
</Change>
- <Change action="insert" order="5" reason="Manual" reasonKey="1"
where="/GroupSnapshot/Subgroups/Group[@key='316']">
- <Attribute name="GPG_UPDATEBY_LOGIN">
  <OldValue key=" " />
  <NewValue key="1">XELSYSADM</NewValue>
</Attribute>
- <Attribute name="GPG_CREATEBY_LOGIN">
  <OldValue key=" " />
  <NewValue key="1">XELSYSADM</NewValue>
</Attribute>
- <Attribute name="Groups.Group Name">
  <OldValue key=" " />
  <NewValue key="316">SUBGROUP102</NewValue>
</Attribute>
</Change>
</Changes>

```

4.1.3 Storage of Snapshots

When Oracle Identity Manager takes a snapshot of a group profile, it stores the snapshot in a GPA table. The structure of this table is as described in [Table 4–2](#).

Table 4–2 Definition of the GPA Table

Column	Data Type	Description
GPA_KEY	NUMBER (19,0)	Key for the audit record
UGP_KEY	NUMBER (19,0)	Key for the group whose group snapshot is recorded
EFF_FROM_DATE	TIMESTAMP (6)	Date and time at which the snapshot entry became effective
EFF_TO_DATE	TIMESTAMP (6)	Date and time at which the snapshot entry was no longer effective In other words, this is the date and time at which the next snapshot entry was created. For the entry representing the latest user profile, the To Date column value is set to NULL.
SRC	VARCHAR2 (4000)	Source of the entry, which is the group name and the API used
SNAPSHOT	CLOB	XML representation of the snapshot
DELTAS	CLOB	XML representation of old and new values corresponding to a change made to the snapshot

Table 4–2 (Cont.) Definition of the GPA Table

Column	Data Type	Description
SIGNATURE	CLOB	Can be used by customers to store a digital signature for the snapshot (for nonrepudiation purposes)

4.1.4 Trigger for Taking Snapshots

When any data element in the group profile snapshot changes, Oracle Identity Manager creates a snapshot.

The creation of group profile snapshots is triggered by events that result in changes in any of the following:

- Group profile data
- Subgroup information
- Group administrators

4.2 Tables Used for Group Profile Auditing

The GPA table stores all the snapshots and changes made to the group profiles.

Oracle Identity Manager Reporting

Oracle Identity Manager includes a custom reporting engine that enables you to run predefined reports against the Oracle Identity Manager transactional database or a secondary database if one is configured. You can add new reports without editing any Java code, and you can obtain reporting data by invoking a stored procedure.

This chapter discusses the following topics:

- [Reporting Features](#)
- [List of Reports](#)
- [Exception Reports](#)
- [How to Create a Report](#)
- [Working with Third-Party Reporting Tools](#)
- [Date Ranges in Historical Reports](#)
- [Limitations](#)

Note: The Oracle Identity Manager reporting engine is not meant to be a replacement for enterprise reporting solutions. The Oracle Identity Manager reporting engine is not optimized for very large data volume and does not provide the rich features you would find in an enterprise reporting application.

For large-scale deployments, especially those taking advantage of the extensive auditing capabilities of Oracle Identity Manager, it is highly recommended that you deploy a dedicated enterprise-class reporting solution. A solution based on tools such as Oracle Business Intelligence Enterprise Edition can provide the flexibility, automation, and performance required for a large-scale enterprise.

5.1 Reporting Features

The following are Oracle Identity Manager reporting features:

- Select and view reports from a predefined list in the Administrative and User Console.
- Use a delegated administration model to control the reports available to a user and the information included in those reports.
- Filter report information.
- View reports on-screen.

- Export reports to CSV files.
- Provide interactive reports.
- Run reports from a secondary database.
- Provide support for Crystal Reports.

The following sections explain data storage for reporting at the data, XML, and API layers in Oracle Identity Manager.

- [Data Layer](#)
- [XML Metadata](#)
- [API Layer](#)

5.1.1 Data Layer

You can change the database schema and add stored procedures in the data layer.

The REP and RPG tables support reporting. The REP table contains the following:

- A list of all reports in the system
- The report name
- The report code
- The report type
- The report description
- The name of the stored procedure for the report
- The name of the data source
- The maximum report size
- The number of filters to be displayed on the report page
- XML metadata for each report

The RPG table is a link table between the REP and UGP table. This table stores information about the reports and the groups that have access permissions for the reports.

Each report is associated with a stored procedure. To run a report, you run the associated stored procedure with relevant arguments. You cannot run a report based on a database query.

There can be many reports in the system. As a result, the stored procedure follows rules to enable the report to be invoked generically. See ["How to Create a Report"](#) on page 5-8 for details.

Each stored procedure has a set of required generic parameters. These parameters provide standard information, for example, starting row, page size, filter columns, and so on. The stored procedure can also have any number of report-specific parameters. Each stored procedure returns two values: a result set representing a page of the entire report data, and a total count for the report data. The standard format for a stored procedure and the XML metadata enable you to add and run any report without changing any Java code.

5.1.2 XML Metadata

XML metadata for each report is stored in the REP table for the report. The metadata provides the following information for the report:

- Layout information
- Representation of all the input parameters and their association with the corresponding stored procedure parameters
- Support for user-defined parameters (operational reports only)
- Display information for each report input parameter, for example, a field label or field type (for example, `TextField`, `LookupField`, and so on)
- Location of each column on the report display page
- Display information for each report data column
- Columns to be included in the filter drop downs
- Clickable columns for interactive reports

For details on the metadata structure, see ["How to Create a Report"](#) on page 5-8.

5.1.3 API Layer

The API layer provides all back-end reporting functionality. The back end is not tied to the front end. You can create custom user interfaces by using the reporting APIs.

5.2 List of Reports

This section lists the operational and historic reports that are available.

[Table 5–1](#) lists the names and description of the operational reports.

Table 5–1 List of Operational Reports

Name	Description
Entitlements Summary	Lists the number of users for each status within each resource.
Policy List	Displays a snapshot of all policies defined within the system.
Delegated Administrators By Organization	Lists all the delegated administrator user groups for organizations.
Attestation Requests by Reviewer	Lists attestation requests by reviewer.
Approval Status By Approver	Provides a summary of all approval tasks.
User Resource Access	Lists the access rights to resources for selected users.
Resource Access List	Lists all users who have access to a selected resource.
Policy Detail	Lists complete details about specific policies defined within the system.
Group Membership Profile	Lists the number of users in different numbers of groups.
OIM Password Expiration	Lists users whose Oracle Identity Manager passwords are about to expire.
Group Membership	Provides a snapshot of users in each group.
Resource Password Expiration	Lists users whose resource passwords are about to expire (as determined by Oracle Identity Manager).
Organization Structure	Lists the hierarchical organization structure and user memberships.

Table 5–1 (Cont.) List of Operational Reports

Name	Description
Requests Initiated	Lists all requests initiated in a specified time interval.
Requests Details By Status	Returns details of all requests with a specified status.
Attestation Process List	Provides a snapshot of all defined attestation processes.
Attestation Requests by Process	Lists attestation requests by process.
Attestation Request Detail	Lists complete details of selected attestation requests.
Financially Significant Resources	Lists complete details of financially significant resources.
Delegated Administrators & Permissions By Organization	Lists all administrator user groups and permissions for organizations.
Delegated Administrators & Permissions By Resource	Lists all administrator user groups and permissions for resources.
Delegated Administrators By Resource	Lists all administrator and authorizer user groups for resources.

Table 5–2 lists the names and description of the historical reports that are available.

Table 5–2 List of Historical Reports

NAME	DESCRIPTION
User Resource Access History	Returns the history of a user's resource access.
Resource Access List History	Returns a history of all users who have had access to a selected resource.
User Profile History	Returns the history of a user's profile.
User Membership History	Returns the history of a user's memberships in a user group.
Group Membership History	Returns the history of a group's memberships.
Resource Activity	Returns the history of all provisioning and approval activities for a resource.
Task Assignment History	Returns the history of all task assignment based on the tasks.
Password Reset Success Failure	Returns the password change metrics for Oracle Identity Manager users.
Account Activity In Resource	Lists all account activities in each resource.
Rogue Accounts By Resource	Lists all the rogue accounts in each resource.
Fine Grained Entitlement Exceptions By Resource	Lists all fine-grained entitlement exceptions by a resource.
Users Created	Lists all users created in a specified time interval.
Users Deleted	Lists all users deleted in a specified time interval.
Users Disabled	Lists all users disabled in a specified time interval.
Users Unlocked	Lists all users unlocked in a specified time interval.

See Also: These reports are also listed in Chapter 14 of the *Oracle Identity Manager Administrative and User Console Guide*.

5.3 Exception Reports

In Oracle Identity Manager, **exception** refers to the difference between accounts that a user is entitled to and the accounts that are actually assigned to a user. The user is assigned these accounts as a result of access policies, provisioning of resources,

approval requests, and reconciliation events. Any difference of these accounts assigned to a user in the target system and the ones assigned to the user in Oracle Identity Manager comprises an exception.

The following exception reports have been introduced in this release:

- **Rogue Accounts By Resource**

This report returns a list of all the rogue accounts existing in a resource. The following exceptions are reported:

- An account that exists in the target system but is not provisioned to the corresponding user in Oracle Identity Manager
- An account that exists in the target system but has been deprovisioned for the corresponding user in Oracle Identity Manager
- An account that exists in the target system but the corresponding user to whom the account is provisioned has been deleted in Oracle Identity Manager

The following exception is not reported in spite of being a rogue account:

- An account that exists in the target system but the corresponding user to whom the account is provisioned has never existed in Oracle Identity Manager

- **Fine Grained Entitlement Exceptions By Resource**

This report returns a list of all the accounts in a resource for which the process form data being reconciled is different from the expected values. It means that this report returns any account existing in the target system that is also provisioned to the corresponding user in Oracle Identity Manager, but for which the process data does not match.

Note:

- After completion of initial target reconciliation, all account-related activities performed directly on a target resource are tracked as exception activity. Account-related activities include account creation, account modification, and entitlement assignment/revocation. The exception reports should be used only if the organization policies enforce that all account-related activities in target resources would always be initiated in Oracle Identity Manager. In addition, remember that exception detection and recording are an extension of account data reconciliation and, therefore, may result in a drop in performance during reconciliation.
 - Both the exception reports depend on reconciliation data. Therefore, these reports will not display any data if the corresponding reconciliation events are archived.
-
-

By default, these two exception reports are not enabled. In order to enable them, you must set the value of the `XL.EnableExceptionReports` property to `TRUE` in the system configuration. If the value of the property is `FALSE` (default value), then the exception reports will not be available on the Historical Reports page of the Administrative and User Console. As a result, the data for the exception reports will not be populated in the `RCX`, `UPA_UD_FORMS`, and `UPA_UD_FORMFIELDS` tables. These reports rely on reconciliation data to be populated.

Suppose you do not enable these exception reports immediately after Oracle Identity Manager installation. Instead, you enable them sometime after you start using Oracle Identity Manager. In such a scenario, the UPA_UD_FORMS, and UPA_UD_FORMFIELDS tables must be populated with baseline form data before you can run the exception reports. To do this, you must use the UPA Form Data Upgrade Utility. See ["Using the UPA Form Data Upgrade Utility"](#) on page 5-6 for instructions.

5.3.1 Using the UPA Form Data Upgrade Utility

The UPA Form Data Upgrade Utility populates the UPA_UD_FORMS and UPA_UD_FORMFIELDS tables with baseline data that acts as a starting point for the exception reports. You must run this utility whenever you enable the exception reporting feature. This utility runs through all the entitlements. For provisioning instances that have a resource with either a process or an object form attached, the utility reads the data on the form and stores them in the two database tables. This forms the baseline. Subsequent changes to the form are automatically captured because exception reporting is enabled. With the baseline available, the exception reports start reporting on exceptions. The utility creates a log file named `UPA_Form_Data_Upgrade_Timestamp.log` after the process is complete.

To use the UPA Form Data Upgrade Utility:

1. Copy the UPA Form Data Upgrade Utility files to your local computer.

Before using the utility, copy the `UPAFormDataUpgradeUtility` directory from the following directory on the installation media to your local computer:

For Oracle Database:

```
installServer/Xellerate/db/oracle/Utilities/UPAFormDataUpgradeUtility
```

Note: In the rest of this procedure, the full path and name of the local directory containing the UPA Form Data Upgrade Utility files is referred to as `UPAFORMDATAUPGRADEUTILITY`.

2. Configure the scripts.

Settings for the Oracle Database Batch/Shell file:

- For Microsoft Windows: Edit the following file:

```
UPAFORMDATAUPGRADEUTILITY\UPAFormDataUpgrade.bat
```

- For Linux or UNIX: Edit the following file:

```
UPAFORMDATAUPGRADEUTILITY/UPAFormDataUpgrade.sh
```

[Table 5–3](#) shows the values of the variables that must be set before you run the utility:

Table 5–3 Variables of the UPA Form Data Upgrade Utility

Variables	Description
ORACLE_HOME	Oracle home directory.
OIM_DB_REMOTE	Describes if the database is running on a remote computer. Set a value for this parameter only if OIM_DB_REMOTE = Y or OIM_DB_REMOTE = N.

Table 5–3 (Cont.) Variables of the UPA Form Data Upgrade Utility

Variables	Description
OIM_DB_ORACLE_SID	SID of the database. Set a value for this parameter only if OIM_DB_REMOTE = N.
OIM_DB_SERVICE_NAME	TNS service name that points to the remote database. Set a value for this parameter only if OIM_DB_REMOTE = Y.

3. Compile the stored procedure.

For Oracle Database, perform the following steps:

- a. Log in to SQL*Plus with the credentials of the Oracle Identity Manager release database schema owner.

- b. Open the following file in a text editor:

```
UPAFORMDATAUPGRADEUTILITY/compile_all_XL_SP_UPA.sql
```

- c. Make the following changes in the compile_all_XL_SP_UPA.sql script:

- Change:

```
@@XL_SP_LoadFormData.sql
```

To:

```
@UPAFORMDATAUPGRADEUTILITY/XL_SP_LoadFormData.sql
```

- Change:

```
@@XL_SP_UPA_FormData_Upgrade.sql
```

To:

```
@UPAFORMDATAUPGRADEUTILITY/XL_SP_UPA_FormData_Upgrade.sql
```

- d. Run the compile_all_XL_SP_UPA.sql script.

```
UPAFORMDATAUPGRADEUTILITY/compile_all_XL_SP_UPA.sql
```

4. Run the utility.

- For Oracle Database on Linux or UNIX:

Run the UPAFormDataUpgrade.sh file from the following location:

```
UPAFORMDATAUPGRADEUTILITY/UPAFormDataUpgrade.sh OIM_DB_User_ID
OIM_DB_User_Password
```

Note: On UNIX, you might also want to clear the command history of the shell by running the following command:

```
history -c
```

- For Oracle Database on Microsoft Windows:

Run the UPAFormDataUpgrade.bat file from the following location:

```
UPAFORMDATAUPGRADEUTILITY\UPAFormDataUpgrade.bat OIM_DB_User_ID
OIM_DB_User_Password
```

5.4 How to Create a Report

The following are tasks for creating a new report:

- [Writing the Stored Procedure](#)
- [Creating the Report XML Metadata](#)
- [Modifying Property Files for Translating Label Names, Report Names, and Report Descriptions](#)
- [Creating an REP Entry and Providing Access to the Report](#)

5.4.1 Writing the Stored Procedure

Each report is based on a single stored procedure. The following are rules for the stored procedure:

- Use a stored procedure, not a user-defined function.
- A stored procedure returns two values: the report data result set and the total number of rows in the report.
- The report result set is paged.

The result set that is returned when you run the stored procedure represents one page of the entire report data. The starting row and the size of the page is specified at the time of running the stored procedure.

- The stored procedure handles filter parameters and user defined input parameters.

Each stored procedure uses generic parameters, and each can use parameters specific to the stored procedure. These parameter types are described in the following sections.

5.4.1.1 Generic Parameters

Generic parameters are common to all the stored procedures. You specify generic parameters in a required sequence before any specific parameters.

There are twelve generic parameters. All twelve generic parameters are required, even if their values are null. The following are the generic parameters in the order that you must specify them:

1. Report Result Set (type=cursor, OUT): The result set that represents the report data.
2. User Key (type=int, IN): The key of the user who runs the report.
This user key ensures that only records are returned for which the user has read permissions.
3. Sort Columns (type=varchar, IN): A list of comma-delimited column names on which the report result set can be sorted.
Reserved for future use.
4. Sort Order (type=varchar, IN): The sort order (ascending or descending) for the report result set.
Reserved for future use.
5. Start Row (type=int, IN): The row number where the result set starts.
6. Page Size (type=int, IN): The size of the result set, or the number of entries in a single page in a multi-page report.

7. **Do Count** (type=int, IN): Can have values 0, 1, or 2.

A value of 0 indicates that the result set is computed and returned, but the total number of rows in the entire report data is not computed. A value of 1 indicates that the result set and the total number of rows are computed. When the value is 2, only the total number of rows is computed and an empty result set is returned.

8. **Total Rows** (type=int, OUT): This is an OUT parameter that returns the total number of rows when the value of the *Do Count* variable is either 1 or 2.

Because the report data is paged, the value of the total number of rows is not the size of the returned result set. It is the size of the entire report.

For example, suppose that a stored procedure returns a list of all users. There are 200 users in the system, the start row=1, and the page size=50. For this stored procedure, the size of the result set is 50, but the value of the total rows OUT parameter is 200.

9. **Filter Column Names** (type=varchar, IN): This is a comma-delimited list of column names on which the report data can be filtered.

A stored procedure has no way of knowing the alias to use for the listed columns. The stored procedure expects that the column names in this list are correctly qualified with the appropriate table aliases, if needed, for example:

```
usr.usr_first_name,obj.obj_name
```

10. **Filter Column Values** (type=varchar, IN): This is a comma-separated list of column values that have a one-to-one correspondence with the column names listed in the previous parameter.

If the previous parameter contains a comma-separated list with two column names, this parameter is a comma separated list of two values. You can use a wild card (%) character, for example, Jo%, Laptop.

11. **User-Defined Column Names** (type=varchar, IN): This is a comma-separated list of column names that represent user-defined columns on system forms.

These names must be appropriately aliased if needed, for example:

```
USR.USR_UDF_SSN
```

12. **User-Defined Column Values** (type=varchar, IN): This is a comma-delimited list of column values for user defined fields.

These values have a one-to-one correspondence with the column names listed in the previous parameter. You can use the wild card (%) character, for example:

```
1234567890
```

5.4.1.2 Specific Parameters

Specific parameters are specific to each report. These parameters have a one-to-one correspondence with the report input parameters, except for the date range input parameter and user-defined parameters.

You must add specific parameters after the generic parameters. Each specific parameter represents one report input parameter on the Report Input page.

All specific parameters that are of the `varchar2` type support the wild card (%) character.

5.4.1.3 Other Stored Procedure Notes

Each time an error is encountered, an exception is thrown. The exception contains an error code. The calling Java code receives the error as a `SQLException` with the error code embedded in it. The stored procedure checks the code for errors based on the following rules:

- The value of Start Row cannot be 0 or null.
- The value of Page Size cannot be 0 or null.
- The value of User Key cannot be 0 or null.
- The value of Do Count can only be 0, 1 or 2.
- There is a one-to-one mapping between Filter Column Names and Filter Column Values.
- There is a one-to-one mapping between User-Defined Column Names and User-Defined Column Values.

Even if there is no data to return for a report, an empty result set is returned.

5.4.1.4 Example of a Stored Procedure Signature

[Example 5-1](#) shows the signature for the User Resource Access report stored procedure for Oracle Database:

Example 5-1 Signature of the User Resource Access Stored Procedure for Oracle Database

```
PROCEDURE XL_SP_UserResourceAccess (
  csrresultset_inout      IN OUT  sys_refcursor,
  intuserkey_in           IN      NUMBER,
  strsortcolumn_in        IN      VARCHAR2,
  strsortorder_in         IN      VARCHAR2,
  intstartrow_in          IN      NUMBER,
  intpagesize_in          IN      NUMBER,
  intdocount_in           IN      NUMBER,
  inttotalrows_out        OUT     NUMBER,
  strfiltercolumnlist_in  IN      VARCHAR2,
  strfiltercolumnvaluelist_in IN  VARCHAR2,
  strudfcolumnlist_in     IN      VARCHAR2,
  strudfcolumnvaluelist_in IN  VARCHAR2,
  struserlogin_in         IN      VARCHAR2,
  strfirstname_in         IN      VARCHAR2,
  strmiddlename_in        IN      VARCHAR2,
  strlastname_in          IN      VARCHAR2,
  struseremail_in         IN      VARCHAR2,
  strorgname_in           IN      VARCHAR2,
  strusergroup_in         IN      VARCHAR2,
  strmgrfirstname_in      IN      VARCHAR2,
  strmgrlastname_in       IN      VARCHAR2,
  struserstatus_in        IN      VARCHAR2,
  struseremptytype_in     IN      VARCHAR2
)
```

In [Example 5-1](#), the first twelve parameters (upto `strudfcolumnvaluelist_in`) are generic parameters. The remaining parameters are specific parameters.

5.4.2 Creating the Report XML Metadata

After you create the stored procedure, you create the XML metadata for the report. All report-specific information is included in the metadata so that the report can be run and displayed correctly. The report metadata provides information such as attributes of the report-specific input parameters, the display properties of the report input parameters and the display properties of the report data, for example, report layout, display labels, and so on.

The root element of the metadata is `report`. This element provides the layout of the report. Three display layouts are supported: Tabular layout, Sectional layout, and Sectional with Report Header layout.

The `report` element has two child elements: `StoredProcedure` and `ReturnColumns`.

5.4.2.1 StoredProcedure Element

The `StoredProcedure` element provides information about the stored procedure-specific parameters and user defined fields. It consists of a single `InputParameters` element that contains multiple `InputParameter` elements.

Each specific stored procedure parameter corresponds to one input parameter on the Report Input page. The exception to this rule is the `DateRange` field, which is represented by two stored procedure parameters. Each input parameter is represented by one `InputParameter` element. The Report Input page can also contain user-defined fields from any system form. Each user-defined field on the Report Input page is also represented by one `InputParameter` element. The number of user-defined fields can change, but the number of stored procedure input parameters does not change each time. User-defined fields are represented by comma-delimited lists.

For example, suppose a report must support seven input parameters. Two parameters are user-defined fields, and five are specific parameters in the signature of the stored procedure (in addition to the twelve generic parameters). These are represented by five `InputParameter` tags. The two user-defined parameters are also represented by two `InputParameter` tags, but they do not have corresponding parameters in the stored procedure signature. Instead, they are passed as comma-delimited lists of column names and their values. Therefore, the metadata must contain a total of seven `InputParameter` tags.

If you want to support the `DateRange` input type, the Report Input page must be supported by two stored procedure parameters: one for the From date and the other for the To date.

The following are the attributes of the `InputParameter` element:

- `name` (required:Yes): The name of the input parameter.
In case of non-user-defined input parameters, this value can be anything. However, for clarity, it should match the name of the corresponding stored procedure input parameter. For user-defined input parameters, this name is the column name of the user-defined column, prefixed by the required table alias, if needed.
- `parameterType` (required:Yes): The SQL data type of the corresponding stored procedure parameter.
In case of user-defined input parameters, this value is `varchar`.
- `order` (required:Yes): The order of the report-specific input parameters.

The ordering starts from 1. You must list the regular input parameters, and the user-defined input parameters later.

- `fieldType` (required: Yes): The type of the display field on the Report Input page.
The supported input types are: `TextField`, `Date`, `DateRange`, `LookupField`, and `Combobox`.
- `fieldLabel` (required: Yes): The property value of the field label for this field.
The property value is a value from the message resources property file (`xlWebAdmin.properties` in this case), which represents the actual label.
- `allowedValues` (required: No, unless `fieldType` is `Combobox`): Lookup codes associated with a combo box.
- `required` (required: No, default: false): If set to true, then the user must provide a value for this field for the report to run.
- `udf` (required: No, default: false): For a user defined field that is represented by the `InputParameter` element, this attribute must be present and have a value of true. Operational reports only.

If the attribute `fieldType` has a value of `LookupField`, a child element called `ValidValues` must be under the `InputParameter` element. The reporting functionality supports three types of lookups: Lookup by code, lookup by method, and lookup by column. The following are the supported attributes of the `ValidValues` element:

- `lookupCode` (required: No): Must be present only if the lookup is by code.
If it is, the value of this attribute is the lookup code.
- `lookupColumn` (required: No): Must be present only if the lookup is by column.
If it is, the value of this attribute is the column code of the lookup column.
- `lookupMethod` (required: No): Must be present only if the lookup is by class or method.
If it is, the value of this attribute is the name of method that provides the lookup values.
- `operationClass` (required: No): Must be present only if the `lookupMethod` attribute is present.
The value of this attribute is the fully qualified name of the class that contains the lookup method.
- `displayColumns` (required: No): Must be present only if the `lookupMethod` attribute is present.
It is a comma-delimited list of column codes which represent columns that is displayed in the lookup.
- `selectionColumn` (required: No): Must be present only if the `lookupMethod` or `lookupColumn` attributes are present.
It represents the column code of the column, the value of which is saved in the database.

Examples of `InputParameter` tags

- Regular `TextField` input parameter:

```
<InputParameter name="strfirstname_in" parameterType="varchar2" order="2"
fieldType="TextField" fieldLabel="report.userResourceAccess.label.firstName"
```



```
required="false" />
```

- **User-Defined input parameter:**

```
<InputParameter name="USR.USR_UDF_SSN" parameterType="varchar2" order="11"
fieldType="TextField" fieldLabel="report.userResourceAccess.label.SSN"
required="false" udf="true" />
```

- **Input parameter of type Combobox:**

```
<InputParameter name="struserstatus_in" parameterType="varchar2" order="10"
fieldType="Combobox" allowedValues="Lookup.WebClient.Users.Status"
fieldLabel="report.userResourceAccess.label.userStatus" required="false" />
```

- **Input parameter of type LookupField with lookupCode:**

```
<InputParameter name="struseremptytype_in" parameterType="varchar2" order="11"
fieldType="LookupField"
fieldLabel="report.userResourceAccess.label.employeeType" required="false" >
  <ValidValues lookupCode="Lookup.Users.Role"/>
</InputParameter>
```

- **Input parameter of type LookupField with lookupColumn**

```
<InputParameter name="struseremail_in" parameterType="varchar2" order="5"
fieldType="LookupField" fieldLabel="report.userResourceAccess.label.userEmail"
required="false" >
  <ValidValues lookupColumn="Users.Xellerate Type" selectionColumn="Lookup
Definition.Lookup Code Information.Decode" />
</InputParameter>
```

- **Input parameter of type LookupField with lookupMethod**

```
<InputParameter name="struserlogin_in" parameterType="varchar2" order="1"
fieldType="LookupField" fieldLabel="report.userResourceAccess.label.userLogin"
required="false" >
  <ValidValues lookupMethod="findUsersFiltered"
    operationClass="Thor.API.Operations.tcUserOperationsIntf"
    displayColumns="Users.User ID,Users.Last Name,Users.First Name"
    selectionColumn="Users.User ID"/>
</InputParameter>
```

If the attribute `fieldType` has a value of `DateRange`, you must include the two child elements `InputStartDate` and `InputEndDate`. Each element must have the following attributes:

- **name (required: Yes):** The name of the parameter.

This should match the name of the stored procedure parameter that represents this date parameter.

- **parameterType (required: Yes):** The SQL type of the stored procedure parameter that represents this date parameter.
- **order (required: Yes):** The order of the stored procedure parameter
- **defaultValue (required: No, default: 01/01/1900 and 12/31/2049):** The default value to provide if the user does not enter a date in the start or end date fields.
- **format (required: No, default: `reports.generic.message.internalDateFormat`):** The format of the default date.

5.4.2.2 ReturnColumns tag

The `ReturnColumns` tag represents the list of all the columns that are being returned by the result set. This tag contains multiple `ReturnColumn` tags, each of which represents one column in the returned result set. The attributes of the `ReturnColumn` tag provides information that is useful for displaying the report data.

The following are the attributes of the `ReturnColumns` tag:

- **name** (required: Yes): This name represents the column code of the result set column that is represented by this particular tag.

If the column code is not available, it can be the alias or the column name.
- **label**: (required: Yes): This provides the property value of the column header/label for this column.

The property value is a value from the message resources property file (`xlWebAdmin.properties` in this case), which represents the actual label.
- **position** (required: Yes): This attribute can have three values: `Table`, `Sectional Header`, or `Report Header`.

This attribute specifies the location of each column. Each column can reside in the table (for any layout), the section header (for Sectional Layout and Sectional with Report Header Layout), or the report header (in case of Sectional with Report Header layout).
- **filterColumn** (required: No, default: false): This attribute specifies whether the column is a filter column.

If the value is true, then the column name is included in the filter drop down lists at the top of the Report Display page.
- **filterColumnName** (required: No, unless `filterColumn` is present): This attribute represents the actual name of the column prefixed by the table alias, if needed.
- **filterType** (required: No, unless `filterColumn` is false): The type of the filter display field on the Report result page. The only supported input type is `Combobox`.
- **filterLookupKey** (required: No, unless `filterColumn` is false): Represents a lookup code associated with a combo box that is shown for a filter display field on the Report result page.
- **clickable** (required: No, default: false): This attribute specifies whether the column value is a link.

This attribute provides support for action-ability of reports.

You can configure interactive reports in the report module. In an interactive report, you define column values to be links that take the user to a page in or outside of the Administrative and User Console when he or she clicks the column value. To configure the links, you add information to the metadata so that the user is taken to the appropriate page. The links can have dynamic or static locations. Clicking on a link opens a new browser window that shares the same browser session but is otherwise self-sufficient.

To configure the links, set the `clickable` attribute to `true`, and configure the `ReturnColumn` element and its two child elements `Link` and `RequestParameters`, as follows:

- **Link element**: Configure the single attribute `href` for this element.

This attribute specifies the base URL of the destination page. An absolute URL begins with `http`, for example, `http://www.xyz.com`. The destination page for an absolute URL is outside the Administrative and User Console. A relative URL, for example, `searchResources.do`, leads to a destination page in the Administrative and User Console.

- `RequestParameters` element: Configure multiple `RequestParameter` elements for this element.

Each `RequestParameter` element represents one request parameter that must be submitted for the destination page to display properly. The `name` attribute specifies the name of the request parameter. If the request parameter is static, that is, it does not depend on any other value in the result set, you configure its value by using the `value` attribute. If the value of the request parameter is dynamic, that is, it depends on another column of the result set, for example, the Resource Key, then you specify the value by using the `column` attribute. The `column` attribute contains the column code for the column whose value is to be returned.

5.4.3 Creating an REP Entry and Providing Access to the Report

Once the report metadata is complete, update the REP and RPG tables to make the report available.

5.4.3.1 Creating a New Entry in the REP Table

The REP table contains a list of all the reports in the system. Defining a new report requires creating a new row in the REP table representing this report. Apart from the common columns, for example, Create Date, Created By, Row Version, and so on, you must specify the columns that are populated. The values in the parentheses are examples for User Resource Access reports.

- `REP_NAME`: This column contains the name of the report that is displayed to the user.
This name is unique in the REP table. (User Resource Access)
- `REP_CODE`: A unique code for the report. (UserResourceAccess)
- `REP_DESCRIPTION`: A description for the report that is displayed to the user. (Resource access rights for selected users).
- `REP_SP_NAME`: The name of the stored procedure that provides the data for the report. (XL_SP_UserResourceAccess)
- `REP_XML_META`: This column is a `clob` that contains the entire metadata for the report defined in the previous section.
- `REP_TYPE`: The type of the report. This can have two values: `Operational` or `Historical`. (Operational)
- `REP_DATASOURCE`: The name of the data source against which the report is run.
This can have two values: `Default` or `Reporting`. Usually, the `Operational` reports are run against the `Default` database while the `Historical` reports are run against the `reporting` database. (Default)
- `REP_MAX_REPORT_SIZE`: This represents the maximum number of records a report can return.

Different reports will return different amounts of data for each record. To ensure a fast response time for a report, a maximum number of records that a report can return is enforced for each report. (5000)

- **REP_FILTER_COUNT:** The number of filter drop downs on the Report Display page for this report (3).

The following is an example of the insert statement that populates the REP table with the User Resource Access report data for an Oracle Database:

```
INSERT INTO REP (REP_KEY, REP_CODE, REP_TYPE, REP_NAME, REP_DESCRIPTION,
                REP_DATASOURCE, REP_SP_NAME, REP_MAX_REP_SIZE, REP_FILTER_COUNT,
                REP_DATA_LEVEL, REP_CREATE, REP_CREATEBY, REP_UPDATE,
                REP_UPDATEBY, REP_ROWVER)
VALUES (rep_seq.nextval, 'UserResourceAccess', 'Operational', 'User Resource
Access','Resource access rights for selected users', 'Default',
'XL_SP_UserResourceAccess', 5000, 3, 1, SYSDATE, <System Administrator User Key>,
SYSDATE, <System Administrator User Key>, HEXTORAW('0000000000000000'));
```

5.4.3.2 Loading XML Metadata

After you create an entry in the REP table for the report, you must load the XML metadata for the report into the REP table's REP_XML_META column. See ["Creating the Report XML Metadata"](#) on page 5-11 for information about creating metadata.

5.4.3.3 Providing a User Group with Access to a Report

The following procedure describes how to provide access to a report.

See Also: *Oracle Identity Manager Administrative and User Console Guide*

To provide access to the new report to a particular user group:

1. Search for the user group by using the Manage User application.
2. Navigate to the detail page for the user group.
3. Click the Allowed Reports link in the additional details drop down.

The Reports page under Group Detail is displayed.

4. Click the **Assign Reports** button.

The Assign Reports page is displayed in the Reports section under Group Detail. The report you have just created is listed on this page.

5. To provide access to the report, assign it to at least one group.

5.4.4 Modifying Property Files for Translating Label Names, Report Names, and Report Descriptions

This section describes how to modify property files for translating label names, report names, and report descriptions. It contains the following topics:

- [Adding Properties for Translating Label Names](#)
- [Adding Properties for Translating Report Names and Descriptions](#)

5.4.4.1 Adding Properties for Translating Label Names

After you write the XML metadata, you must introduce the new field label properties in the metadata file in the properties files. You add the properties to the files corresponding to the locales that you want to support. The languages that are supported in Release 9.1.0.1 and their associated property files are as follows:

- Chinese (Simplified) (xlWebAdmin_zh_CN.properties)
- Chinese (Traditional) (xlWebAdmin_zh_TW.properties)
- Danish (xlWebAdmin_da.properties)
- English (xlWebAdmin_en_US.properties)
- French (xlWebAdmin_fr.properties)
- German (xlWebAdmin_de.properties)
- Italian (xlWebAdmin_it.properties)
- Japanese (xlWebAdmin_ja.properties)
- Korean (xlWebAdmin_ko.properties)
- Portuguese (Brazilian) (xlWebAdmin_pt_BR.properties)
- Spanish (xlWebAdmin_es.properties)

When Oracle Identity Manager encounters an unsupported locale, it refers to the `xlWebAdmin.properties` file. You add to these files the properties included as `fieldLabel` attributes of `InputParameter` tags and `label` attributes of `ReturnColumn` tags. Instead of providing the actual name of the input field labels or return column labels, you specify the property name, which is then looked up from the respective properties file. This makes it simpler for internationalization.

The following examples illustrate how to set the `fieldLabel` attributes of the `InputParameter` and `ReturnColumn` tags.

Example 1

```
<InputParameter name="struserlogin_in" parameterType="varchar2" order="1"
fieldType="TextField" fieldLabel="report.userResourceAccess.label.userLogin"
required="false" />
```

This `InputParameter` tag is from `UserResourceAccess.xml` metadata file for the User Resource Access report. The `fieldLabel` attribute of this tag has the value of `report.userResourceAccess.label.userLogin`. The corresponding entry for this property in the `xlWebAdmin_en_US.properties` file is:

```
report.userResourceAccess.label.userLogin=Userid
```

Example 2

```
<ReturnColumn name="Users.First Name"
label="report.userResourceAccess.label.firstName"
position="SectionHeader" filterColumn="true"
filterColumnName="usr usr_first_name" />
```

This `ReturnColumn` tag is from `UserResourceAccess.xml` metadata file for the User Resource Access report. The `label` attribute of this tag has the value of `report.userResourceAccess.label.firstName`. The corresponding entry for this property in the `xlWebAdmin_en_US.properties` file is:

```
report.userResourceAccess.label.firstName=First Name
```

Note: If the actual label names change (as a result of customization, for example), the property names need not be changed.

Perform the following steps to edit the `xlWebAdmin_en_US.properties` file:

1. Extract the `xlWebApp.war` file in the `OIM_HOME/webapp` directory to a temporary directory.
2. Open in a text editor the `xlWebAdmin_en_US.properties` file in the temporary directory where you extracted the `xlWebApp.war` file.
3. Add the appropriate properties as `fieldLabel` attributes of `InputParameter` tags and label attributes of `ReturnColumn` tags.
4. Re-create the `xlWebApp.war` file, and copy it to the `OIM_HOME/webapp` directory.

5.4.4.2 Adding Properties for Translating Report Names and Descriptions

After you create an entry in the `REP` table, you must add translation properties for the report name and description to the properties files. This procedure is necessary to localize the English entries in the database and must be performed even if you only want to provide support for the English locale.

To add properties for translating report names and descriptions:

1. Extract the `xlWebApp.war` file in the `OIM_HOME/webapp` directory to a temporary directory.
2. Open in a text editor the `xlDefaultAdmin.properties` file from the temporary directory where you extracted the `xlWebApp.war` file.
3. In the `xlDefaultAdmin.properties` file, locate the `global.resultSet.Reports.Report~Name` property. This property identifies the names of all reports in the system. Each report assigned to the property is separated by the vertical bar (`|`) character. Append a vertical bar, and add the name of the new report to the property value. Ensure that there are no spaces between the pipe character and the report name, and that there are no trailing spaces following the report name.
4. Locate in the `xlDefaultAdmin.properties` file the `global.resultSet.Reports.Report~Description` property, which contains descriptions of all reports in the system. Each report assigned to the property is separated by a vertical bar (`|`) character. Append a pipe character and the description of the new report to the property value. Ensure that there are no spaces between the pipe character and the description, and that there are no trailing spaces following the description.
5. Open in a text editor the properties file representing the locale for which you want to translate report names and descriptions.
6. Add a `global.resultSet.Reports.Report~Name.ReportName` property, which identifies the report name and its localized value. The *ReportName* portion of the property name represents the name of the report. Spaces in the property name are represented by tildes (`~`). You assign a localized value to the property. For example, you add the following property to the `xlWebAdmin_en_US.properties` file to represent a report named Group Information:


```
global.resultSet.Reports.Report~Name.Group~Information=Group Information
```
7. Add a `global.resultSet.Reports.Report~Description.Report~Description` property, which contains a report description along with and its localized value. The *Report~Description* portion of the property name represents the default report descriptions. Spaces in the report description are represented by tildes (`~`). You assign a localized value to the property. For example, you add the following

property to the `xlWebAdmin_en_US.properties` file for a description of the Group Information:

```
global.resultSet.Reports.Report~Description.Description~of~the~Group~Informatio
n~report=Description of the Group Information report
```

8. Add `global.ReportName.Lookup.Report-Name` and `global.ReportDesc.Lookup.Report-Description` lookup properties for each new report. The *Report-Name* and *Report-Description* portions of each property name represent the default report name and description, respectively. Spaces in the report description are represented by hyphens (-). You assign localized values to each property. For example, you add the following properties to the `xlWebAdmin_en_US.properties` file for the Group Information report:

```
global.ReportName.Lookup.Group-Information=Group Information
global.ReportDesc.Lookup.Description~of~the~Group~Information~report=Descriptio
n of the Group Information report
```

9. Recreate the `xlWebApp.war` file and copy it to the `OIM_HOME/webapp` directory.

5.5 Working with Third-Party Reporting Tools

Third-party reporting tools can run reports against Oracle Identity Manager by using the provided stored procedures. You do not need to understand the data model or write queries for predefined reports. You can use any reporting tool for custom stored procedures and custom reports.

Information about the snapshot and changes are stored in XML form in the UPA table. A third-party XML reporting tool can generate reports from this table. If XML is not a desired format, the reporting tool can use the reporting tables related to the user profile audit feature to retrieve data: `UPA_USR`, `UPA_FIELDS`, `UPA_GRP_MEMBERSHIP`, `UPA_RESOURCE`, `UPA_UD_FORMS`, and `UPA_UD_FORMFIELDS`.

5.6 Date Ranges in Historical Reports

The historical reports accept two different date ranges as input parameters, the Data Snapshot Date Range and the Data Changes Date Range.

The Data Snapshot Date Range is actually the existing date range for historical reports. It has been labeled this release onward. It filters records depending on whether the records are applicable in the specified date range.

The Data Change Date Range is a new feature of this release. It filters the records depending on whether the records were changed in the specified date range.

You need to specify only one date range. The Data Snapshot Date Range is a superset of the Data Change Date Range. All changes in data are automatically included in the Data Snapshot Date Range.

Note: If you specify values in both the date ranges, then the Data Change Date Range overrides the Data Snapshot Date Range and the latter is ignored.

5.7 Limitations

The following are the limitations of reporting for this release:

- If you use the filter or input parameters while retrieving reports, the "_" character behaves as a single character wildcard.
- If the "User Id reuse" system property is set to `true`, a user can be created with a same ID as that of a deleted user. In case you create a new user through this feature, all the reports in which the user data is displayed in a section would show the data for both deleted and active users in a single section. This is because both the users will share the same user ID.

Secondary Datasource Reporting

You can configure Oracle Identity Manager to use one database for current transactional data and a secondary database for historical data. The secondary database eases the load on the transactional database.

You can use different data sources for the secondary database. The following sections describe how to configure Oracle Identity Manager and your application server to use a secondary data source.

This chapter discusses the following topics:

- [Writing User Profile Audits to a Secondary Datasource](#)
- [Steps to Set Up a Secondary Data Source](#)
- [Using JBoss Application Server with a Secondary Data Source](#)
- [Using Oracle WebLogic Server with a Secondary Data Source](#)
- [Using IBM WebSphere Application Server with a Secondary Data Source](#)
- [Using Oracle Application Server with a Secondary Data Source](#)

6.1 Writing User Profile Audits to a Secondary Datasource

User profile audit data can increase in size quickly. Oracle recommends that you use a secondary database to store this information. The following system property enables reading and writing to this database directly:

`XL.UserProfileAuditInSecondaryDS.`

By default, the `XL.UserProfileAuditInSecondaryDS` property is set to `false`. If this property is set to `true`, the system reads and writes all user profile data directly to and from the secondary database.

If you configure a secondary database, all historical reports are automatically configured to run against it.

The user profile audit interacts directly with the secondary database. You must replicate other tables from the transactional database because the report needs them for access control and filtering of the report. You can disable these tables and constraints for ease of data backup, restore, or replication.

[Table 6–1](#) lists the tables and constraints.

Table 6–1 Tables and Constraints Used in Historical Reports

Table Name	Foreign Key Constraint Name	Referenced Table Name	Referenced Column Name
AAD	FK_AAD_FK_AAD_AC_ACT	ACT	ACT_KEY
	FK_AAD_FK_AAD_UG_UGP	UGP	UGP_KEY
ACT	FK_ACT_ACT	ACT	PARENT_KEY
	FK_ACT_SRP	SRP	SRP_KEY
GPG	FK_GPG_UGP	UGP	UGP_KEY
	FK_GPG_UGP_KEY_UGP	UGP	GPG_UGP_KEY
OUG	FK_OUG_OBJ	OBJ	OBJ_KEY
	FK_OUG_UGP	UGP	UGP_KEY
POL			
PTY			
REQ	FK_REQ_ORC	ORC	ORC_KEY
	FK_REQ_OST	OST	OST_KEY
	FK_REQ_USR	USR	USR_KEY
UGP			
USG	FK_USG_RUL	RUL	RUL_KEY
	FK_USG_UGP	UGP	UGP_KEY
	FK_USG_USR	USR	USR_KEY
USR	FK_USR_ACT	ACT	ACT_KEY
OSI	FK_OSI_ACT	ACT	ACT_KEY
	FK_OSI_ASSIGNED_TO_USR	USR	USR_KEY
	FK_OSI_MIL	MIL	MIL_KEY
	FK_OSI_ORC	ORC	ORC_KEY
	FK_OSI_ORD	ORD	ORD_KEY
	FK_OSI_PKG	PKG	PKG_KEY
	FK_OSI_REQ	REQ	REQ_KEY
	FK_OSI_RSC	RSC	RSC_KEY
	FK_OSI_SCH	SCH	SCH_KEY
	FK_OSI_SCH_OSI_RECOVER_Y_FOR	SCH	SCH_KEY
	FK_OSI_SCH_OSI_RETRY_FO R	SCH	SCH_KEY
	FK_OSI_TLG	TLG	TLG_KEY
	FK_OSI_TOS	TOS	TOS_KEY
	FK_OSI_TO_UGP	UGP	UGP_KEY
	FK_OSI_TO_USR	USR	USR_KEY
OST	FK_OST_OBJ	OBJ	OBJ_KEY
OBI	FK_OBI_OBJ	OBJ	OBJ_KEY

Table 6–1 (Cont.) Tables and Constraints Used in Historical Reports

Table Name	Foreign Key Constraint Name	Referenced Table Name	Referenced Column Name
	FK_OBI_ORC	ORC	ORC_KEY
	FK_OBI_QUE	QUE	QUE_KEY
	FK_OBI_REQ	REQ	REQ_KEY
	FK_OBI_USR	USR	USR_KEY
OBJ	FK_OBJ_SDK	SDK	SDK_KEY
OBA	FK_OBA_OBJ	OBJ	OBJ_KEY
	FK_OBA_UGP	UGP	UGP_KEY
RCE	FK_RCE_ACT	ACT	ACT_KEY
	FK_RCE_ADMIN_UGP	UGP	UGP_KEY
	FK_RCE_ADMIN_USR	USR	USR_KEY
	FK_RCE_OBJ	OBJ	OBJ_KEY
	FK_RCE_ORC	ORC	ORC_KEY
	FK_RCE_USR	USR	USR_KEY
MIL	FK_MIL_DEFAULT_ASSIGNED_USR	USR	USR_KEY
	FK_MIL_EVT	EVT	EVT_KEY
	FK_MIL_TOS	TOS	TOS_KEY
RCX			
ORC	FK_ORC_ACT	ACT	ACT_KEY
	FK_ORC_ORC_PARENT_KEY	ORC	ORC_KEY
	FK_ORC_ORD	ORD	ORD_KEY
	FK_ORC_PKG	PKG	PKG_KEY
	FK_ORC_PKH	PKH	PKH_KEY
	FK_ORC_REQ	REQ	REQ_KEY
	FK_ORC_TOS	TOS	TOS_KEY
	FK_ORC_USR	USR	USR_KEY
OIU	FK_OIU_LAST_ATTESTED_USR	USR	USR_KEY
	FK_OIU_OBI	OBI	OBI_KEY
	FK_OIU_ORC	ORC	ORC_KEY
	FK_OIU_OST	OST	OST_KEY
	FK_OIU_POL	POL	POL_KEY
	FK_OIU_REQ	REQ	REQ_KEY
	FK_OIU_USR	USR	USR_KEY
OSH	FK_OSH_BY_USR	USR	USR_KEY
	FK_OSH_SCH	SCH	SCH_KEY
	FK_OSH_STA	STA	STA_KEY

Table 6–1 (Cont.) Tables and Constraints Used in Historical Reports

Table Name	Foreign Key Constraint Name	Referenced Table Name	Referenced Column Name
	FK_OSH_TO_UGP	UGP	UGP_KEY
	FK_OSH_TO_USR	USR	USR_KEY
SCH			
RQU	FK_RQU_REQ	REQ	REQ_KEY
	FK_RQU_USR	USR	USR_KEY
RQO	FK_RQO_OBI	OBI	OBI_KEY
	FK_RQO_OBJ	OBJ	OBJ_KEY
	FK_RQO_POL	POL	POL_KEY
	FK_RQO_REQ	REQ	REQ_KEY
STA			

6.2 Steps to Set Up a Secondary Data Source

To set up a secondary database:

1. Create the secondary database.
You can back up and restore the transactional database under a different database name, or you can replicate the transactional database.
2. Set up the application server to use the secondary database.
See the following sections for details.
3. Set the system property `XL.UserProfileAuditInSecondaryDS` to `True`.
This stores user profile audit data in the secondary database.
4. Configure daily replication of the data so that the tables in the secondary database listed in [Table 6–1](#) are updated from the primary database. After the secondary database is functional, do not replicate the entire primary database in the secondary database or any audit data that has been stored in the secondary database will be deleted.
Or, set up either a full restore or replication.
5. Ensure that all stored procedures are replicated correctly in the secondary database.
6. Define a connection URL as follows:

- For Oracle Database:

```
jdbc:oracle:thin:@IP of database:SID
```

6.3 Using JBoss Application Server with a Secondary Data Source

To create a new data source on JBoss Application Server, a new file called `xlreportds-service.xml` is created by the setup in the deployment directory. This file creates an alias to the transactional database by using the `java:jdbc/xlXAReportingDS` setting.

To point to a secondary database on JBoss Application Server:

1. Open the `xell-ds.xml` file in an editor.
2. Add the following to `xell-ds.xml` as a second `xa-datasource` tag for the Oracle database:

```
<xa-datasource>
<jndi-name>jdbc/xlXAReportingDS</jndi-name>
<track-connection-by-tx>true</track-connection-by-tx>
<isSameRM-override-value>false</isSameRM-override-value>
<xa-datasource-class>oracle.jdbc.xa.client.OracleXADataSource </xa-datasource-
class>
<xa-datasource-property name="URL">jdbc:oracle:thin:@<IP of database system>:
1521:XELL </xa-datasource-property>
<xa-datasource-property name="User">sysadm</xa-datasource-property>
<xa-datasource-property name="Password">sysadm</xa-datasource-property>
<exception-sorter-class-name> org.jboss.resource.adapter.jdbc.vendor.
OracleExceptionSorter </exception-sorter-class-name>
<no-tx-separate-pools/>
<valid-connection-checker-class-name> org.jboss.resource.adapter.jdbc.vendor.
OracleValidConnectionChecker </valid-connection-checker-class-name>
</xa-datasource>
```

The class names for Oracle Database are as follows:

- Oracle:

```
oracle.jdbc.xa.client.OracleXADataSource
```

3. Change the database name, user name, and password to connect to the database you set up as the secondary database.
4. Delete the `xlreportds-service.xml` file.
5. Restart the JBoss server.

Note: Do not add the `xa-datasource` block in this section or point the `jdbc/xlXAReportingDS` to the transactional database because it causes errors. To point to the same transactional database, keep the `xlreportds-service.xml` file as is.

6.3.1 Cluster Configuration for JBoss Application Server

In a standalone setup, the `xell-ds.xml` and `xlreportds-service.xml` files are in the `JBOSS_HOME/server/default/deploy/` directory.

In a clustered setup, the `xell-ds.xml` file is in the `JBOSS_HOME/server/all/farm/` directory, and `xlreportds-service.xml` in the `JBOSS_HOME/server/all/deploy/` directory.

To configure a cluster for JBoss Application Server:

1. Copy the changes to the `xell-ds.xml` file to all computers in the cluster.
2. Restart the JBoss servers on all computers in the cluster.

6.4 Using Oracle WebLogic Server with a Secondary Data Source

Before changing the data source that Oracle Identity Manager uses for reporting, create a new data source in Oracle WebLogic Server. Perform the procedure given in Oracle WebLogic Server documentation to set up a new data source.

To configure Oracle WebLogic Server with a secondary data source by using the Oracle database:

1. Log in to the administrative console of the application server.
2. In the Change Center region, click **Lock & Edit**.
3. In the Domain Structure region, navigate to **Services, JDBC, and Data Sources**.
4. In the Data Sources region on the right pane, click **New**.
5. Set the following properties under Create a New JDBC Data Source:
 - **Name:** `xlXAReportingDS`
 - **JNDI name:** `jdbc/xlXAReportingDS`
 - **Database type:** `Oracle`
 - **Database Driver:** `*Oracle's Driver (Thin XA) Versions: 9.0.1, 9.2, 10, 11`
6. Click **Next**.
7. On the Transaction Options page, click **Next**.
8. Set the following properties under the Connection Pool tab:
 - **Database Name:** `SID`
 - **Host Name:** `@Database_IP_Address`
 - **Port:** `@port_no`
 - **Database User Name:** `secondary_database_user_name`
 - **Password:** `secondary_database_password`
 - **Confirm Password:** `secondary_database_password`
9. Click **Next**.
10. Ensure that the Test Database Connection page shows the following properties:
 - **Driver Class Name:** `oracle.jdbc.xa.client.OracleXADataSource`
 - **URL:** `jdbc:oracle:thin:@Database_IP_Address:port_no:SID`
 - **Database User Name:** `secondary_database_user_name`
 - **Password:** `secondary_database_password`
 - **Confirm Password:** `secondary_database_password`
 - **Properties:** `User=secondary_database_user_name`
11. Click **Test Configuration**.

The "Connection test succeeded" message is displayed.
12. Click **Next**.
13. On the Select Targets page, select **Admin Server** and then click **Finish**.
14. In the Change Center region, click **Active Changes**.
15. Edit the `weblogic.profile` file, and add the JNDI information that points to the new data source in the `OIM_HOME/xellerate/Profiles` directory.

Comment out the existing data source entry for `xlXADS` and add the information for `xlXAReportingDS` as follows:

```
# Reporting data source
# datasource.report=jdbc/xlXADS
datasource.report=jdbc/xlXAReportingDS
```

16. Ensure that the Design Console System property is set to True as follows:

```
XL.UserProfileAuditInSecondaryDS=True
```

17. Stop and restart Oracle Identity Manager.

18. Run the following command for the changes to take effect:

```
OIM_HOME/xellerate/setup/patch_weblogic.cmd/sh
WEBLOGIC_ADMIN_PASSWORD OIM_DATASOURCE_PASSWORD
```

Note: If the patch application process fails, then Oracle Weblogic might have some files locked. Perform the following steps to redeploy Oracle Identity Manager:

1. Back up the applications.
 2. Log in to the administrative console of the application server and remove the Xellerate and Nexaweb applications.
 3. Delete the WLXellerateFull.ear and WLNexaweb.ear files from the *OIM_HOME/xellerate/OIMApplications* directory.
 4. Run the `patch_Weblogic.cmd/sh patch`.
-

19. Stop and restart Oracle Identity Manager.

6.4.1 Cluster Configuration for Oracle WebLogic Server

To configure a cluster for Oracle WebLogic Server, you must deploy the secondary data source on all members of the cluster.

6.5 Using IBM WebSphere Application Server with a Secondary Data Source

Before changing the data source used by Oracle Identity Manager for reporting, you must create a new data source in IBM WebSphere Application Server. See the IBM WebSphere Application Server manuals for information about setting up a new data source.

To configure IBM WebSphere Application Server with a secondary data source by using the Oracle database:

1. Log in to the WebSphere administrator console.
2. Create a new data source with the following details:
 - **Name:** *XAReportingDataSource*
 - **JNDI name:** *jdbc/xlXAReportingDS*

3. Define the connection URL as follows:

```
jdbc:oracle:thin:@IP_of_database:port_number:SID
```

For example: `jdbc:oracle:thin:@192.168.161.134:1521:xeltest`

4. Use the following J2C authentication data values:

- **Alias:** *secondary_user_alias*
 - **User:** *secondary_user*
 - **Password:** *secondary_user_password*
 - **Description:** *Descriptive_text_for_the_data*
5. Select the component-managed authentication aliases for XAReportingDataSource with the following values:
 - **Component-managed authentication alias:** *J2C_Authentication_Data Entries*
 - **Container-managed authentication alias:** *J2C_Authentication_Data Entries*
 6. Save and synchronize changes among all nodes.
 7. Open the `websphere.profile` file, and add the JNDI information that points to the new data source in the `OIM_HOME/xellerate/Profiles/` directory.

Comment out the existing data source entry for `xlXADS` and add the information for `xlXAReportingDS` as follows:


```
# Reporting data source
#datasource.report=jdbc/xlXADS
datasource.report=jdbc/xlXAReportingDS
```
 8. Set the following Java Client System property to true:

`XL.UserProfileAuditInSecondaryDS=True`
 9. Run `patch_websphere.cmd` or `patch_websphere.sh` as applicable from the `OIM_HOME/xellerate/setup` directory.

6.5.1 Cluster Configuration for IBM WebSphere Application Server

To configure a cluster for IBM WebSphere Application Server:

1. Modify each `websphere.profile` file on all nodes participating in the cluster.
2. Run the `patch_websphere.cmd` or `patch_websphere.sh` as applicable from the `OIM_HOME/xellerate/setup` directory from the network deployment manager (NDM) node.
3. Stop and restart all nodes and servers.

6.6 Using Oracle Application Server with a Secondary Data Source

Before changing the data source used by Oracle Identity Manager for reporting, you must create a new data source in Oracle Application Server.

To configure Oracle Application Server with a secondary data source by using the Oracle database:

1. Log in to the Oracle Enterprise Manager 10g Application Server Console for the Oracle Application Server instance.
2. Click the Oracle Application Server instance on which Oracle Identity Manager is deployed.

The Administration tab displays the list of administration tasks you can perform on the selected Oracle Application Server instance.

3. Expand the Services section of the table by either clicking the expand icon or clicking **Expand All**.
4. In the **JDBC Resources** row, click the task icon.
5. To create a connection pool, in the Connection Pools section, click **Create** and then select the following values for the **Application** and **Connection Pool Type** parameters:
 - **Application:** default
 - **Connection Pool Type:** New Connection Pool
6. Click **Continue**.
7. For the following parameters, enter (or change to) the specified values:
 - **Name:** xlXAReportingConnectionPool
 - **Connection Factory Class:** oracle.jdbc.xa.client.OracleXADataSource
 - **JDBC URL:** jdbc:oracle:thin:@IP_of_database:port_number:SID
For example: jdbc:oracle:thin:@127.0.0.1:1521:xeltest
 - **Username:** Enter the secondary database user name.
 - **Use Cleartext Password:** Enter the secondary database password.

See Also: Oracle Application Server documentation for information about using the Indirect Password feature

Alternatively, to view information about this feature, click the icon displayed next to the Use Indirect Password field in the Credentials section of the same tab.
8. On the Attributes tab, enter the values specified for xlXAConnectionPool. To view those values:
 - a. Log in to another instance of the Oracle Enterprise Manager 10g Application Server Console for the Oracle Application Server instance.
 - b. Click the Oracle Application Server instance on which Oracle Identity Manager is deployed.
 - c. On the **Administration** tab, expand the **Services** section of the table.
 - d. In the JDBC Resources row, click the task icon.
 - e. Click **xlXAConnectionPool**.

The values that you must copy are displayed on the Attributes tab.
9. Click **Finish**.
10. In the Data Sources section, click **Create** to create a datasource with the following parameters:
 - **Application:** default
 - **Datasource Type:** Managed Data Source
11. Click **Continue**.
12. For the following parameters, enter (or change to) the specified values:
 - **Name:** xlXAReportingDS
 - **JNDI Location:** jdbc/xlXAReportingDS

- **Transaction Level:** Global & Local transaction
- **Connection Pool:** xlXAReportingConnectionPool

13. Click Finish.

- 14.** Open the `oc4j.profile` file, and add the JNDI information that points to the new data source in the `OIM_HOME/Profiles/` directory. Comment out the existing data source entry for `xlXADS` and add the information for `xlXAReportingDS` as follows:

```
# Reporting data source
#datasource.report=jdbc/xlXADS
datasource.report=jdbc/xlXAReportingDS
```

- 15.** Based on the operating system on which Oracle Identity Manager is installed, run one of the following scripts:

For UNIX:

```
patch_oc4j.sh oc4j_admin_password oim_db_user_password
```

For Microsoft Windows:

```
patch_oc4j.cmd oc4j_admin_password oim_db_user_password
```

- 16.** Restart Oracle Application Server.

6.6.1 Cluster Configuration for Oracle Application Server

To configure a cluster for Oracle Application Server:

1. Perform steps 2 through 13 of the procedure to configure Oracle Application Server with a secondary data source in all the Oracle Application Server instances on which Oracle Identity Manager is installed.
2. Modify each `OIM_HOME/profiles/oc4j.profile` file on all cluster members participating in the cluster.
3. Based on the operating system on which Oracle Identity Manager is installed, run `patch_oc4j.sh` or `patch_oc4j.cmd` on all cluster members participating in the cluster.
4. Stop and restart all cluster members.

Sample Code for a Custom Post-Processor

The sample post-processor in this section gets the group entitlements from the Active Directory integration. The Active Directory integration uses a child table to store the group membership.

Create a table to store the information you need in the reporting database by using the following SQL scripts.

```
CREATE SEQUENCE UPA_UD_ADUSRC_SEQ
INCREMENT BY 1
START WITH 1
CACHE 20

/*=====*/
/* Table: UPA_UD_ADUSRC
*/
/*=====*/
CREATE TABLE UPA_UD_ADUSRC (
    UPA_UD_ADUSRC_KEYNUMBER(19)      NOT NULL,
    UPA_RESOURCE_KEYNUMBER(19)       NOT NULL,
    OIU_KEYNUMBER(19)                NOT NULL,
    UD_ADUSRC_GROUPNAMEVARCHAR2(256) NOT NULL,
    STATUSVARCHAR2(7),
    UPA_UD_ADUSRC_EFF_FROM_DATE TIMESTAMP      NOT NULL,
    UPA_UD_ADUSRC_EFF_TO_DATETIMESTAMP,
    CREATE_DATETIMESTAMP              NOT NULL,
    UPDATE_DATETIMESTAMP              NOT NULL,
    CONSTRAINT PK_UPA_UD_ADUSRC PRIMARY KEY (UPA_UD_ADUSRC_KEY)
)

COMMENT ON TABLE UPA_UD_ADUSRC IS
'Stores AD group entitlements'

CREATE INDEX IDX_UPA_UD_ADUSRC_EFF_FROM_DT ON UPA_UD_ADUSRC (
    UPA_UD_ADUSRC_EFF_FROM_DATE ASC
)
```

Use the following code for the custom post-processor:

```
package sample.audit.processor;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
```

```

import java.sql.Statement;
import java.sql.Timestamp;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.sql.Types;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;

import com.thortech.xl.audit.auditdataprocessors.CustomAuditDataProcessor;
import com.thortech.xl.audit.engine.AuditData;
import com.thortech.xl.audit.exceptions.AuditDataProcessingFailedException;
import com.thortech.xl.util.logging.LoggerMessages;

public class ADUserGroupMembershipProcessor extends CustomAuditDataProcessor {

    private static final String CHANGE_TAG = "Change";
    private static final String ATTRIBUTE_TAG = "Attribute";
    private static final String NAME_ATTRIBUTE = "name";
    private static final String CHANGE_LOCATION_ATTRIBUTE = "where";
    private static final String ACTION_ATTRIBUTE = "action";

    private static final String CHILD_DATA_PREFIX = "/Data";
    private static final String RESOURCE_DATA_PREFIX =
"/ProcessData/Children/Child";
    private static final String RESOURCE_PROFILE_PREFIX =
"/UserProfileSnapshot/ResourceProfile/ResourceInstance";

    private static final String AD_RESOURCE_NAME = "AD User";

    /*private static final String[] UPA_UD_ADUSRC_COLUMNS =
    {"UPA_UD_ADUSRC_KEY", "OIU_KEY", "UD_ADUSRC_KEY", "UD_ADUSRC_GROUPNAME",
    "UPA_UD_ADUSRC_EFF_FROM_DATE", "UPA_UD_ADUSRC_EFF_TO_DATE",
    "CREATE_DATE", "UPDATE_DATE"};*/

    public void processAuditData(Connection operationalDB,
        Connection reportingDB, List auditDataList, Timestamp auditEpoch)
        throws AuditDataProcessingFailedException {
        for (Iterator iter = auditDataList.iterator(); iter.hasNext();) {
            AuditData auditData = (AuditData) iter.next();
            // Retrieve data from AuditData value object
            //String auditeeID = auditData.getAuditeeID();
            List changeElements = getChangeElements(auditData.getChanges());
            // Retrieve AD User Group Membership related changes
            List ADUserGrpMembershipChangeElements =
                getADUserGroupMembershipChangeElements(changeElements);
            // Process change elements
            for (Iterator iterator =
ADUserGrpMembershipChangeElements.iterator(); iterator.hasNext();) {
                Element changeElement = (Element) iterator.next();
                // Retrieve the resource instance key (OIU_KEY) from the XPath
expression
                long resourceInstanceKey = getResourceInstanceKey(changeElement);
                // Get the object name for this resource instance key
                String resName = getResourceName(auditData.getUpdatedSnapshot(),
resourceInstanceKey);

```

```

        if(resName == null || !resName.equals(AD_RESOURCE_NAME))
            continue;// this is not the AD User resource so, skip it and check
the next one...
        // Retrieve the child table key (UD_ADUSRC_KEY)
        long UDADUSRCKey = getUDADUSRCKey(changeElement);
        // Retrieve the current record, if present
        HashMap ADUserGroupMembershipProfile =
            readADUserGroupMembershipData(reportingDB,
resourceInstanceKey, UDADUSRCKey);
        // Reset the default columns
        ADUserGroupMembershipProfile.put("UPA_UD_ADUSRC_KEY",null);
        ADUserGroupMembershipProfile.put("OIU_KEY",null);
        ADUserGroupMembershipProfile.put("UD_ADUSRC_KEY",null);
        // Apply the changes
        String action = changeElement.getAttribute(ACTION_ATTRIBUTE);
        if (action.equalsIgnoreCase("Delete")) {
            ADUserGroupMembershipProfile.put("STATUS","DELETE");
        } else {

ADUserGroupMembershipProfile.put("STATUS",action.toUpperCase());
            Element groupNameElement =

getFirstChildElementByName(changeElement,ATTRIBUTE_TAG,NAME_ATTRIBUTE,"UD_ADUSRC_G
ROUPNAME");

            Attribute attrDetails = getAttributeDetails(groupNameElement);

ADUserGroupMembershipProfile.put("UD_ADUSRC_GROUPNAME",attrDetails.getNewValue());
        }
        // Set values for the default columns
        ADUserGroupMembershipProfile.put("OIU_KEY",new
Long(resourceInstanceKey));
        ADUserGroupMembershipProfile.put("UD_ADUSRC_KEY",new
Long(UDADUSRCKey));
        ADUserGroupMembershipProfile.put("UPA_UD_ADUSRC_EFF_FROM_DATE",
auditEpoch);
        ADUserGroupMembershipProfile.put("UPA_UD_ADUSRC_EFF_TO_DATE",
null);
        ADUserGroupMembershipProfile.put("CREATE_DATE", new
Timestamp(System.currentTimeMillis()));
        ADUserGroupMembershipProfile.put("UPDATE_DATE", new
Timestamp(System.currentTimeMillis()));
        // Update existing active record if present
        updateActiveADUserGroupMembershipProfile(reportingDB,
resourceInstanceKey, UDADUSRCKey, auditEpoch);
        // Insert new record
        insertNewADUserGroupMembershipProfile(reportingDB,
ADUserGroupMembershipProfile);
    }
}

private long insertNewADUserGroupMembershipProfile(Connection reportingDB,
HashMap ADUserGroupMembershipProfile)
throws AuditDataProcessingFailedException {
    long key = 0;
    try {
        String insertSQL =

generateNewADUserGroupMembershipInsertSQL(reportingDB,ADUserGroupMembershipProfile
);

```

```

        key = executeInsert(reportingDB, insertSQL,
ADUserGroupMembershipProfile);
    } catch (SQLException e) {
        String errMsg = "Unable to insert new AD User Group Membership
Profile";
        throw new AuditDataProcessingFailedException(errMsg,e);
    }
    return key;
}

private String generateNewADUserGroupMembershipInsertSQL(Connection
reportingDB,
    HashMap ADUserGroupMembershipProfile) throws SQLException {
    String valuesPlaceholder = "";
    String columnNames = "";
    String dbType = reportingDB.getMetaData().getDatabaseProductName();

    if (dbType.startsWith("Oracle")) {
        valuesPlaceholder = "?, ";
        columnNames = "UPA_UD_ADUSRC_KEY, ";
    }

    for (Iterator iter = ADUserGroupMembershipProfile.keySet().iterator();
iter.hasNext();) {
        String columnName = (String) iter.next();
        Object columnValue = ADUserGroupMembershipProfile.get(columnName);
        if (!columnName.equals("UPA_UD_ADUSRC_KEY") && columnValue != null) {
            valuesPlaceholder += "?, ";
            columnNames += columnName + ", ";
        }
    }

    // Trim the place holder variable and column names variable
    valuesPlaceholder =
(valuesPlaceholder.trim()).substring(0,valuesPlaceholder.length()-2);
    columnNames = (columnNames.trim()).substring(0,columnNames.length()-2);

    String insertSQL = "INSERT INTO UPA_UD_ADUSRC (" + columnNames + ") " +
        "VALUES (" + valuesPlaceholder + ")";

    return insertSQL;
}

private void updateActiveADUserGroupMembershipProfile(Connection reportingDB,
    long resourceInstanceKey, long UDADUSRCKey, Timestamp auditEpoch)
    throws AuditDataProcessingFailedException {
    String updateSQL = "UPDATE UPA_UD_ADUSRC " +
        "SET UPA_UD_ADUSRC_EFF_TO_DATE=? " +
        "WHERE OIU_KEY=? " +
        "AND UD_ADUSRC_KEY=? " +
        "AND UPA_UD_ADUSRC_EFF_TO_DATE is null";

    try {
        PreparedStatement pstmt = reportingDB.prepareStatement(updateSQL);
        pstmt.setTimestamp(1,auditEpoch);
        pstmt.setLong(2,resourceInstanceKey);
        pstmt.setLong(3,UDADUSRCKey);
        pstmt.executeUpdate();
    } catch (SQLException e) {
        String errMsg = "Failed to update active AD user group membership

```

```

        profile. " +
            "Data in UPA_UD_ADUSRC could be inconsistent";
        if (dbLogger.isDebugEnabled())
            dbLogger.debug(errMsg,e);
        throw new AuditDataProcessingFailedException(errMsg, e);
    }

}

/**
 *
 * @param operationalDB
 * @param resourceInstanceKey
 * @param UDADUSRCKey
 * @return
 * @throws AuditDataProcessingFailedException
 */
private HashMap readADUserGroupMembershipData(Connection reportingDB,
        long resourceInstanceKey, long UDADUSRCKey)
        throws AuditDataProcessingFailedException {
    String query = "SELECT * " +
        "FROM UPA_UD_ADUSRC " +
        "WHERE OIU_KEY=" + resourceInstanceKey + " " +
        "AND UD_ADUSRC_KEY=" + UDADUSRCKey + " " +
        "AND UPA_UD_ADUSRC_EFF_TO_DATE is null";

    try {
        return transformADUserGroupMembershipProfile(executeQuery(reportingDB,
query));
    } catch (SQLException e) {
        throw new AuditDataProcessingFailedException("Unable to read data from
" +
            "JDBC resultset", e);
    } catch (Exception e) {
        String errMsg = "AD User Group Membership information stored in " +
            "UPA_UD_ADUSRC is inconsistent";
        if (dbLogger.isDebugEnabled())
            dbLogger.debug(errMsg, e);
        throw new AuditDataProcessingFailedException(errMsg, e);
    }
}

/**
 *
 * @param result
 * @return
 * @throws Exception
 */
private HashMap transformADUserGroupMembershipProfile(ResultSet result)
        throws Exception {
    HashMap ADUserGroupMembershipProfile = new HashMap();
    ResultSetMetaData metadata = result.getMetaData();

    // Move the cursor to the beginning of the resultset
    if (result.next()) {
        //
        for (int i = 0; i < metadata.getColumnCount(); i++) {
            Object columnValue = null;
            String columnName = metadata洗ColumnName(i+1);
            int columnType = metadata.getColumnType(i+1);
            switch (columnType) {

```

```

        case Types.INTEGER:
            columnValue = new Long(result.getLong(i+1));
            break;
        case Types.VARCHAR:
        case Types.CHAR:
        case Types.LONGVARCHAR:
            columnValue = result.getString(i+1);
            break;
        case Types.TIMESTAMP:
            columnValue = result.getTimestamp(i+1);
            break;
        default:
            columnValue = null;
    }
    if (result.isNull()) {
        columnValue = null;
    }
    ADUserGroupMembershipProfile.put(columnName, columnValue);
}
// Check if more than one record was returned. If so throw an
exception
if (result.next()) {
    String errMsg = "More than one active record found for AD group" +
        result.getString("UD_ADUSRC_GROUPNAME");
    throw new Exception(errMsg);
}
}

return ADUserGroupMembershipProfile;
}

/**
 *
 * @param changeElement
 * @return
 */
private long getResourceInstanceKey(Element changeElement) {
    long resourceInstanceKey = 0;
    String changeLocation =
changeElement.getAttribute(CHANGE_LOCATION_ATTRIBUTE);
    int keyStartPosition =
changeLocation.indexOf(RESOURCE_PROFILE_PREFIX)+RESOURCE_PROFILE_PREFIX.length()+7
;
    int keyEndPosition =
keyStartPosition+changeLocation.substring(keyStartPosition).indexOf("'")-1;
    resourceInstanceKey =
Long.parseLong(changeLocation.substring(keyStartPosition,keyEndPosition+1));
    return resourceInstanceKey;
}

/**
 *
 * @param changeElement
 * @return
 */
private String getResourceName(Document snapshot, long resourceInstanceKey) {

    Element parentElement = snapshot.getDocumentElement();
    NodeList childNodes = parentElement.getChildNodes();

```

```

        for (int i = 0; i < childNodes.getLength(); i++) {
            Node childNode = childNodes.item(i);

            if ((childNode.getNodeType() == Node.ELEMENT_NODE) &&
                childNode.getNodeName().equals("ResourceProfile")) {
                NodeList resourceProfileNodeList = childNode.getChildNodes();
                for (int j = 0; j < resourceProfileNodeList.getLength(); j++) {
                    Node resNode = childNodes.item(j);
                    if ((resNode.getNodeType() == Node.ELEMENT_NODE) &&
                        resNode.getNodeName().equals("ResourceInstance")) {
                        Element resourceInstanceElement = (Element)resNode;
                        String key = resourceInstanceElement.getAttribute("key");
                        if(key != null && Long.parseLong(key) == resourceInstanceKey)
                        {
                            Element name =
getFirstChildElementByName(resourceInstanceElement, ATTRIBUTE_TAG, NAME_ATTRIBUTE,
"Objects.Name");
                            return name.getFirstChild().getNodeValue();
                        }
                    }
                }
            }
        }
        return null;
    }

    /**
     *
     * @param changeElement
     * @return
     */
    private long getUDADUSRCKey(Element changeElement) {
        long UDADUSRCKey = 0;
        String changeXPath =
changeElement.getAttribute(CHANGE_LOCATION_ATTRIBUTE);
        String processDataXPath =
changeXPath.substring(changeXPath.indexOf(RESOURCE_DATA_PREFIX));
        String childDataXPath =
processDataXPath.substring(processDataXPath.indexOf(CHILD_DATA_PREFIX));
        int keyStartPosition = CHILD_DATA_PREFIX.length()+7;
        int keyEndPosition =
keyStartPosition+childDataXPath.substring(keyStartPosition).indexOf("'")-1;
        UDADUSRCKey = Long.parseLong(childDataXPath.substring(keyStartPosition,
keyEndPosition+1));
        return UDADUSRCKey;
    }

    /**
     *
     * @param changeElements
     * @return
     */
    private List getADUserGroupMembershipChangeElements(List changeElements) {
        List ADUserGrpMembershipChangeElements = new ArrayList();

        for (Iterator iter = changeElements.iterator(); iter.hasNext();) {
            Element change = (Element) iter.next();
            if (isChildrenData(change.getAttribute(CHANGE_LOCATION_ATTRIBUTE)))
                ADUserGrpMembershipChangeElements.add(change);
        }
    }

```

```

    }

    return ADUserGrpMembershipChangeElements;
}

/**
 *
 * @param attribute
 * @return
 */
private boolean isChildrenData(String changeLocation) {
    if (changeLocation.startsWith(RESOURCE_PROFILE_PREFIX) &&
        changeLocation.indexOf(RESOURCE_DATA_PREFIX,
RESOURCE_PROFILE_PREFIX.length()) > 0)
        return true;
    else
        return false;
}

/**
 *
 * @param connection
 * @param query
 * @return
 */
protected ResultSet executeQuery(Connection connection, String query) {
    ResultSet result = null;
    try {
        Statement stmt = connection.createStatement();
        if (stmt.execute(query)) {
            result = stmt.getResultSet();
        }
    } catch (SQLException e) {
        if (dbLogger.isDebugEnabled()) {
            dbLogger.debug(LoggerMessages.getMessage("DBQueryExecutionError",
query), e);
        }
    }
    return result;
}

/**
 *
 * @param connection
 * @param userGroupMembershipProfile
 * @param updateSQL
 */
protected long executeInsert(Connection connection, String insertSQL,
    HashMap ADUserGroupMembershipProfile) throws SQLException {
    PreparedStatement insertStmt = connection.prepareStatement(insertSQL);
    String dbType = connection.getMetaData().getDatabaseProductName();

    long newKey = 0;
    int columnIndex = 1;
    //
    // Get new key for Oracle and set it into the prepared stmt
    if (dbType.startsWith("Oracle")) {
        ResultSet nextValRS = executeQuery(connection, "select
UPA_UD_ADUSRC_SEQ.nextval from dual");
        long nextVal = nextValRS.getLong(1);
    }
}

```

```

        insertStmt.setLong(columnIndex++,nextVal);
        newKey = nextVal;
    }

    // Set column names and values for other columns in UPA_UD_ADUSRC
    for (Iterator iter = ADUserGroupMembershipProfile.keySet().iterator();
iter.hasNext();) {
        String columnName = (String) iter.next();
        Object columnValue = ADUserGroupMembershipProfile.get(columnName);
        if (!columnName.equals("UPA_UD_ADUSRC_KEY") && columnValue != null) {
            if (columnValue.getClass().getName().endsWith("Long")) {

insertStmt.setLong(columnIndex++, ((Long)columnValue).longValue());
            } else if (columnValue.getClass().getName().endsWith("String")) {
                insertStmt.setString(columnIndex++, (String)columnValue);
            } else if (columnValue.getClass().getName().endsWith("Timestamp"))
{
                insertStmt.setTimestamp(columnIndex++, (Timestamp)columnValue);
            }
        }
    }

    insertStmt.executeUpdate();

    if (dbType.startsWith("Microsoft SQL Server")) {
        ResultSet nextValRS = executeQuery(connection, "select @@identity");
        long nextVal = nextValRS.getLong(1);
        newKey = nextVal;
    }
    return newKey;
}
}

```

Index

A

- API layer, 5-3
- AUD, 2-3
- aud_jms, 2-3
- Audit Engine, 1-2
- audit engine, 2-1
- audit levels, 2-1
- Audit Transaction, 1-2
- auditing, user profile, 3-1

C

- cluster configuration
 - JBoss, 6-5
 - WebSphere, 6-8
- connection URLs, 6-4
- CSV files, 5-2
- custom post-processors, 2-2
 - creating, 2-3
- CustomAuditDataProcessor, 2-3

D

- data collection, 3-1, 4-1
 - archiving, 3-1, 4-1
 - capturing, 3-1, 4-1
- data layer, 5-2
- data storage, 5-2
 - API Layer, 5-3
 - data layer, 5-2
 - XML metadata, 5-3
- Do Count, 5-9

F

- Filter Column Names, 5-9
- Filter Column Values, 5-9
- filters, 5-1

G

- GenerateSnapshot script, 2-1
- GenerateSnapshot.bat, 2-1
- GenerateSnapshot.sh, 2-1
- generic parameters, 5-8
 - Do Count, 5-9

- Filter Column Names, 5-9
- Filter Column Values, 5-9
- Page Size, 5-8
- Report Result Set, 5-8
- Sort Columns, 5-8
- Sort Order, 5-8
- Start Row, 5-8
- Total Rows, 5-9
- User Key, 5-8
- User-Defined Column Names, 5-9
- User-Defined Column Values, 5-9
- group profile snapshot changes XML, 4-3
- group profile snapshot XML, 4-2
- GroupAdmin, 4-2
- GroupInfo, 4-2
- GroupMembership, 3-2
- GroupSnapshot, 4-2

I

- InputParameter, 5-11
 - attributes, 5-11
 - examples, 5-12
- InputParameters, 5-11
- Issue Audit Messages Task, 2-3

J

- Java Client System property, 6-8
- JBoss, 6-4
 - cluster configuration, 6-5
 - database class names, 6-5
 - jdbc/xlXAReportingDS, 6-4
 - secondary data source, 6-4
 - standalone setup, 6-5
 - xa-datasource, 6-5
 - xlreportds-service.xml, 6-4, 6-5
- jdbc/xlXAReportingDS, 6-4, 6-7

N

- network deployment manager, 6-8
- new report creation, 5-8
 - stored procedures, 5-8

O

ObjectData, 3-3
Oracle Application Server
 secondary data source, 6-8
Oracle Identity Manager
 reporting, 5-1
Oracle Identity Manager Auditing, 1-1
 design components, 1-1

P

Page Size, 5-8
PolicyProfile, 3-2
post-processors, 3-8
 custom, 2-2
 sample code, A-1
 using, types, 2-2
 XML metadata, 2-2
processAuditData, 2-3
ProcessData, 3-3
profile auditing, 1-2

R

REP entries, 5-15
 Oracle Database, 5-16
 report, 5-16
 report access, 5-16
 updating tables, 5-15
 User Resource Access report, 5-15
REP table, 5-15
REP_CODE, 5-15
REP_DATASOURCE, 5-15
REP_DESCRIPTION, 5-15
REP_FILTER_COUNT, 5-16
REP_MAX_REPORT_SIZE, 5-15
REP_NAME, 5-15
REP_SP_NAME, 5-15
REP_TYPE, 5-15
REP_XML_META, 5-15
Report Result Set, 5-8
reporting, 5-1
 access, 5-15
 creation, 5-8
 data storage, 5-2
 engine, 5-1
 features, 5-1
 lookup by code, 5-12
 lookup by column, 5-12
 lookup by method, 5-12
 secondary data sources, 6-1
 third-party tools, 5-19
 User Resource Access Report, creating, 5-15
 XML metadata, 5-11
reports
 customized, 1-2
 secondary data source, 1-2
 standard, 1-2
ResourceInstance, 3-3
ResourceProfile, 3-2

ReturnColumns, 5-11, 5-14
 attributes, 5-14
 child tags, 5-14
 Link, 5-14
 redirecting links, 5-14
 RequestParameter, 5-15
 RequestParameters, 5-14

S

scheduled task, Issue Audit Messages Task, 2-3
secondary data sources, 6-1
 connection URL, 6-4
 JBoss, with, 6-4
 Oracle Application Server, with, 6-8
 setting up, 6-4
 User Profile Audit tables, 6-1
 user profile audit, writing, 6-1
 WebLogic, with, 6-5
 WebSphere, with, 6-7
 XL.UserProfileAuditInSecondaryDS, 6-1, 6-4
secondary databases, 5-2
snapshot
 storing, 3-7, 4-6
snapshot changes XML, 3-5, 4-3
Sort Order, 5-8
specific parameters, 5-9
Start Row, 5-8
stored procedures, 5-8
 generic parameters, 5-8
 notes, 5-10
 signature, example, 5-10
 specific parameters, 5-9
StoredProcedure, 5-11
Subgroups, 4-2

T

Total Rows, 5-9

U

UPA, 3-8, 4-7
UPA Form Data Upgrade Utility
 running, 5-7
 scripts, configuring, 5-6
 stored procedure, compiling, 5-7
UPA_FIELDS, 3-7, 3-8, 5-19
UPA_GRP_MEMBERSHIP, 3-7, 3-8, 5-19
UPA_RESOURCE, 3-7, 3-8, 5-19
UPA_UD_FORMFIELDS, 3-7
UPA_UD_FORMS, 3-7, 5-19
UPA_USR, 3-7, 3-8, 5-19
User Key, 5-8
user profile audit tables, 2-3, 3-8, 4-7
 AUD, 2-3
 aud_jms, 2-3
 GPA, 4-7
 UPA, 3-8, 4-7
 UPA_FIELDS, 3-8
 UPA_GRP_MEMBERSHIP, 3-8

- UPA_RESOURCE, 3-8
- UPA_USR, 3-8
- user profile auditing, 2-1, 3-1
 - audit engine, 2-1
 - data collection, 3-1, 4-1
 - GroupSnapshot, 4-2
 - Issue Audit Messages Task, 2-3
 - post-processors, 3-8
 - UserProfileSnapshot, 3-2
 - XL.UserProfileAuditDataCollection, 3-1
- user profile audits
 - tables used, 2-3, 3-8, 4-7
- user profile snapshot
 - trigger, 3-8, 4-7
- user profile snapshot changes XML, 3-5
- user profile snapshot XML, 3-2
- User Resource Access report
 - Oracle Database, 5-10
 - signature, 5-10
 - stored procedure, 5-10
- User Resource Access reports
 - REP entries, 5-15
- User-Defined Column Names, 5-9
- User-Defined Column Values, 5-9
- UserInfo, 3-2
- UserProfileAuditor, 2-3
- UserProfileSnapshot, 3-2

W

- WebLogic
 - Oracle Database, configuring, 6-6
 - secondary data source, 6-5
- WebSphere
 - cluster configuration, 6-8
 - connection URL, 6-7
 - J2C authentication data values, 6-7
 - jdbc/xlXAReportingDS, 6-7
 - patch_websphere.cmd, 6-8
 - patch_websphere.sh, 6-8
 - secondary data source, 6-7
 - websphere.profile, 6-8
 - XAReportingDataSource, 6-7
 - xlXAReportingDS, 6-8
- websphere.profile, 6-8

X

- xell-ds.xml, 6-5
- xlreportds-service.xml, 6-4, 6-5
- XL.UserProfileAuditDataCollection, 3-1
- XL.UserProfileAuditInSecondaryDS, 6-1, 6-4, 6-8
- xlWebAdmin.properties, 5-14
 - xlWebApp.war, 5-18
- xlXAReportingDS, 6-8
- XML metadata, 5-3
 - creating, 5-11
 - ReturnColumns tag, 5-14
 - StoredProcedure tag, 5-11

Y

- YPA_UD_FORMFIELDS, 5-19

