Oracle® Role Manager

Administrator's Guide

Release 10*g* (10.1.4.2)

E14610-01

June 2009

Oracle Role Manager Administrator's Guide Release 10*g* (10.1.4.2)

E14610-01

Primary Author: Carla Fabrizio

Contributing Author: Prakash Hulikere

Contributors: Vijay Appadorai, Miles Chaston, Ashish Chugh, April Escamilla, Bennett Falk, Stephen Grenholm, Seth Klein, Madhup Kumar, Devender Sharma

.

Contents

4 Creating and Maintaining System Identities

5 Configuring Oracle Role Manager for Single Sign-On

Index

List of Tables

Preface

Oracle Role Manager Administrator's Guide describes the administrative tools provided for Oracle Role Manager and how to use them. It provides context, examples, and specific instructions for Oracle Role Manager system administration.

Audience

This document is intended for those who are involved in the administration of Oracle Role Manager, and Oracle database administrators (DBAs) and system administrators.

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.

Deaf/Hard of Hearing Access to Oracle Support Services

To reach Oracle Support Services, use a telecommunications relay service (TRS) to call Oracle Support at 1.800.223.1711. An Oracle Support Services engineer will handle technical issues and provide customer support according to the Oracle service request process. Information about TRS is available at http://www.fcc.gov/cgb/consumerfacts/trs.html, and a list of phone numbers is available at http://www.fcc.gov/cgb/dro/trsphonebk.html.

Related Documents

For more information, refer to the following documents:

- *Oracle Role Manager Release Notes*
- *Oracle Role Manager User's Guide*
- *Oracle Role Manager Developer's Guide*
- *Oracle Role Manager Java API Reference*
- *Oracle Role Manager Integration Guide*
- *Oracle Role Manager Install Guide*

Conventions

The following text conventions are used in this document:

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

1 Introduction

This chapter introduces the Oracle Role Manager administration tools.

It contains the following topics:

- Overview of Oracle Role Manager Administration
- Displaying the Administrative Console

1.1 Overview of Oracle Role Manager Administration

Oracle Role Manager administration tools can be divided into the following categories:

- Configuration and Data Model Deployment
- Data Load
- System Identity Management

Each of these areas of administration can be performed on the command line; data load can also be initiated remotely using the Oracle Role Manager administrative console.

1.1.1 Configuration and Data Model Deployment

Oracle Role Manager configuration is stored in the database and must be deployed (copying the configuration into the database) before any data is loaded into the system. When installing Oracle Role Manager with the Install Software and Configure option, this deployment is done automatically. When installing Oracle Role Manager with the Install Software Only option, this must be performed manually.

Many administrators will select the Install Software Only option so that there is the opportunity to change the default configuration or customize the data model to prepare for loading data into an extended model. Refer to Chapter 2 for more information.

1.1.2 Data Load

The loading of data into Oracle Role Manager can be initiated directly from the command line using the `load.bat` or `load.sh` scripts and from the Oracle Role Manager administrative console.

The command-line scripts, provided for convenience, can be used for regularly scheduled, automated data loads. When using the administrative console, the Oracle Role Manager server must be deployed to the application server and running before the load process can be initiated. Refer to Chapter 3 for more information.

1.1.3 System Identity Management

System Identities are system users that can be used for access to the Oracle Role Manager system. System Identities normally represent external systems; one example could be a user-provisioning system that accesses Oracle Role Manager as a part of role resolution workflows or access provisioning processes; another example could be for simple data synchronization. Refer to Chapter 4 for more information.

1.2 Displaying the Administrative Console

The URL for the Oracle Role Manager administrative console, including the port number, is determined by the configuration of the application server on which the Oracle Manager server is deployed.

The URL typically includes the name of the application server host computer and the port number assigned during application server configuration.

For example, in WebSphere:

```
http://mgmthost1.acme.com:9080/ormconsole
```

Note: By default, there is only one user of the Oracle Role Manager administrative console, the Oracle Role Manager System Administrator account. The user name and password for this System Identity is set during initial deployment.

2 Component Configuration

This chapter includes information about the default configuration of the Oracle Role Manager server and how to modify these defaults.

This chapter includes the following sections:

- Understanding Default Server Configuration
- Default Configuration Files
- Deploying Customizations

2.1 Understanding Default Server Configuration

This section shows the default values that are set during initial deployment of Oracle Role Manager to help you determine whether you need to use different values for your installation.

Each configurable component of the Oracle Role Manager server has a corresponding XML file to use as a starting place, should you find that you need to modify the configuration. The configurable components in Oracle Role Manager are:

- Authentication
- Bootstrap
- Business Logic Plug-ins
- Cache
- Deployment
- Finalization
- Internationalization
- Timers

2.1.1 Authentication

Oracle Role Manager authentication configuration controls the form of accepted SSO tokens, encryption algorithm, System Identity credentials, and person credentials for direct access to the Oracle Role Manager Web UI. For more information about authentication, refer "About the Single Sign-On Configuration with Oracle Role Manager" on page 5-1.

Table 2–1 shows the default configuration for the Authentication component of Oracle Role Manager.

Table 2–1 Authentication Configuration Values

Element	Default Value
sso-token	Mapping between the `person` entity class and the `userID` attribute.
encryption-algorithm	oracle.iam.rm.authentication.util.SHAEncryption
system-credentials-mapping	Mapping between the `systemIdentity` entity class and the `userID` attribute for username, and between the `systemIdentity` entity class and the `userPassword` attribute for password.
user-credentials-mapping	Mapping between the `person` entity class and the `userID` attribute for username, and between the `person` entity class and the `userPassword` attribute for password
failure-policy	Defines the lockout attempt threshold, which is the maximum number of attempts a user can attempt for logging in.
lockout-attempt-threshold	The maximum number of attempts a user can attempt for logging in. The standard default value is 5.

2.1.2 Bootstrap

The Bootstrap configuration is used to initialize the core System Identities and the System Administrator role during initial deployment.

The entitlements for the roles set in this configuration are the minimum required to allow loading of other system roles and mappings to system entitlements. The bootstrap configuration defines two system identities: the System Administrator (the user that can log in to the system via the Web UI and command-line tools), and the System User (the identity of the Role Manager Server itself, which is used to perform system level background tasks, for example, Batch Role Resolution).

> **Note:** In the event where the initial state for these System Identities has been locked out, it can be recovered using the Rebootstrap tool. Refer to Section 4.5 for more information.

Table 2–2 shows the default configuration for the Bootstrap component of Oracle Role Manager.

Table 2–2 Bootstrap Configuration Values

Element	Default Value
system-admin	
display-name	System Administrator
unique-name	System Administrator
admin-role display-name	System Administrator
admin-role unique-name	System Administrator
admin-role delegatable	false
admin-role privileges	`systemRole` with `all` permission and `sysRolePrivilege` with `all` permission.
system-user	
display-name	System User

Table 2–2 (Cont.) Bootstrap Configuration Values

Element	Default Value
unique-name	System User

2.1.3 Business Logic Plug-ins

The configuration settings for Business Logic (BL) determine the cache size limit of plug-in packs and the time out value. You may need to increase the size limit of additional plug-in packs, which you have added.

The time out setting specifies the amount of time (in seconds) between submitting a business transaction for finalization and returning control to the user if the process is taking too long. You may want to shorten the value if you want the system to "fail" faster, or lengthen the value if time outs occur too frequently.

Table 2–3 shows the default configuration for the Business Logic Plug-in component of Oracle Role Manager.

Table 2–3 Business Logic Plug-in Configuration Values

Element	Default Value
plugin-cache-config size-limit	20 (number of cached plugin-pack objects)
finisher-config default-timeout-sec	60 seconds

2.1.4 Cache

You may want to reduce the heartbeat period (in milliseconds) for more frequent cleaning of the cache or increase the heartbeat period to handle a larger window when the cache is larger than configured.

Table 2–4 shows the default configuration for the cache component of Oracle Role Manager.

Table 2–4 Cache Configuration Values

Element	Default Value
heartbeat-period	5000 milliseconds

2.1.5 Deployment

The deployment configuration provides the information about which tablespaces must be used to deploy tables and indexes. By default tables and indexes are deployed to the database user's default tablespace. This configuration allows:

- the installer to define which tablespaces must be used
- the ModelManager to distribute the tables and indexes

Table 2–5 shows the default configuration values for the tablespace names used during deployment.

Table 2–5 Default Tablespace Configuration

Tablespace	Default Value
Data tables	ORM_DATA
Indexes	ORM_INDEX

2.1.6 Finalization

The Finalization configuration settings determine the expiration period and renewal period of the finalization lease. The expiration period is the amount of time (in milliseconds) a finalization node will be down before another node attempts to take its place; the smaller it is, the faster fail over will occur.

The renewal period is the amount of time (in milliseconds) between lease renewals; the smaller it is, the more "up to date" the lease is, however, this can cause more database traffic. The renewal period must be shorter than expiration period. If it is not shorter than expiration period, the lease can expire, causing fail over when the finalization server is still running, which will affect performance.

Table 2–6 shows the default configuration for the Finalization component of Oracle Role Manager.

Table 2–6 Finalization Configuration Values

Element	Default Value
lease-config expiration-period	15000
lease-config renewal-period	5000

2.1.6.1 Troubleshooting Finalization Lease Error

In the rare case when the application server crashes, it would not get a chance to cancel its finalization lease stored in the database.

Normally, the finalization lease expires after the lease-configuration period is over and the application server, when started is able to obtain a new finalization lease. However if the application server JVM crashes and if the database time is ahead of the application server, the finalization lease expires only after the time difference plus the lease configuration expiration period. This results in a pending state, where no transactions will get finalized in the database.

This pending state causes the deployment command *ORM_HOME/bin/deploy.sh* or `deploy.bat` to fail with the error message, "Could not obtain finalization lease: ensure that the server is down." This error condition persists even if you reboot all application servers and database systems. It is irrecoverable for the time difference period (a maximum period of one day) and recovers automatically after the time difference period is over.

To recover from this error condition, you need to direct the application server to obtain the lease by force:

> **Note:** You must perform this procedure only if the error condition persists even after rebooting all application servers and database systems.

Depending on the application server you are using, perform the following steps:

1. **For JBoss**:

   Edit the following file:

   JBOSS_HOME/bin/run.bat or run.sh

   Add the following:

   -Doracle.iam.rm.finalization_lease.obtain_by_force=true JVM property to JAVA_OPTS

2. **For WebLogic**:

   Edit the following file:

   WEBLOGIC_ORM_DOMAIN_HOME/bin/startWebLogic.cmd or startWebLogic.sh

   Add the following:

   -Doracle.iam.rm.finalization_lease.obtain_by_force=true JVM property to JAVA_OPTIONS

Note:

- This error does not occur for Oracle Role Manager on WebSphere, because the finalization lease is managed by WebSphere application server and not by the Oracle Role Manager database.
- After the lease is obtained by force, you may see JTA transaction abort warnings on server startup: "javax.transaction.RollbackException: Can't commit because the transaction is in aborted state."

  This is a benign message. The business transaction is not lost and it is processed when the lease is obtained.
- You must remove the JVM property from the application servers after the successful startup of the server.

2.1.7 Internationalization

The i18n configuration file provides the default information about cache configuration size limit and age-limit for the i18n configuration. You can use this file to change the default values.

By default there is only one i18n file bundled with Oracle Role Manager. If you add any i18n files, increase the size-limit to the number of i18n files. The age-limit controls the time interval, after which the cache will be refreshed. Table 2.1.7 shows the default values for the i18n configuration.

Table 2–7 Internationalization Default Configuration Values

Element	Default Value
cache-config size-limit	10 (number of 18n files in cache)
age-limit	18000 milliseconds

2.1.8 Timers

There are two configurable timer components in Oracle Role Manager, one for the main server, a singleton configuration for the timer subsystem as a whole. The second timer, for batch resolution can have several configurations, one per timer (identified by the job ID), used for integrations with external systems.

The main Timer configuration sets the thread pool property (refer to Table 2–8). Oracle recommends that this default value not be changed.

Table 2–8 Timer Configuration Values

Element	Default Value
thread-pool-property	5

The batch resolution job updates the user-to-role assignments calculated for complex dynamic roles (roles having complex rules that dynamically determine membership). The batch resolution timer can have multiple jobs configured (identified by the job ID).

The Batch Resolution Timer configuration sets preferences for the batch resolution job. Table 2–9 shows the default configuration values for setting the implementing Java class and whether the timer type is `simple` (defining a repeat interval of *n* milliseconds between invocations) or a `cron` timer (defining a UNIX-style cron timer). The default is the `simple` timer type. (Refer to Section 2.1.8.1 for more information about cron expressions.)

Table 2–9 Batch Resolution Timer Configuration Values

Element	Default Value
factory-classname	oracle.iam.rm.resolution.impl.BatchResolutionTimerFactory
job-id	BatchResolutionJob
singleton	true
simple repeat-interval	14400000
cron cron-expression	N/A

2.1.8.1 Cron Expressions

A `cron` expression is a string comprised of six or seven fields separated by white space. Fields can contain any of the allowed values, along with various combinations of the allowed special characters for that field. The fields in the expected order is shown in Table 2–10.

Table 2–10 Cron Expressions Allowed Fields and Values

Name	Required	Allowed Values	Allowed Special Characters
Seconds	Y	0-59	, - * /
Minutes	Y	0-59	, - * /
Hours	Y	0-23	, - * /
Day of month	Y	1-31	, - * ? / L W C
Month	Y	0-11 or JAN-DEC	, - * /
Day of week	Y	1-7 or SUN-SAT	, - * ? / L C #
Year	N	empty or 1970-2099	, - * /

Example 2–1 Cron Expressions

Cron expressions can be as simple as `* * * * ? *` or as complex as `0 0/5 14,18,3-39,52 ? JAN,MAR,SEP MON-FRI 2002-2010`.

Here are some more examples:

Expression	Means
0 0 12 * * ?	Fire at 12:00 PM (noon) every day
0 15 10 ? * *	Fire at 10:15 AM every day
0 15 10 * * ?	Fire at 10:15 AM every day
0 15 10 * * ? *	Fire at 10:15 AM every day

Expression	Means
0 15 10 * * ? 2005	Fire at 10:15 AM every day during the year 2005
0 * 14 * * ?	Fire every minute starting at 2:00 PM and ending at 2:59 PM, every day
0 0/5 14 * * ?	Fire every 5 minutes starting at 2:00 PM and ending at 2:55 PM, every day
0 0/5 14,18 * * ?	Fire every 5 minutes starting at 2:00 PM and ending at 2:55 PM, AND fire every 5 minutes starting at 6:00 PM and ending at 6:55 PM, every day
0 0-5 14 * * ?	Fire every minute starting at 2:00 PM and ending at 2:05 PM, every day
0 10,44 14 ? 3 WED	Fire at 2:10 PM and at 2:44 PM every Wednesday in the month of March
0 15 10 ? * MON-FRI	Fire at 10:15 AM every Monday, Tuesday, Wednesday, Thursday and Friday
0 15 10 15 * ?	Fire at 10:15 AM on the 15th day of every month
0 15 10 L * ?	Fire at 10:15 AM on the last day of every month
0 15 10 ? * 6L	Fire at 10:15 AM on the last Friday of every month
0 15 10 ? * 6L	Fire at 10:15 AM on the last Friday of every month
0 15 10 ? * 6L 2002-2005	Fire at 10:15 AM on every last friday of every month during the years 2002, 2003, 2004, and 2005
0 15 10 ? * 6#3	Fire at 10:15 AM on the third Friday of every month
0 0 12 1/5 * ?	Fire at 12 PM (noon) every 5 days every month, starting on the first day of the month
0 11 11 11 11 ?	Fire every November 11 at 11:11 AM

2.2 Default Configuration Files

To view the default configuration XML files, you will need to extract them from an archive file. You may want to use these files for convenience as a starting place for your configuration changes.

To get the default configuration files:

1. If you have not already extracted the default configuration files, extract them as follows:

   a. On the Oracle Role Manager installation host, navigate to *ORM_HOME*`/config`.

   b. Using an utility like WinZip or gunzip, extract the entire contents of `configurations.car` into a temporary location.

2. From the temporary location used to extract the files, navigate to `configurations/config`.

   This directory contains subdirectories for all the configurable Oracle Role Manager server components.

These can be modified and used as a starting place for customizing your configuration.

Note: The only difference between `configurations.car` and `configurations_hardened.car` is in the *config/oracle.iam.rm.bootstrap/default.xml file*, between system privileges assigned to the System Administrator.

2.3 Deploying Customizations

Oracle Role Manager configuration is stored in the database and must be deployed (copied into the database) before any data is loaded into the system.

If you need to alter the standard configuration or standard data model, you will need to run a command to deploy your customizations to the database.

Note: For information on how to upgrade the data model and configurations from an existing installation of Oracle Role Manager to the latest release, refer Oracle Role Manager Installation Guide.

This procedure assumes you have already completed the following steps:

- A database instance has been created for Oracle Role Manager with the appropriate tablespaces.
- The Oracle Role Manager database owner and application user schemas have been created.
- The service on which Oracle Role Manager is installed is started and the database is accessible.

Refer to the *Oracle Role Manager Installation Guide* for more information about these assumptions.

To deploy model and configuration customizations:

1. Create an archive file containing your customizations and append the file name with `.car`.
2. In *ORM_HOME*`/config`, edit the following two lines in the `db.properties` file to match your environment:

   ```
   db.driverClass=oracle.jdbc.driver.OracleDriver
   db.connection_string=jdbc:oracle:thin:@//$HOST$:$PORT$/$SERVICE_NAME$
   ```

   where `$HOST$` is the database host name, `$PORT$` is the database listener port, and `$SERVICE_NAME$` is the database service name on which the Oracle Role Manager users/schemas were created.
3. In a command window, navigate to *ORM_HOME*`/bin`.
4. Run the following command to deploy the configuration and data model and create the root entities.

   ```
   deploy "<collection_of_cars>" <orm-owner> <ormapp-user> <admin-user>
   ```

   where:

   `<collection_of_cars>` contains the relative paths and file names of all CAR files to deploy, for example, in a customized deployment, on a UNIX-based system:

   ```
   ../config/configurations_custom.car:../config/oim_integration_custom.car
   ```

This collection must be within quotes with delimiters appropriate to the platform (a semicolon (`;`) for Windows, otherwise a colon (`:`)).

<orm-owner> is the username of the Oracle Role Manager database owner user/schema.

<ormapp-user> is the username of the Oracle Role Manager application user/schema.

<admin-user> is the username of the Oracle Role Manager System Administrator to create.

5. At the prompts, type the passwords of the Oracle Role Manager database owner, Oracle Role Manager application user, and Oracle Role Manager Administrator account.

2.4 Logging Configuration

The logging.properties file determines the logging messages of the command-line tools. It provides the information about logging level, filename, and location.

To configure logging, reset the following default configuration values:

- Set the global logging level using the following syntax:

```
.level = INFO
```

- Set the logging level for messages that are printed on the console to `FINE` and above:

```
 java.util.logging.ConsoleHandler.level = FINE
```

- Set the logging level for messages that are printed in the file to FINE and above:

```
java.util.logging.FileHandler.level = FINE
```

- Set the file size limit (in bytes) and the number of most recent log files to be retained. The product of these two parameters (limit * count) must be less than your free disk space.

```
java.util.logging.FileHandler.limit = 100000
```

```
java.util.logging.FileHandler.count = 10
```

> **Note:** A reasonable file size limit is 5000000 (~5MB) with a count of 200, which makes a total of about 1 GB.

- Set the hierarchy indexing manager logger to log `FINE` messages, for example:

```
oracle.iam.rm.hierarchy.level = FINE
```

For more information about logging configuration, visit the Java Web site at

http://java.sun.com/j2se/1.4.2/docs/guide/util/logging/overview.html.

2.5 Abandoned Transaction Cleanup

Abandoned transactions are those pending transactions which have seen no activity in a configurable time-to-live period. The transactions are abandoned either because of a network problem between Role Manager client and server or the user of Role Manager navigates away from the transaction page without completing the transaction. Role

Manager uses a configurable scheduled task to cleanup such abandoned transactions. The following factors are considered to cleanup the abandoned transactions:

- Any pending transaction that has no activity within the time-to-live window is eligible for cleanup. However, the actual cleanup only occurs when the scheduled task for cleanup is run.
- An excessively short time-to-live window will interfere with normal user activities. Therefore Oracle recommends a time-to-live value of at least 1 hour.

You can consider these two factors to configure the scheduled task. The default time at which the scheduled task is set to run is 1 a.m. and time-to-live value is 1 hour. These values can be set by unpacking the configurations.car file and editing the `oracle.iam.rm.timer.abandonedTransactionCleanupTimer.xml` file. The following is the default configuration file:

```
<?xml version="1.0" encoding="UTF-8"?>
<timer-config xmlns="http://xmlns.oracle.com/iam/rm/timer/config/1_0">
    <job-configs>
        <job-config>
<factory-classname>oracle.iam.rm.bizxact.impl.AbandonedTransactionCleanupFactory</
factory-classname>
            <job-id>AbandonedTransactionCleanupJob</job-id>
            <group-id>TransactionGroup</group-id>
            <parameters>
                <parameter>
                    <id>timeToLive</id>
                    <integer>60</integer>  <!-- hourly, represented by
minute-granularity -->
                </parameter>
            </parameters>
            <singleton>true</singleton>
            <!--
            The default invocation interval is 0 0 1 * * ?
            This cron-style expression translates to 1:00 AM every day.
            Refer to the Oracle Role Manager Administrator's Guide for more
information.
            -->
            <cron>
                <cron-expression>0 0 1 * * ?</cron-expression>
            </cron>
        </job-config>
    </job-configs>
</timer-config>
```

3 Data Load

This chapter provides procedures for initial load of data into Oracle Role Manager. The data loader can be used to load data into the system. For information about implementing special processing as part of a load procedure, contact your Oracle Consulting Services representative.

This chapter assumes you have deployed the standard data model provided with Oracle Role Manager or a custom model built on the standard model. It also assumes that you understand the business requirements associated with the data that must be loaded into Oracle Role Manager.

It contains the following topics:

3.1 Load Process Overview

To determine the best approach for loading data into Oracle Role Manager, it is important to understand the overall load process along with the sample scripts and procedures provided with Oracle Role Manager.

The overall load process of data into Oracle Role Manager (see Figure 3–1) involves the following components:

- Load request

  The load request defines which load procedures should run as part of the data load. This file also specifies the order for loading objects in the required sequence.

- Data files

  Normally in CSV format (although any character delimiter is supported), data files contain the actual data to load into Oracle Role Manager. For more information about data files, refer "Load Requests" on page 3-9.

> **Note:** Oracle does not recommend you to use Microsoft Excel to edit the CSV file, because it inserts extra quotes when you insert double quotes in the file.

- Load procedures

  Load procedures contain the object creation and relationship creation procedures that map to the Business Logic (BL) definitions. Load procedures are a clean representation of the default load operations, uncluttered by the system-level details contained in the BL definitions. For more information about load procedures, refer "Default Loader Procedures (Standard Model)" on page 3-11.

- Business logic definitions

  The BL definitions contain detailed procedures representing the default loadable objects, attributes, and relationships and the XML mapping to Business Logic Plug-ins for business operations called by the loader request. For more information about Business Logic Plug-ins, refer "Business Logic Plug-ins" on page 2-3. For information about car file, refer to Step 4 of "Loading Data From an External Database" on page 3-5.

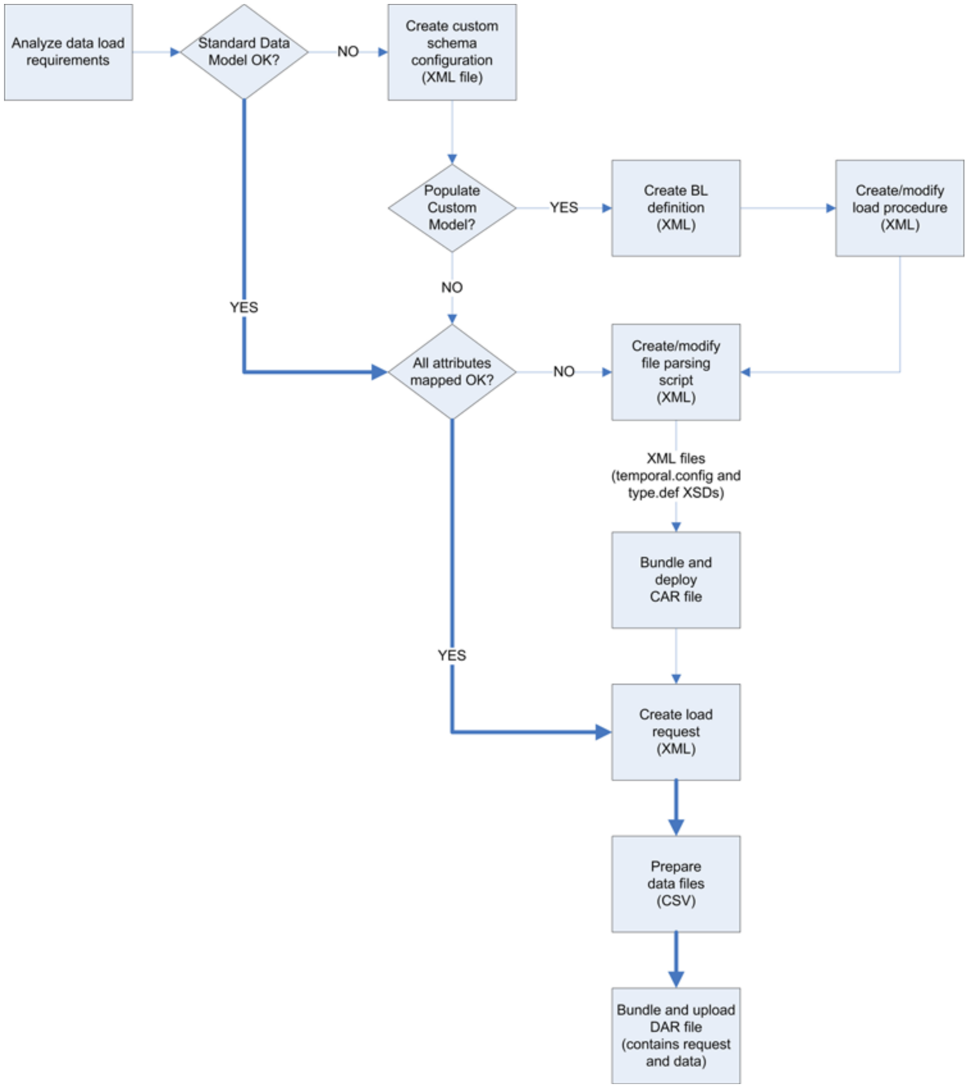
- File parsing scripts

  These scripts contain mappings to load procedures and the load sequence of input parameters (attributes) within the load operation relative to object type. This commonly includes only a subset of the object's attributes into which to load data.

All data loaded into Oracle Role Manager is processed by the BL layer to ensure that various constraints are enforced at load time. For example, IT roles in Oracle Role Manager only gain membership through business role mappings. This is enforced by having the object definitions in the load procedures match those in the BL definition files. Another example is that the BL definition for the person object contains either the same or a superset of attribute definitions as those included in the createPerson load procedure.

If you have data to load into custom object definitions or custom attributes, you will need to add new business logic and load procedures.

Figure 3–1 Overall Data Load Process


Analyze data load requirements
Standard Data Model OK?
NO
Create custom schema configuration (XML file)
Populate Custom Model?
YES
Create BL definition (XML)
Create/modify load procedure (XML)
NO
YES
All attributes mapped OK?
NO
Create/modify file parsing script (XML)
XML files (temporal.config and type.def XSDs)
Bundle and deploy CAR file
YES
Create load request (XML)
Prepare data files (CSV)
Bundle and upload DAR file (contains request and data)

This is a flowchart for the overall data load process.

****

3.2 Data Load Scenarios

Before loading data, there are three questions you should ask to help identify the approach to take in loading your data:

1. Does the deployed data model contain all the object types and attributes you want to load?
2. Do the standard load procedures for each of your object types contain all of the operations you need?
3. Do the load operations in file parsing scripts for each of your object types contain all of the attributes you want to load?

The following three examples describe the possible business scenarios around initial data loads into Oracle Role Manager. Choose the most suitable scenario, which will identify the steps that you will need to follow. (For information about the Oracle Role

Manager standard defaults, refer "Default Loader Procedures (Standard Model)" on page 3-11.)

For each of these examples, you can refer to Figure 3–1 to help visualize the load process flow for your deployment of Oracle Role Manager.

Example 3–1 The data you want to load already maps to the standard data model and standard load procedures.

The standard data model and the file parser scripts must contain all the object types and attributes that you want to load and there are no model changes required to load your data set.

Even if your business model requires data model extensions, because you don't need to load data into the extended schema, you can still use the sample scripts.

This example requires the following steps:

1. Create load request.
2. Prepare data files.
3. Bundle and upload data with request.

Example 3–2 The standard data model supports the data you want to load, but the attributes in file parser load scripts aren't what you want.

In other words, the mismatch is only in the way the file parser script orders or maps the subset of attributes for a particular of multiple object types. For example, if the person file parser script maps six attributes for person data and you want to load twelve attributes.

As in Example 3–1, if the data model has been extended but the data you want to load is not part of the custom model, fewer components are involved in the load process.

This example requires the following steps:

1. Create new file parser scripts from existing sample scripts.
2. Bundle and deploy new configuration.
3. Create load request.
4. Prepare data files.
5. Bundle and upload data with request.

Example 3–3 The data to load must go into a custom model.

Whether your data model extensions into which you want to load data are an added attribute or a new object type, loading data into a custom model requires supporting business logic and new process definitions.

This example assumes the data model has already been extended and requires the following steps:

1. Create BL definitions.
2. Create load procedures.
3. Create file parser scripts.
4. Bundle and deploy new configuration.
5. Create load request.
6. Prepare data files.

7. Bundle and upload data with request.

3.3 Loading Data From an External Database

You can load the data into Role Manager from an external database by performing the following steps:

1. Write an SQL query to select the data that you want to load into Role Manager. Your query should return all attributes that Role Manager requires for the kind of object being loaded. Assign a name to each column of the query returned. The following is an example query that returns the required attributes for Role Manager PERSON objects:

Example 3–4 Query that returns attributes for Role Manager PERSON objects.

```
select 'A000001' as id,
 'John' as first_name,
 'Smith' as last_name,
 'John Smith' as display_name,
 'active' as status
from dual
```

2. Define a data source that your application server can use to execute queries. The procedure for doing this varies from one application server to the other.
3. Create a load script that will execute your query and return query results to the appropriate Role Manager task.

   For example a load script named persondb_script.xml creates a procedure named loadPersonsFromDB. The version of the script and its dependencies are declared. In this case the business logic dependency was copied from another load script. If you go through several iterations of the script while debugging, it is important to increment the version number whenever the script is changed. When deploying the script, the new version number signals Role Manager that any previous versions are obsolete. If the version number is not incremented, then the deploy task exits without deploying the new script.

> **Note:**
> - The loadPersonsFromDB procedure includes the query developed in the example and uses the JNDI name for the data source that will execute the query. This procedure will call the Role Manager loader's standard "createPerson" procedure.
> - In the input-params section of the script columns in the query's result set are mapped to the parameters of the createPerson procedure.

Example 3–5 Load Data

```
<!-- persondb_script.xml -->
<?xml version="1.0" encoding="UTF-8"?>
<load-script xmlns="http://xmlns.oracle.com/iam/rm/loader/script/1_0"
xmlns:t="http://xmlns.oracle.com/iam/rm/type/def/1_0"
id="persondb_script" version="10.1.4.6">

<dependencies>
```

```
<business-logic-dependency def-id="bizlogic.sample" version="10.1.4.1"/>
</dependencies>

<procedures>
<procedure id="loadPersonsFromDB">
<operations>
<database-load id="database"
datasource="java:/ExternalDS"
query-sql="select 'A000002' as id, 'John' as first_name, 'Smith' as last_name,
'John Smith' as display_name, 'active' as status from dual">
<procedure-call id="call"
procedure-id="createPerson"
script-id="procedures">

<input-params>
<column name="givenName" column-name="first_name"><t:string-ext/></column>
<column name="sn" column-name="last_name"><t:string-ext/></column>
<column name="displayName" column-name="display_name"><t:string-ext/></column>
<column name="uniqueName" column-name="id"><t:string-ext/></column>
<column name="status" column-name="status"><t:string-ext/></column>
</input-params>
</procedure-call>
</database-load>
</operations>
</procedure>

</procedures>

</load-script>

<!-- end persondb_load.xml>
```

4. Deploy the load script. You can do this by creating a car file that contains just the script(s) that loads the external data. First put the load script in a directory with the path name that loader expects (config/oracle.iam.rm.loader). A test directory (test_dbload) was used to isolate this experiment from the Role Manager installation:

   test_dbload\config\oracle.iam.rm.loader

   To create the car file:

   a. Change to the parent of the config directory (test_dbload). Use zip to create the car file:

      jar -cvf test_dbload.car config

   b. Copy testdb_load.car to the *ORM_HOME*/config directory.

   c. Make sure the appserver has been stopped and then deploy the new car file using *ORM_HOME*/bin/deploy.bat:

      deploy.bat "..\config\test_dbload.car" ormowner ormuser admin

5. Create a load request that calls the loadPersonsFromDB procedure. Example 3–6 calls "persondb_script" and "loadPersonsFromDB". No parameters are required for this load, but it seems that a parameter section is required even if it is empty. The ordering mode ("trusted-sequential") was copied from another load script.

Example 3–6 Load Request

```
<!-- load-request.xml -->
```

```
<?xml version="1.0" encoding="UTF-8"?>
<load-request xmlns="http://xmlns.oracle.com/iam/rm/loader/data/1_0"
load-script-id="persondb_script" procedure-id="loadPersonsFromDB"
ordering-mode="trusted-sequential">

<parameters>
</parameters>
</load-request>
```

6. Create a DAR file to contain the load request. Use zip to create the dar file:

   zip dbtest.dar load-request.xml

7. Load the data by performing the following steps:

   a. Start the application server.

   b. Open ormconsole in a web browser and click **Upload**.

   c. Enter the username and password for the admin user and browse for dbtest.dar.

   d. Click **Load**.

3.3.1 Data Loading from the Command Line

The load.sh file is designed to allow loading DAR files in the Role Manager console in a command-line pattern. Using this feature, you can load DAR files automatically. The following is the command used to load the DAR file:

```
load.bat server_url dar_file orm_username
```

An example for this command is:

```
load.bat http://localhost:8080/ ../data/my_people.dar admin
```

When you execute this command, the password for the administrator user is prompted. If you are loading the DAR file automatically, you can avoid the password prompt using the following command:

```
load.bat http://localhost:8080/ ../data/my_people.dar admin/admin123
```

Running the automated load process may raise a security issue. This is because, users with access to the computer running the tool can see the administrator's password in the process list. To avoid compromising, Oracle recommends that you create a custom system identity for data loading and grant that identity a role configured with the minimum system privileges for the data to be loaded. The following example illustrates this scenario.

Example 3–7 Loading Data With Custom System Identity

Assume that the command-line tool is only used for loading people as part of reconciliation process. The system identity used for the tool must be able to run the person reconciliation business operation. To enable this and to limit the impact of this user's credentials being exposed, perform the following:

- Create a new permission called "reconcile" and associate it with the person object type.
- Create a new business operation for the person attribute reconciliation and assign "reconcile" privilege on person.
- Create a new loader script that invokes the new business operation.

- Create a new system role and associate it with the new "reconcile person" privilege.
- Create a new system identity and grant the new system role to it. For more information about system identity, refer to Chapter 4.

The newly created system identity can now only be used to load person details to be reconciled.

> **Note:** The system identity created for loading person details to be reconciled cannot be used for loading anything other than person details.

3.4 Understanding the Standard Model (Default)

The default loader components can be used to understand the data loading process. This section describes these components in more detail and shows you where to find the scripts that you will either use by default or use as starting places, should you need to create new ones.

It may be useful to familiarize yourself with the standard data model along with any schema extensions that are planned or already deployed.

In this section:

- Sample Loader Scripts and Standard Model Description
- Default Loader Procedures (Standard Model)
- Business Logic Definitions
- File Parsing Scripts
- Load Requests

3.4.1 Sample Loader Scripts and Standard Model Description

To view the sample loader scripts and procedures, you will need to extract them from the archive files provided in your Oracle Role Manager installation. You may want to refer to these file on an ongoing basis or you may want to use them for convenience as a starting place for your customized load processes.

To get the sample loader scripts and related files:

1. On the Oracle Role Manager installation host, navigate to *ORM_HOME*/config.
2. Using an utility like WinZip or gunzip, extract the entire contents of `standard.car` into a temporary location.
3. In the temporary location used to extract the files, navigate to `config/oracle.iam.rm.loader`.

   These are copies of the files that are used when running the standard data procedure.
4. From the same location, navigate to `config/oracle.iam.rm.bizlogic.def`.

   You will see the `bizlogic.loader.xml` file that is referenced when running the standard data procedure.
5. From the same location, navigate to `config/oracle.iam.rm.temporal`.

You will see the `standard.xml` file that represents the data model supporting the loader and the Oracle Role Manager Web UI.

These can be modified and used as a starting place for custom procedures.

3.4.2 Load Requests

Load requests specifies which load operations to run for a particular data load, while mapping the load operations to data files bundled with the load request in a DAR file (data archive).

Example 3–8 Load Request

```
<load-request load-script-id="reporting_script"
procedure-id="buildReportingHierarchy">
  <parameters>
    <resource-ref name="reporting_file">
      <resource-path>reporting.txt</resource-path>
    </resource-ref>
  </parameters>
</load-request>
```

Load requests must be contained in a single file to ensure the operations are run in the correct sequence. The supported sequence of operations is shown in Table 3–1.

Table 3–1 Required Sequence of Load Operations

Operations in Sequence	File Parser
Roles	
loadBusinessRoles	business_role_script
(Maps first to the data file containing the business roles, then to membership rules data, and finally to eligibility rules data.)	
loadSystemRoles	system_role_script
loadITRoles	it_role_script
loadApprovers	approver_script
loadITPrivileges	it_privileges_script
Note: Although in the user interface, the term `ITPrivilege` has been changed to `Entitlement`, it remains unchanged in the script.	
loadITRolePrivilegeMappings	itrole_to_privilege_script
loadSystemRolePrivilegeMappings	systemrole_to_privilege_script
loadBusinessRoleToItRoleMappings	businessrole_to_itrole_script
Organizations	
loadOrganizationsWithParents	organization_script
loadOrganizationalUnitsWithParents	organizational_unit_script
loadPersons	person_script
buildPersonReportingHierarchy	reporting_person_script
buildLocationHierarchy	location_script
buildCostCenterHierarchy	cost_center_script

Table 3–1 (Cont.) Required Sequence of Load Operations

Operations in Sequence	File Parser
buildReportingHierarchy	reporting_script
RoleGrants	
loadBusinessRoleGrants	business_role_grant_script
loadSystemRoleGrants	system_role_grant_script

> **Note:** Relationships between objects cannot be created until those objects already exist, so depending on these relationships, sequence can be an important relative to the business logic of the load operations.

3.4.3 File Parsing Scripts

The file parsing scripts determine which attributes to load and in what order. The matching data files must use the same order. It is recommended that there be a single parsing script for each entity type.

In the following example, note the input parameters in the operations section. This is where the order is specified.

Example 3–9 File Parser (reporting_script)

```
<?xml version="1.0" encoding="UTF-8"?>
<load-script xmlns="http://xmlns.oracle.com/iam/rm/loader/script/1_0"
             xmlns:t="http://xmlns.oracle.com/iam/rm/type/def/1_0"
        id="reporting_script" version="10.1.4">
<dependencies>
  <script-dependency script-id="procedures" version="10.1.4"/>
</dependencies>

<procedures>
  <procedure id="buildReportingHierarchy">
    <input-params>
      <input-param name="reporting_file">
        <t:binary>
          <t:non-null-constraint id="non-null">
            <t:violation-message>The binary must be
provided.</t:violation-message>
          </t:non-null-constraint>
        </t:binary>
      </input-param>
    </input-params>
    <operations>
      <file-load id="file" file-param="reporting_file">
        <string-tokenizer string-delimiter="^" token-separator=",">
          <data-events>
            <data-event id="add_reporting">
              <procedure-call id="call" procedure-id="addToReportingHierarchy"
script-id="procedures">
                <input-params>
                <token name="child-name" index="0"><t:string-ext/></token>
                <token name="parent-name" index="1"><t:string-ext/></token>
                </input-params>
                </procedure-call>
              </data-event>
            </data-events>
          </string-tokenizer>
        </file-load>
      </operations>
    </procedure>
  </procedures>
</load-script>
```

3.4.4 Business Logic Definitions

The business logic (BL) definitions further define the allowable operations that can be invoked by load requests. These definitions, in a single XML file (`bizlogic.loader.xml`), contain the same operations as those in the load procedures file (see Table 3–2) yet also include further details such as:

- Load confirmation text and argument mappings used for audit messages
- The plug-in configuration containing the ID used to execute the load operation

Example 3–10 Plug-in Configuration (addToReportingHierarchy)

```
<snapshot-logic-definition
plugin-pack-id="oracle.iam.rm.bizlogic.plugin.standard_ext"
plugin-id="add_to_hierarchy">
  <ext config-version="1.0">
    <config>
      <![CDATA[
        <add-to-hierarchy
xmlns="http://xmlns.oracle.com/iam/rm/bizlogic/plugin/standard_ext/1_0"
          hierarchy-type="reportingHierarchy"
          parent-id-attribute-name="reportingOrg_id"
          root-id-attribute-name="reportingHierarchyRoot_id">
          <attributes>
            <attribute attribute-id="child-name" argument-id="child-name"/>
            <attribute attribute-id="parent-name" argument-id="parent-name"/>
          </attributes>
        </add-to-hierarchy>
      ]]>
    </config>
  </ext>
</snapshot-logic-definition>
```

In the preceding example, the plug-in ID is `add_to_hierarchy` and the configuration specifies the hierarchy type and the relationship paths, which allows that the Java plug-in class that handles this operation can be used for adding any object to any hierarchy, if it's supported by the schema.

3.4.5 Default Loader Procedures (Standard Model)

The default loader procedures, in a single XML file (`procedures.xml`), provide a convenient view into the standard data model as it relates to the default load operations. This file maps procedures to the business logic operations that can be called by load requests.

The load procedures define data loading operations that will be called in a load request (see Table 3–2) that can be used to create objects and relationships between those objects. The load procedures also illustrate data loading for all the attributes and object types in the standard data model. Example 3–11 illustrates procedure for data loading operation, `addToReportingHierarchy` defined in the definition-id, `bizlogic.loader`.

Example 3–11 Load Procedure (addToReportingHierarchy)

```
<procedure id="addToReportingHierarchy">
  <input-params>
    <input-param name="child-name">
      <t:string>
        <t:length id="length" max-length="256">
          <t:violation-message>The organization's name can be no longer than 256 
characters.</t:violation-message>
        </t:length>
      </t:string>
    </input-param>
    <input-param name="parent-name">
      <t:string>
        <t:length id="length" max-length="256">
          <t:violation-message>The parent's name can be no longer than 256 
characters.</t:violation-message>
        </t:length>
      </t:string>
    </input-param>
  </input-params>
  <operations>
    <business-transaction-operation id="add_to_reporting_hierarchy" 
definition-id="bizlogic.loader" operation-id="addToReportingHierarchy">
      <input-params>
        <param name="child-name" param-name="child-name"> <t:string-ext/></param>
        <param name="parent-name" param-name="parent-name"> 
<t:string-ext/></param>
      </input-params>
    </business-transaction-operation>
  </operations>
</procedure>
```

Note: For File Parser definitions, refer to the oracle.iam.rm.loader file in the following location:

ORM_HOME/config/standard.car/config/oracle.iam.rm.loader

Table 3–2 Default Load Procedures in the Standard Model

Operation	Description	Attributes	File Parser
addManagerToPersonHiearchy	Creates a relationship between a manager and a person.	This operation requires the following attributes: ▪ child_email—email of the managed person. ▪ parent_email—email of the manager. **Note**: The default procedure identifies the person and the manager by their email attributes. If your data file contains the `uniqueName` attribute instead, you may need to first create and deploy your custom loader scripts. You need to add your own custom BL and procedure files and call the procedure from your `load-request.xml`.	person_manager_script
addPersonToReportingHiearchy	Creates a relationship between a person and a reporting hierarchy (organization in the reporting hierarchy).	This operation requires the following attributes: ▪ uniqueName—Name representing the person. ▪ parent-name—Name of parent organization in the reporting hierarchy.	reporting_person_script
addOrgHeadToOrganization	Creates a relationship between an organization head and the organization.	This operation requires the following attributes: ▪ org-name (displayName)—Name of the organization to which to add the org head. ▪ orgHead-mail (uniqueName)—Identifier representing the organization head.	organization_head_script
addToCostCenterHierarchy	Creates a relationship between an organization and the cost center hierarchy.	This operation requires the following attributes: ▪ child-name—Name of the organization to add to the cost center hierarchy. ▪ parent-name—Name of parent organization in the cost center hierarchy.	cost_center_script
addToLocationHierarchy	Creates a relationship between an organization and the location hierarchy.	This operation requires the following attributes: ▪ child-name—Name of the organization to add to the location hierarchy. ▪ parent-name—Name of parent organization in the location hierarchy.	location_script

Table 3–2 (Cont.) Default Load Procedures in the Standard Model

Operation	Description	Attributes	File Parser
addToReportingHierarchy	Creates a relationship between an organization and the reporting hierarchy.	This operation requires the following attributes: ■ child-name—Name of the organization to add to the reporting hierarchy. ■ parent-name—Name of parent organization in the reporting hierarchy.	reporting_script
createApprover	Creates an Approver Role with the attributes.	This operation requires the following attributes: ■ displayName—User-readable name of the Approver Role. ■ uniqueName—Identifier for the Approver Role. ■ description—Text description of the Approver Role. ■ eligibilities—Grant Policy eligibility rule in XML format. ■ membershipRule—Membership rule in XML format. ■ roleType—Either `dynamic` or `static` (required).	approver_script
createApproverRoleGrant	Grants an Approver Role to a person with the attributes.	This operation requires the following attributes: ■ uniqueName—identifying the person to whom to grant the Approver Role. ■ role_title (displayName)—Name of the Approver Role to grant.	approver_role_grant_script

Table 3–2 (Cont.) Default Load Procedures in the Standard Model

Operation	Description	Attributes	File Parser
createBusinessRole	Creates a Business Role with the attributes.	This operation requires the following attributes: ▪ description—Text description of the Business Role. ▪ displayName—User-readable name of the Business Role. ▪ eligibilities—Grant Policy eligibility rule in XML format. ▪ approverType—Approver for the Static Business Role. It can be `roleowner`, `specificperson`, or `noapproval`. ▪ isDelegatable—Either `true` or `false` depending on whether the Business Role can be delegated to another user by the grantee. If not specified, the default is `false`. ▪ membershipRule—Membership rule in XML format. ▪ roleType—Either `dynamic` or `static`. ▪ socHierarchyType (socHierarchy_id)— Name of the Business Role that can be used for sphere of control for the role. ▪ uniqueName—Name for the Business Role.	business_role_script
createBusinessRoleGrant	Grants a Business Role to a person by creating a role grant relationship with the attributes.	This operation requires the following attributes: ▪ uniqueName—Name of the person to whom to grant the Business Role. ▪ role_title (displayName)—Name of the Business Role to grant.	business_role_grant_script
createBusinessRoleToItRoleMapping	Maps a Business Role to an IT Role by creating a relationship with the attributes.	This operation requires the following attributes: ▪ businessRole (displayName)—Name of the Business Role to map. ▪ itRole (displayName)—Name of the IT Role to map.	business_to_itrole_script

Table 3–2 (Cont.) Default Load Procedures in the Standard Model

Operation	Description	Attributes	File Parser
createITRole	Creates an IT Role with the attributes. If you have deployed the data model extensions for the integration with Oracle Identity Manager, additional attributes can also be loaded. Refer to the *Oracle Role Manager Integration Guide for Oracle Identity Manager* for more information.	This operation requires the following attributes: ■ displayName—User-readable name of the IT Role (required) ■ isDelegatable—Either `true` or `false` depending on whether the role can be delegated to another user by the grantee. If not specified, the default is `false`. ■ roleType—Either `dynamic` or `static`. ■ uniqueName—Identifier for the IT Role. ■ isFinanceRelated—Either true or false. If not specified, the default is false. ■ isHighRisk—Either true or false. If not specified, the default is false. ■ isNpiRelated—Either true or false. If not specified, the default is false. ■ isSoxRelated—Either true or false. If not specified, the default is false.	it_role_script
createITPrivilege **Note**: Although in the user interface, the term `ITPrivilege` has been changed to `Entitlement`, it remains unchanged in the script.	Creates an ITPrivilege with the attributes.	This operation requires the following attributes: ■ displayName—Name of the Entitlement. ■ oimEntitlementId—Unique identifier of the corresponding Entitlement in Oracle Identity Manager. ■ ITPrivilege Details—Used for additional content from external systems. ■ resourceName—Name of the resource. ■ uniqueName—Name for the ITPrivilege.	it_privilege_script

Table 3–2 (Cont.) Default Load Procedures in the Standard Model

Operation	Description	Attributes	File Parser
createITRolePrivilegeMapping	Maps an IT Role to an IT Privilege by creating a relationship with the attributes.	This operation requires the following attributes: ■ itPrivilege (displayName)—Name of the IT Privilege to map. ■ itRole (displayName)—Name of the IT Role to map. **Note**: Although in the user interface, the term `ITPrivilege` has been changed to `Entitlement`, it remains unchanged in the script.	itrole_to_privilege_script
createOrganization	Creates an Organization with the attributes.	This operation requires the following attributes: ■ displayName—User-readable name of the Organization. ■ uniqueName—Name representing the Organization.	organization_script
createPerson	Creates a `person` object with the attributes.	This operation requires the following attributes: ■ displayName—User-readable full name of the person. ■ givenName —First name of the person. ■ mail—E-mail address of the person. ■ sn—Surname (family name) of the person. ■ status—Either `active` or `inactive`. If not provided, the default is `inactive`. ■ userID—User name used to log on to the Oracle Role Manager system. ■ userPassword—Password used for authentication for access to Oracle Role Manager. ■ uniqueName—Name representing the person.	person_script
createOrganizationalUnit	Creates an Organizational Unit with the attribute.	This operation requires the following attributes: ■ displayName—User-readable name of the Organizational Unit. ■ uniqueName—Name representing the Organizational Unit.	organizational_unit_script

Table 3–2 (Cont.) Default Load Procedures in the Standard Model

Operation	Description	Attributes	File Parser
createSystemRole	Creates a System Role with the attributes.	This operation requires the following attributes: ■ displayName—User-readable name of the System Role. ■ isDelegatable—Either `true` or `false` depending on whether the System Role can be delegated to another user by the grantee. If not specified, the default is `false`. ■ roleType—Either `dynamic` or `static`. ■ socHierarchyType (socHierarchy_id)— Name of the hierarchy that can be used for sphere of control for the role. ■ uniqueName—Identifier for the System Role.	system_role_scrip t
createSystemRole Grant	Grants a System Role to a person by creating a role grant relationship with the attributes.	This operation requires the following attributes: ■ rootSocBinding and orgSocBinding (displayName)—Name of the organization to which to binds the sphere of control of the role grant. ■ uniqueName—Name of the person to whom to grant the System Role. ■ role_title (displayName)—Name of the System Role to grant.	system_role_grant _script
createSystemRole PrivilegeMapping	Maps a System Role to a System Privilege by creating a relationship with the attributes.	This operation requires the following attributes: ■ systemPermission and systemResource (displayName)—Name of the System Privilege to map. ■ systemRole (displayName)—Name of the System Role to map.	systemrole_to_pri vilege_script

Note: Refer to the `standard.xml` file to see constraint information for each attribute in these load operations.

3.5 Configuring the DAR File Size

By default the largest DAR file you can load is 10MB, loading a larger file results in an error. You can configure the maximum DAR file size using the following instructions.

For WebLogic Server

To configure the DAR file size for WebLogic server:

1. Go to Environment, Servers, ORM Server.
2. On the **Configuration** tab, click the **Server Start** subtab.
3. In the **Arguments** field, append the following argument to the new value.

   ```
   -Doracle.iam.rm.loader.max_upload_size=<new value>
   ```

4. Restart the WebLogic application server.

For JBoss Server

To configure the DAR file size for JBoss server:

1. Edit the config file:

   ```
   JBOSS_HOME/bin/run.bat
   ```

2. Add the following argument to JAVA OPTS:

   ```
   -Doracle.iam.rm.loader.max_upload_size=<new_value>
   ```

3. Restart the JBoss application server.

For WebSphere Server

To configure the DAR file size for WebSphere server:

1. Go to Servers, Application Servers, ORM Server.
2. In the Server Infrastructure section, expand **Java and Process Management**, and then click **Process Definition**.
3. In the Additional Properties section, click **Java Virtual Machine,** and then click **Custom Properties**.
4. Click **New** and enter the following information:
   a. In the **Name** field, type `oracle.iam.rm.loader.max_upload_size`.
   b. In the **Value** field, type the maximum size of data upload that you want to set, for example, 2 byte.
   c. In the **Description** field, type the description for the maximum upload size that you set, for example, maximum size limit for the Oracle Role Manager loader.
   d. Click **Ok**.
   e. Restart the WebSphere application server.

3.6 Preparing Data Files

The data files that you bundle with the load request must match the file names specified in the load request.

Data files, normally text files in comma-separated (",") format and string delimiter for all object types in caret ("^") format, contain actual data to load into Oracle Role Manager. Data files can use any character as a delimiter if it is set as the `token-separator` attribute in the script. The order of data, separated by the delimiter (with no spaces) must match the order of the input parameters in the respective file parsing script.

It is recommended that you separate the data files by type of entity and relationship to have enough flexibility to load them in the correct sequence.

To prepare your data files, bundle them with the load request as a DAR (data archive) file as follows:

1. In a command window, change to the directory containing the `loader-request.xml` file and other data files.
2. Run the following command:

   ```
   zip -r custom _data.dar
   ```

> **Note:** Ensure the loader request refers to data files that exist. For example, your person data file might have a different file name than `person.txt`, the sample person data file.

3.7 Running the Data Loader

Initiating the load process involves several steps to prepare the archive files expected by the loader. In addition, the Oracle Role Manager server must be running on the application server. (Refer to the *Oracle Role Manager Installation Guide* for more information.)

To run the loader:

1. If you have customizations:
   a. Make sure that the Oracle Role Manager users/schemas exist on the database.
   b. Create a CAR file (configuration archive with `.car` extension) containing the new BL definitions, load procedures or file parsing scripts.
   c. Deploy the configuration using the procedure described in Section 2.3.
2. Create a DAR file (data archive with `.dar` extension) containing the data files with the loader request file.
3. Deploy the Oracle Role Manager server to your application server as described in the *Oracle Role Manager Installation Guide*.
4. In a web browser, go to the application server host and port used for the Oracle Role Manager data loader. For example:

   ```
   http://<host>:8080/ormconsole
   ```

5. Type the user name and password of the administrator who has the appropriate permissions to import data into Oracle Role Manager.
6. Click **Browse** to navigate to the newly create `DAR` file, then click **Load**.

The page will display the progress of your data load. You can click **refresh** at any time to refresh the page.

4 Creating and Maintaining System Identities

This chapter includes the following sections:

- About System Identities
- Creating System Identities
- Updating System Identities
- Deleting System Identities
- Restoring the Oracle Role Manager System Administrator

The procedures in this section assume that you have already completed the following steps:

- A database instance has been created for Oracle Role Manager with the appropriate tablespaces.
- The Oracle Role Manager database owner and application user schemas have been created and contain no data.
- The database is accessible.
- The Oracle Role Manager administrative tools are accessible.
- The application server on which Oracle Role Manager is or will be deployed is not running.

Refer to the *Oracle Role Manager Installation Guide* for more information about these assumptions.

4.1 About System Identities

System Identities are system user objects that are created to access the Oracle Role Manager system. System Identities can be used to represent external systems or system administrator logins, for example, the default system administrator. The purpose is to allow entities, not represented as people within Oracle Role Manager to have the ability to login.

Although System Identities can be created or modified as part of a data load process, the command-line administrative tool described in this chapter is what administrators will use to create and manage System Identity objects.

The command-line tool provides the following functions for System Identities:

- Create
- Update
- Delete

As with the other administrative tools provided with Oracle Role Manager, the System Identity management tool must be run at the command line with the appropriate classpath and access to the Oracle Role Manager libraries.

4.2 Creating System Identities

The System Identity Tool creates System Identities and their attributes in the database used by Oracle Role Manager. This database is defined by the combination of the provided database properties (JDBC driver class name and JDBC connection URL) that are identified by the provided username/password.

When creating System Identities, you must provide a file that contains attribute values for the System Identity. The attributes for System Identity creation are the same as those allowed during data load. For information about what attributes are available, refer to the Oracle Role Manager Developer's Guide.

You will have to load data in a .dar file to grant system roles to the created System Identity. See *ORMHome/samples/sample_data/admin_systemrole_privilege_mapping.dar* as an example.

Example 4–1 Creating a System Identity for the PeopleSoft System

```
systemidentity_create appuser peoplesoft peoplesoft.txt
```

This would create the `peoplesoft` System Identity with any attribute values as specified in the `peoplesoft.txt` file, whose contents might resemble:

```
#Attributes for the Peoplesoft system identity
displayName=Peoplesoft Identity
uniqueName=peoplesoft
description=The System Identity that represents the Peoplesoft system for
integration purposes
```

To create a System Identity:

1. Create a text file that contains the required and optional attributes to set for the System Identity. (Refer to the preceding example.)
2. In a command-line window, navigate to the home directory where Oracle Role Manager is installed.
3. Navigate to *ORM_HOME*`/config`, and edit the `db.properties` file to match your environment:

   ```
   db.driverClass=oracle.jdbc.driver.OracleDriver
   db.connection_string=jdbc:oracle:thin:@$HOST$:$PORT$/$SERVICE_NAME$
   ```

   where `$HOST$` is the database host name, `$PORT$` is the database listener port, and `$SERVICE_NAME$` is the database service name on which the Oracle Role Manager users/schemas were created.
4. In a command window, navigate to *ORM_HOME*/bin.
5. Run the following command to create a System Identity:

   ```
   systemidentity_create <ormapp-user> <new-user> <attrfile>
   ```

   where:

   `<ormapp-user>` is the username of the database "application" user/schema for Oracle Role Manager.

<admin-user> is the username to use as the Oracle Role Manager System Administrator.

<attrfile> is the path to the file containing the required attributes for role creation.

6. At the prompt, type the password of the Oracle Role Manager application user.
7. At the prompt, type the password for the Oracle Role Manager system administrator account.

4.3 Updating System Identities

The System Identity Tool can also be used to update passwords and other attributes of System Identities already in the system.

When updating System Identities without attribute updates, the attributes file is not required. If the tool doesn't detect any new information, no updates will occur.

Example 4–2 Updating the System Identity for the PeopleSoft System

```
systemidentity_update appuser peoplesoft newattributes.txt
```

This would update the `peoplesoft` System Identity with new attributes.

To update a System Identity:

1. In a command-line window, navigate to the home directory where Oracle Role Manager is installed.
2. Navigate to *ORM_HOME*/`config`, and edit the `db.properties` file to match your environment:

   ```
   db.driverClass=oracle.jdbc.driver.OracleDriver
   db.connection_string=jdbc:oracle:thin:@$HOST$:$PORT$/$SERVICE_NAME$
   ```

   where `$HOST$` is the database host name, `$PORT$` is the database listener port, and `$SERVICE_NAME$` is the database service name on which the Oracle Role Manager users/schemas were created.
3. In a command window, navigate to *ORM_HOME*/bin.
4. Run the following command to update the System Identity:

   ```
   systemidentity_update <ormapp-user> <admin-user> <attrfile>
   ```

   where:

   <ormapp-user> is the username of the database "application" user/schema for Oracle Role Manager.

   <admin-user> is the username of the System Identity to update.

   <attrfile> is the path to the file containing any changed attributes for the System Identity. This file is optional. If not provided, attributes will not be updated.
5. At the prompt, type the password of the Oracle Role Manager application user.
6. To update the password of the System Identity:
   a. Type `Y` at the prompt.
   b. Type the new password of System Identity.

4.4 Deleting System Identities

The System Identity Tool can also be used to delete System Identities already in the system.

> **Note:** Delete System Identities with caution. Only the Oracle Role Manager System Identities are recoverable. If you mistakenly delete a System Identity, you must create it again and re-grant any roles that had been granted to the original System Identity. For information about recovering the Oracle Role Manager System Identities, refer "Restoring the Oracle Role Manager System Administrator" on page 4-4.

Example 4–3 Deleting the System Identity for the PeopleSoft System

```
systemidentity_delete appuser peoplesoft
```

This would delete the `peoplesoft` System Identity along with any relationships, role grants and entitlements.

To delete a System Identity:

1. In a command-line window, navigate to the home directory where Oracle Role Manager is installed.
2. Navigate to *ORM_HOME*`/config`, and edit the `db.properties` file to match your environment:

   ```
   db.driverClass=oracle.jdbc.driver.OracleDriver
   db.connection_string=jdbc:oracle:thin:@$HOST$:$PORT$/$SERVICE_NAME$
   ```

   where `$HOST$` is the database host name, `$PORT$` is the database listener port, and `$SERVICE_NAME$` is the database service name on which the Oracle Role Manager users/schemas were created.
3. In a command window, navigate to *ORM_HOME*/bin.
4. Run the following command to delete the Oracle Role Manager System Identity:

   ```
   systemidentity_delete <ormapp-user> userID
   ```

   where:

   `<ormapp-user>` is the username the database "application" user/schema for Oracle Role Manager.
5. At the prompt, type the password of the Oracle Role Manager application user.

4.5 Restoring the Oracle Role Manager System Administrator

The RebootstrapTool can be used for recovering from a system where the role grants or entitlement mappings for the system administrator have been corrupted or removed. This tool reinstates the current default configuration stored in the database with the name *oracle.iam.rm.bootstrap/default.xml*.

If you have deployed configuration_hardened.car, then the configuration file, *configuration_hardened.car/config/oracle.iam.rm.bootstrap/default.xml* is stored in Oracle Role Manager database. When the rebootstrap tool is run again, this configuration file (*configuration_hardened.car/config/oracle.iam.rm.bootstrap/default.xml*), is fetched from the database and used, so that it gives the same privileges to the System Administrator.

To restore the Oracle Role Manager System Administrator:

> **Note:** You must stop the server before performing the following steps.

1. In *ORM_HOME*`/config`, update the `db.properties` file that contains the following two lines:

   ```
   db.driverClass=oracle.jdbc.driver.OracleDriver
   db.connection_string=jdbc:oracle:thin:@$HOST$:$PORT$/$SERVICE_NAME$
   ```

   where `$HOST$` is the database host name, `$PORT$` is the database listener port, and `$SERVICE_NAME$` is the database service name on which the Oracle Role Manager users/schemas were created.

2. In a command window, navigate to *ORM_HOME*`/bin`.
3. Run the following command to recover the Oracle Role Manager system administrator:

   ```
   rebootstrap_tool <ormapp-user> <admin-user>
   ```

   where:

   `<ormapp-user>` is the username of the Oracle Role Manager application user/schema.

   `<admin-user>` is the username of the Oracle Role Manager system administrator you want to restore.

4. At the prompt, type the password of the Oracle Role Manager application user.
5. At the prompt, type a password for the system administrator to be restored. This can be the original password or a new password.

4.6 Resetting the Failed Login Count

This feature enables you to reset the user's password in case the user account is locked out. A counter is used to record the number of failed attempts performed for each user's account. If the failed attempts exceeds the configurable limit, then the user account is locked. Perform one of the following procedures to unlock the account:

1. Reset the login attempt counter by performing the following steps:
   a. Log in to Oracle Role Manager Admin Console.
   b. Go to Security and click **Reset User**. The Reset User's Login Failure Count page is displayed. You can use this screen to reset the failed login attempt counter for both users and system identities and is the only way to reset the counter for users.
   c. In the **User Type** field, select the user type, either `person` or `system identity`.
   d. In the **User Name** field, enter the user name whose account has been locked.
   e. Click **Reset Count**. For information about setting the default count, refer Table 2–1, " Authentication Configuration Values".
2. If all System Identities are locked making you unable to use the Oracle Role Manager console, then run the following script to unlock one of the accounts:

```
systemidentity_update <ormapp-user> <admin-user> <attrfile>
```

where:

`<ormapp-user>` is the username of the database application user/schema for Oracle Role Manager.

`<admin-user>` is the username of the System Identity to update.

`<attrfile>` is the path to the file containing any changed attributes for the System Identity. This file is optional and if not provided, then attributes will not be updated.

Note: You must stop the server before performing Step 2.

5 Configuring Oracle Role Manager for Single Sign-On

This chapter describes managing user authentication and authorization by using Oracle Access Manager when a user logs into Oracle Role Manager.

This chapter covers the following topics:

5.1 About the Single Sign-On Configuration with Oracle Role Manager

Oracle Role Manager has two authentication mechanisms:

- Default mode, where Oracle Role Manager manages the credential validation and session maintenance.
- Single sign-on (SSO) mode, where HTTP header variable is used by the SSO system to communicate with Oracle Role Manager.

The header variable should contain the user ID of the Oracle Role Manager user.

The configuration of Oracle Access Manager with Oracle Role Manager provides a secure web-based infrastructure for role management for all customer applications and processes. Oracle Access Manager integrates identity and access management across Oracle Role Manager, enterprise resources, and other domains deployed on eBusiness networks. Oracle Access Manager provides the foundation for managing the identities of customers, partners, and employees across internet applications. These user identities are combined with security policies for protected web interaction.

Note: Oracle certifies Oracle Access Manager for SSO configuration with Oracle Role Manager, but you can use any other SSO providers.

The configuration of Oracle Access Manager with Oracle Role Manager adds the following features to Oracle Role Manager implementations:

- Oracle Access Manager authentication and authorization services for Oracle Role Manager.
- Oracle Access Manager single sign-on for Oracle Role Manager and other Oracle Access Manager-protected resources within a single domain or across multiple domains.
- Oracle Access Manager authentication schemes, which provide a single sign-on for Oracle Role Manager such as, users must enter a user name and password in a window supplied by the web server.
- Session timeout, Oracle Access Manager enables you to set the length of time for a user session to be valid.
- Oracle Access Manager authentication schemes, the following schemes provide single sign-on for Oracle Identity Manager:
  - **Basic**: Users must enter a user name and password in a window supplied by the Web server.

    This method can be redirected to SSL.
  - **Form**: This method is similar to the basic challenge method, but users enter information in the custom HTML form.

    You can choose the information users must provide in the form that you create.
  - **X509 Certificates:** X.509 digital certificates over SSL.

    A user's browser must supply a certificate.
  - **Integrated Windows Authentication (IWA):** Open an Internet Explorer (IE) browser, request a Oracle Access Manager-protected Web resource to complete single sign-on.
  - **Custom**: Additional forms of authentication can be incorporated through use of the Oracle Access Manager Authentication Plug-in API.

5.2 Configuration Design

To achieve the Oracle Access Manager single sign-on with Oracle Role Manager:

- Deploy an HTTP Server in front of the J2EE application server on which the Oracle Role Manager is deployed.
- Deploy the HTTP Server as a reverse proxy.
- Deploy a WebGate on the HTTP Server.

  See Also: *Oracle Access Manager Installation Guide* for more information about setting up a WebGate on an HTTP server.

- Populate a header variable with an attribute value that is stored in the LDAP directory used by Oracle Access Manager.
- Configure Oracle Role Manager to use the single sign-on mode of authentication.

Figure 5–1 shows the configuration design for single sign-on between Oracle Role Manager and Oracle Access Manager.

When configured, Oracle Access Manager enables Single Sign-On between Oracle Role Manager Web UI and another application.

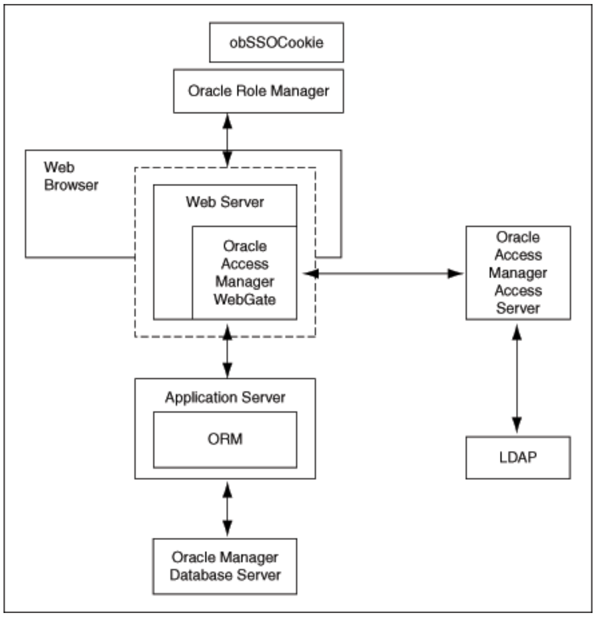
You can access the administrative console with a web browser. The WebGate intercepts the your HTTP request and checks for the presence of an obSSOCookie. If the cookie does not exist or it has expired, an error message is shown asking you to verify the credentials.

On the Oracle Role Manager side, there is a J2EE Servlet Filter, which is configured to intercept requests to the faces servlet. The filter verifies if the user is authenticated, that is if the user has a ClientEntity in the session, and allows the request to proceed if the value is true. If the user is not authenticated, then the filter looks for a particular header, configured by the filter's configuration in the web.xml file, to use as the person identifier. If the header is present, then the header's value is used to create a ClientEntity that the Web UI uses for the rest of the session.

Oracle Access Manager verifies the credentials, and if the user is authenticated, the WebGate redirects the user to the requested resource and passes the required header variable to Oracle Identity Manager. Oracle Identity Manager, which has been configured to read a HTTP Header variable instead of its authentication, reads the HTTP Header and uses the value stored in the variable as the logged in user.

Figure 5–1 shows the configuration design of Oracle Role Manager for single sign-on.

Figure 5–1 Configuration Design of Oracle Role Manager for Single Sign-On


obSSOCookie
Oracle Role Manager
Web Browser
Web Server
Oracle Access Manager WebGate
Oracle Access Manager Access Server
Application Server
ORM
LDAP
Oracle Manager Database Server

This figure shows the configuration design of Oracle Role Manager for single sign-on. The description of the design is provided in the same section.

****

The following steps demonstrate the single sign-on process:

1. A user attempts to access Oracle Role Manager Web UI.
2. A WebGate that is deployed on the HTTP server intercepts the request.
3. The WebGate checks the Access Server to determine if the resource (the Oracle Role Manager URL) is protected.

   The security policy in the Access System contains an authentication scheme, authorization rules, and allowed operations based on authentication and authorization success or failure.

4. If a valid session does not exist, and the resource is protected, WebGate prompts the user for credentials.
5. If the credentials are validated, Oracle Access Manager performs the actions that are defined in the security policy for the resource and sets an HTTP header variable that maps to the Oracle Role Manager user ID.
6. If a valid session cookie exists, and if the user is authorized to access the resource, WebGate redirects the user to the requested Oracle Role Manager resource.
7. The administrative console reads the HTTP header variable and sets the value as the logged-in user.
8. The administrative console generates the applications pages, pending any further authorization checks performed in Oracle Role Manager.

5.2.1 Preparing Your Environment

To prepare your environment for the integration, perform the following steps:

1. Install a supported directory server according to vendor instructions, for example, iPlanet.
2. Install and configure Oracle Access Manager using the directory server as the LDAP repository.
3. Ensure that the Oracle Role Manager J2EE application server is proxied by an HTTP server (Apache 2.0).
4. Configure the Web browser (Internet Explorer) to allow cookies, according to vendor instructions.
5. Ensure that user IDs in Oracle Role Manager and Oracle Access Manager are same.

5.2.2 Setting Up Oracle Role Manager for Single Sign-On

To configure Oracle Role Manager for single sign-on with Oracle Access Manager, perform the following procedure:

1. Extract webui.ear and locate the file web.xml. The file is present in the WEB-INF directory.
2. Open the web.xml file in a text editor.
3. Locate the following section:

```
<filter>
<filter-name>SSO Filter</filter-name>
<filter-class>oracle.iam.rm.ui.webapp.SSOInterceptor</filter-class>
<init-param>
<param-name>httpHeader</param-name>
<param-value>username</param-value>
</init-param>
<init-param>
<param-name>alternativeWelcome</param-name>
<param-value>/pages/inbox/find_outbox.jsf</param-value>
</init-param>
</filter>
```

4. Replace the value username with a name such as ORM_UID.

> **Note:** The name can be any value, but the same name is to be used for header variable while creating access policy in OAM Access System.

5. Save and close the file.
6. Disable the logout link by opening the `header.xhtml` file present in the pages/components folder and add an attribute `rendered="false"` to the `commandLink` tag with an attribute id="`logout`".

   You can achieve this by replacing the tag:

   ```
   <h:commandLink id="logout" value="#{b:text('button.signout')}"
   action="#{ClientSession.gotoSignoutAction}"/>
   ```

   with

   ```
   <h:commandLink id="logout" value="#{b:text('button.signout')}"
   action="#{ClientSession.gotoSignoutAction}" rendered="false"/>
   ```

7. Re-create the file webui.war.
8. Deploy the WAR file, webui.war to the application server.

5.3 Configuring Apache As a Proxy for JBoss

Oracle Role Manager runs on a J2EE application server, for example, JBoss, WebLogic, and WebSphere. You cannot install an AccessGate directly in front of these application servers. You can deploy a Web server, for example, Apache, in front of these application servers. You can deploy the AccessGate on the web server, and configure the web server to route requests to the Oracle Role Manager Application and forward responses back to the user.

For application servers such as JBoss, you must deploy an additional plug-in, referred to as the mod_jk plug-in or the JBoss plug-in, on the Web server.

To configure the Apache HTTP server as a proxy for JBoss:

1. Download and install Apache HTTP Server 2.0.63.
2. Download the latest stable version of mod_jk 1.2.26 binary that supports the installed Apache HTTP Server, from the following URL:

   http://www.apache.org/dist/jakarta/tomcat-connectors/jk/binaries/

3. Rename it to mod_jk.so.
4. Copy this file to the following directory:

   Apache_install_dir/modules

5. Modify Apache_install_dir /conf/httpd.conf and add a single line at the end of the file:

   # Include mod_jk's specific configuration file

   include conf/mod-jk.conf

6. Create the following text files in the directory Apache_install_dir\conf:
   - mod-jk.conf
   - workers.properties

- uriworkermap.properties

  Oracle recommends that you do not rename uriworkermap.properties and workers.properties. If you do, your configuration may stop working. The locations of these files are defined under two registry keys: worker_file and worker_mount_file. These files are in HKEY_LOCAL_MACHINE\SOFTWARE\Apache Software Foundation\Jakarta Isapi Redirector\version_number.

7. Copy the following configuration into the mod-jk.conf file:

```
# Load mod_jk module
# Specify the filename of the mod_jk lib
LoadModule jk_module modules/mod_jk.so

# Where to find workers.properties
JkWorkersFile conf/workers.properties

# Where to put jk logs
JkLogFile logs/mod_jk.log

# Set the jk log level [debug/error/info]
JkLogLevel info

# Select the log format
JkLogStampFormat  "[%a %b %d %H:%M:%S %Y]"

# JkOptions indicates to send SSK KEY SIZE
JkOptions +ForwardKeySize +ForwardURICompat -ForwardDirectories

# JkRequestLogFormat
JkRequestLogFormat "%w %V %T"

# Mount your applications
JkMount /application/* loadbalancer

# You can use external file for mount points.
# It will be checked for updates each 60 seconds.
# The format of the file is: /url=worker
# /examples/*=loadbalancer
JkMountFile conf/uriworkermap.properties

# Add shared memory.
# This directive is present with 1.2.10 and
# later versions of mod_jk, and is needed for
# for load balancing to work properly
JkShmFile logs/jk.shm

# Add jkstatus for managing runtime data
<Location /jkstatus/>
    JkMount status
    Order deny,allow
    Deny from all
    Allow from 127.0.0.1
</Location>
```

8. Copy the following into the workers.properties file:

> **Note:** The local values <Host IP or DNS Name> must be substituted in the following command.

```
# Define list of workers that will be used
# for mapping requests
worker.list=loadbalancer,status

# Define Node1
# modify the host as your host IP or DNS name.
worker.node1.port=8009
worker.node1.host=<host IP or DNS Name>
worker.node1.type=ajp13
worker.node1.lbfactor=1
worker.node1.cachesize=10

# Load-balancing behaviour
worker.loadbalancer.type=lb
worker.loadbalancer.balance_workers=node1
worker.loadbalancer.sticky_session=1
#worker.list=loadbalancer

# Status worker for managing load balancer
worker.status.type=status
```

9. Copy the following into the uriworkermap.properties file:

```
# Simple worker configuration file

# Mount the Servlet context to the ajp13 worker
/jmx-console=loadbalancer
/jmx-console/*=loadbalancer
/web-console=loadbalancer
/web-console/*=loadbalancer
/webui=loadbalancer
/webui/*=loadbalancer
/ormconsole=loadbalancer
/ormconsole/*=loadbalancer
```

10. Edit *JBOSS_HOME*/server/all/deploy/jbossweb-tomcat50.sar/server.xml (replace /all with your own server name) and locate the <Engine….> element and add an attribute jvmRoute:

```
<Engine name="jboss.web" defaultHost="localhost" vmRoute="node1">
</Engine>
```

11. Edit *JBOSS_HOME*/server/all/deploy/jbossweb-tomcat50.sar/META-INF/jboss-service.xml (replace /all with your own server name) and locate the <attribute> element with a name of UseJK and set its value to "true":

```
<attribute name="UseJK">true</attribute>
```

12. Start the Apache server and test the installation.

5.4 Configuring Apache As a Proxy for WebLogic

To configure the Apache HTTP server as a proxy for WebLogic:

1. Download and install Apache HTTP Server 2.0.63.
2. Download the apache plugin(s) for WebLogic 10.3 from the following location: http://download.oracle.com/otn/bea/weblogic/server103/server103_apacheplugins.zip
3. Copy the appropriate `mod_wl_20.so` from `server103_apacheplugins.zip` into `Apache_install_dir/modules`.

   Note: Appropriate mod_wl_20.so means, if WebLogic is installed on Win 32 machine, then you must copy \win\32\mod_wl_20.so from `server103_apacheplugins.zip`.

4. Modify `Apache_install_dir /conf/httpd.conf` to add the following at the end of the file:

   ```
   LoadModule weblogic_module modules/mod_wl_20.so
   <IfModule mod_weblogic.c>
   WebLogicHost <hostname>
   WebLogicPort <port>
   </IfModule>
   <LocationMatch ^/webui>
   SetHandler weblogic-handler
   </LocationMatch>
   ```

   Note: Replace <hostname> and <port> for the appropriate values from the WebLogic Installation.

5. In the WebLogic domain configuration, add the following element:

   ```
   <enforce-valid-basic-auth-credentials>false</enforce-valid-basic-auth-credenti
   als>
   ```

   to the last line of `<security-configuration>` in config.xml for the users domain, usually in *DOMAIN_NAME*/config/config.xml. This keeps WebLogic from trying to authenticate basic authentication headers.

5.5 Configuring Apache as a Proxy for WebSphere

To configure the Apache HTTP server as a proxy for WebSphere:

1. Download and install Apache HTTP Server 2.0.63 or Apache HTTP Server 2.2.11.
2. Copy mod_ws_20.so from websphere_install\server\plugin\win32 into modules in Apache_install_dir/modules.
3. Modify Apache_install_dir /conf/httpd.conf to add the following at the end of the file:

   ```
   LoadModule proxy_module modules/mod_proxy.so
   LoadModule proxy_http_module modules/mod_proxy_http.so
   LoadModule rewrite_module modules/mod_rewrite.so

   ProxyRequests Off
   ```

```
<Proxy>
   Order deny,allow
   Allow from all
</Proxy>
```

RewriteEngine on

```
ProxyPass /webui/ http://localhost:9080/webui/
ProxyPassReverse /webui/ http://localhost:9080/webui/
RewriteRule ^/webui$ /webui/ [R]
```

5.6 Setting Up a WebGate on an HTTP Server

To set up a WebGate on an HTTP server:

1. Install and configure Oracle Access Manager on a supported platform, using a supported LDAP server.
2. Create an AccessGate and install it on the Apache server.

   The following is the sample configuration for an access gate:

   AccessGate Name: AccessGate_Apache

   State: Enabled

   Hostname: <hostname where Apache is installed>

   Port: 80

   AccessGate Password: abcd1234

   Access Management Service: On

   Primary HTTP Cookie Domain: idc.oracle.com

   Preferred HTTP Host: <hostname where Apache is installed>
3. Associate the Access Server.

   > **See Also:** *Oracle Access Manager Installation Guide* for more information about setting up a WebGate on an HTTP server.

5.7 Setting Up Oracle Access Manager for Single Sign-On With Oracle Role Manager

To configure Oracle Access Manager for single sign-on with Oracle Role Manager, perform the following procedure:

1. In the landing page for the Access System, click **Policy Manager** and then click **Create Policy Domain**.
2. Create a policy domain and policies to restrict access to the Oracle Role Manager URLs.
3. In the Access System Console, define host identifiers for Oracle Role Manager.
4. Go to Policy Manager, Oracle Role Manager policy domain, Resources tab, and define resources for Oracle Access Manager to protect. Table 5–1 shows the resource definition for Oracle Access Manager.

Table 5–1 Resource Definition

Resource	Resource Type	URL Prefix	Description
All	http	/webui	webui

5. Click the **Authorization Rules** tab and define an authorization rule to determine which authenticated users can access the Oracle Role Manager URLs. Table 5–2 shows the authorization rules for the users who access Oracle Role Manager.

Table 5–2 Authorization Rules

Authorization Rules			
Name	authz		
Description	authz		
Enabled	Yes		
Allow takes precedence	No		
On Success	**Type**	**Name**	**Return Attribute**
HTTP Header Variable	headervar	ORM_UID	uid
HTTP Header Variable			
Allow Access			
Role	Any one		

6. Click the **Default Rules** tab. The Authentication Rule subtab is selected. Perform the following steps:

   a. Define an authentication rule, for example, `Basic Over LDAP`.

   b. Click the **Actions** subtab and define an authorization action that sets a custom HTTP header variable upon successful authorization.

      The header variable must contain a value that maps to the Oracle Role Manager user ID. Table 5–3 shows the authorization expression for the custom HTTP header variable.

Table 5–3 Authorization Expression for Custom HTTP Header Variable

Default Rules			
Authentication Rule			
	authn		
Authentication Scheme	Basic Over LDAP		
On Success			
HTTP Header Variable	**Type**	**Name**	**Return Attribute**
	headervar	ORM_UID	uid
HTTP Header Variable			
Authorization Expression			
Expression	*authz*		

Table 5–3 (Cont.) Authorization Expression for Custom HTTP Header Variable

Default Rules				
	Duplicate Actions	No policy defined for this Authorization Expression. The Access System level default policy for dealing with duplicate action headers will be employed.		
	On Success			
	HTTP Header Variable	**Type**	**Name**	**Return Attribute**
		headervar	ORM_UID	uid
	HTTP Header Variable			
	Audit Rule			
	There is no Audit Rule defined			

7. Click the **Policies** tab, and then click **Add**. Define an access policy in the Oracle role Manager policy domain and add the Oracle Role Manager URL resources to this policy. Table 5–4 shows the access policy to add the Oracle Role Manager URL resources to it.

Table 5–4 Access Policy

Policy				
	Name	webui		
	Description	webui		
	Resource Type	http		
	Resource Operation(s)	GET POST PUT HEAD OPTIONS CONNECT		
	Resource	*all*		
	Authentication Rule			
		authn		
	Authentication Scheme	Basic Over LDAP		
	On Success			
	HTTP Header Variable	**Type**	**Name**	**Return Attribute**
		HeaderVar	ORM_UID	uid
	HTTP Header Variable			
	Authorization Expression			
	Expression	*authz*		

Table 5–4 (Cont.) Access Policy

Policy				
	Duplicate Actions	No policy defined for this Authorization Expression. The Access System level default policy for dealing with duplicate action headers will be employed.		
	On Success			
	HTTP Header Variable	**Type**	**Name**	**Return Attribute**
		headervar	ORM_UID	uid
	HTTP Header Variable			
	Audit Rule			
	There is no Audit Rule defined			

See Also: Oracle Access Manager Installation Guide for more information.

Index