

Oracle® Fusion Middleware

Application Security Guide

11g Release 1 (11.1.1)

E10043-10

November 2011

Oracle Fusion Middleware Application Security Guide, 11g Release 1 (11.1.1)

E10043-10

Copyright © 2003, 2011, Oracle and/or its affiliates. All rights reserved.

Primary Author: Carlos Subi

Contributing Author: Vinaye Misra, Gail Flanegin

Contributor: Amit Agarwal, Soumya Aithal, Moushmi Banerjee, Andre Correa, Marc Chanliau, Pratik Datta, Jordan Douglas, Guru Dutt, Todd Elwood, Vineet Garg, Vikas Ghorpade, Sandeep Guggilam, Shiang-Jia Huang, Dan Hynes, Michael Khalandovsky, Supriya Kalyanasundaram, Lakshmi Kethana, Ganesh Kirti, Ashish Kolli, Rohit Koul, Nithya Muralidharan, Frank Nimphius, Craig Perez, Sudip Regmi, Bhupindra Singh, Kk Sriramadhesikan, Mamta Suri, Kavita Tippana, Srikant Tirumalai, Ramana Turlapati, Jane Xu, Sam Zhou.

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

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

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

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

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

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

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

Contents

Preface	xxxi
Audience	xxxi
Documentation Accessibility	xxxi
Related Documentation	xxxii
Conventions	xxxii
What's New in This Guide	xxxiii
New Features in Release 11gR1 PS5	xxxiii
New Features in Oracle Identity Management 11gR1 PS1	xxxiii
New Features in Release 11gR1 PS3	xxxiv
New Features in Oracle Identity Management 11gR1	xxxiv
New Features in Release 11gR1 PS2	xxxiv
New Features in Release 11gR1 PS1	xxxv
New Features in Release 11gR1	xxxv
Desupported Features from 10.1.3.x	xxxv
Links to Upgrade Documentation	xxxvi
Part I Understanding Security Concepts	
1 Introduction to Oracle Platform Security Services	
What is Oracle Platform Security Services?	1-1
OPSS Main Features	1-2
Supported Server Platforms	1-2
OPSS Architecture Overview	1-3
Benefits of Using OPSS	1-3
Oracle ADF Security Overview	1-4
OPSS for Administrators	1-5
OPSS for Developers	1-5
Scenario 1: Enhancing Security in a Java EE Application	1-6
Scenario 2: Securing an Oracle ADF Application	1-6
Scenario 3: Securing a Java SE Application	1-7
2 Understanding Users and Roles	
Terminology	2-1

Role Mapping	2-4
Permission Inheritance and the Role Hierarchy.....	2-4
The Authenticated Role	2-7
The Anonymous User and Role	2-7
Anonymous Support and Subject.....	2-8
Administrative Users and Roles	2-8
Managing User Accounts	2-9
Principal Name Comparison Logic	2-9
How Does Principal Comparison Affect Authorization?	2-9
System Parameters Controlling Principal Name Comparison.....	2-10
The Role Category	2-11

3 Understanding Identities, Policies, Credentials, Keys, and Certificates

Authentication Basics	3-1
Supported LDAP Identity Store Types	3-2
Oracle WebLogic Authenticators.....	3-2
Using an LDAP Authenticator.....	3-3
Configuring the LDAP Identity Store Service	3-3
Additional Authentication Methods.....	3-4
WebSphere Identity Stores	3-4
Policy Store Basics	3-4
Credential Store Basics	3-5
Keystore Service Basics	3-6
Keystore Repository Types.....	3-6
Keystore Repository Scope and Reassociation	3-6

4 About Oracle Platform Security Services Scenarios

Supported LDAP-, DB-, and File-Based Services	4-1
Management Tools	4-2
Packaging Requirements	4-4
Example Scenarios.....	4-4
Other Scenarios.....	4-6

Part II Basic OPSS Administration

5 Security Administration

Choosing the Administration Tool According to Technology.....	5-1
Basic Security Administration Tasks.....	5-2
Setting Up a Brand New Production Environment	5-3
Typical Security Practices with Fusion Middleware Control.....	5-4
Typical Security Practices with the Administration Console.....	5-4
Typical Security Practices with Oracle Entitlements Server.....	5-5
Typical Security Practices with OPSS Scripts.....	5-5

6 Deploying Secure Applications

Overview.....	6-2
---------------	-----

Selecting the Tool for Deployment	6-2
Deploying Java EE and Oracle ADF Applications with Fusion Middleware Control.....	6-3
Deploying Oracle ADF Applications to a Test Environment	6-5
Deploying to a Test Environment.....	6-6
Typical Administrative Tasks after Deployment in a Test Environment.....	6-7
Deploying Standard Java EE Applications	6-7
Migrating from a Test to a Production Environment	6-8
Migrating Providers other than Policy and Credential Providers.....	6-9
Migrating Identities Manually	6-9
Migrating Policies and Credentials at Deployment.....	6-10
Migrating Policies Manually	6-11
Migrating Credentials Manually	6-15
Migrating Large Volume Policy and Credential Stores	6-18
Migrating Audit Policies	6-19
Migrating Keystore Service Keys and Certificates	6-19

Part III Advanced OPSS Administration

7 Configuring the Identity Store Service

Introduction to the Identity Store Service	7-1
About the Identity Store Service	7-1
Service Architecture.....	7-1
Application Server Support.....	7-2
Java SE Support	7-2
Configuring the Identity Store Provider	7-2
Configuring the Identity Store Service	7-3
What is Configured?.....	7-3
Configuring Multi-LDAP Lookup.....	7-3
Global/Connection Parameters	7-3
Back-End/Connection Parameters.....	7-4
Configuration in WebLogic Server	7-4
Configuring the Service for Single LDAP	7-5
Configuring the Service for Multiple LDAP using Fusion Middleware Control.....	7-5
Configuring the Service for Multiple LDAP using WLST	7-5
Configuring Other Parameters	7-6
Restarting Servers	7-6
Examples of the Configuration File.....	7-6
Configuring Split Profiles	7-7
Configuring Custom Authenticators.....	7-7
Configuration in Other Application Servers.....	7-8
Configuring the Service for Single LDAP	7-8
Configuring the Service for Multiple LDAP	7-8
Java SE Environments	7-11
Querying the Identity Store Programmatically	7-11
SSL for the Identity Store Service	7-12
Connections from Oracle WebLogic Server to Identity Store.....	7-12

One-way SSL in a Multi-LDAP Scenario	7-12
Two-way SSL in a Multi-LDAP Scenario	7-13
Connections in a Single-LDAP Scenario	7-13

8 Configuring the OPSS Security Store

Introduction to the OPSS Security Store	8-1
Using an LDAP-Based OPSS Security Store	8-2
Multiple-Node Server Environments	8-3
Prerequisites to Using an LDAP-Based Security Store	8-3
Setting Up a One- Way SSL Connection to the LDAP	8-4
Using a DB-Based OPSS Security Store	8-6
Prerequisites to Using a DB-Based Security Store	8-6
Creating the OPSS Schema in an Oracle Database	8-6
Dropping the OPSS Schema in an Oracle Database	8-7
Creating a Data Source Instance	8-7
Maintaining a DB-Based Security Store	8-8
Setting Up an SSL Connection to the DB	8-9
Configuring the OPSS Security Store	8-9
Reassociating the OPSS Security Store	8-10
Reassociating with Fusion Middleware Control	8-10
Securing Access to Oracle Internet Directory Nodes	8-14
Reassociating with the Script reassociateSecurityStore	8-14
Migrating the OPSS Security Store	8-15
Migrating with Fusion Middleware Control	8-15
Migrating with the Script migrateSecurityStore	8-16
Examples of Use	8-18
Configuring the Identity Provider, Property Sets, and SSO	8-18
Configuring the Identity Store Provider	8-18
Configuring Properties and Property Sets	8-18
Specifying a Single Sign-On Solution	8-19
The OPSS SSO Framework	8-19
Configuring an SSO Solution with Fusion Middleware Control	8-20
OAM Configuration Example	8-20

9 Managing the Policy Store

Managing the Policy Store	9-1
Managing Policies with Fusion Middleware Control	9-2
Managing Application Policies	9-3
Managing Application Roles	9-4
Managing System Policies	9-6
Managing Application Policies with OPSS Scripts	9-7
listAppStripes	9-9
createAppRole	9-10
deleteAppRole	9-10
grantAppRole	9-11
revokeAppRole	9-11
listAppRoles	9-12

listAppRolesMembers	9-12
grantPermission.....	9-13
revokePermission	9-14
listPermissions	9-15
deleteAppPolicies.....	9-15
createResourceType	9-16
getResourceType	9-17
deleteResourceType.....	9-17
createResource	9-18
deleteResource	9-18
listResources	9-19
listResourceActions.....	9-19
createEntitlement	9-20
getEntitlement.....	9-21
deleteEntitlement	9-21
addResourceToEntitlement	9-22
revokeResourceFromEntitlement	9-22
listEntitlements.....	9-23
grantEntitlement.....	9-23
revokeEntitlement.....	9-24
listEntitlement.....	9-25
listResourceTypes	9-25
reassociateSecurityStore.....	9-26
Running an Offline Script after Reassociating to a DB-Based Store	9-27
Caching and Refreshing the Cache.....	9-28
An Example.....	9-29
Granting Policies to Anonymous and Authenticated Roles with WLST Scripts	9-30
Application Stripe for Versioned Applications in WLST Scripts.....	9-30
Managing Application Policies with Oracle Entitlements Server.....	9-31
Guidelines to Configure the Policy Store.....	9-31

10 Managing the Credential Store

Credential Types.....	10-1
Encrypting Credentials.....	10-1
Managing the Credential Store.....	10-3
Managing Credentials with Fusion Middleware Control.....	10-3
Managing Credentials with OPSS Scripts	10-6
listCred.....	10-7
updateCred	10-7
createCred	10-8
deleteCred	10-8
modifyBootStrapCredential.....	10-9
addBootStrapCredential.....	10-9
exportEncryptionKey	10-10
importEncryptionKey.....	10-10
restoreEncryptionKey.....	10-10

11 Managing Keys and Certificates with the Keystore Service

About the Keystore Service	11-1
Structure of the Keystore Service	11-1
Types of Keystores	11-2
Domain Trust Store	11-2
About Keystore Service Commands	11-3
Getting Help for Keystore Service Commands	11-3
Keystore Service Command Reference	11-3
changeKeyPassword	11-4
changeKeyStorePassword	11-5
createKeyStore	11-5
deleteKeyStore	11-6
deleteKeyStoreEntry	11-6
exportKeyStore	11-7
exportKeyStoreCertificate	11-7
exportKeyStoreCertificateRequest	11-8
generateKeyPair	11-8
generateSecretKey	11-9
getKeyStoreCertificates	11-9
getKeyStoreSecretKeyProperties	11-10
importKeyStore	11-10
importKeyStoreCertificate	11-11
listExpiringCertificates	11-12
listKeyStoreAliases	11-12
listKeyStores	11-13

12 Introduction to Oracle Fusion Middleware Audit Framework

Benefits and Features of the Oracle Fusion Middleware Audit Framework	12-1
Objectives of Auditing	12-1
Today's Audit Challenges	12-2
Oracle Fusion Middleware Audit Framework in 11g	12-2
Overview of Audit Features	12-3
Oracle Fusion Middleware Audit Framework Concepts	12-4
Audit Architecture	12-4
Key Technical Concepts	12-7
Audit Metadata Storage	12-8
Audit Data Storage	12-8
Analytics	12-9

13 Configuring and Managing Auditing

Audit Administration Tasks	13-1
Managing the Audit Data Store	13-2
Create the Audit Schema using RCU	13-2
Set Up Audit Data Sources	13-3
Multiple Data Sources	13-4
Configure a Database Audit Data Store for Java Components	13-4

View Audit Data Store Configuration	13-4
Configure the Audit Data Store	13-5
Deconfigure the Audit Data Store	13-6
Configure a Database Audit Data Store for System Components	13-6
Deconfigure the Audit Data Store	13-8
Tuning the Bus-stop Files.....	13-8
Configuring the Stand-alone Audit Loader	13-9
Configuring the Environment.....	13-10
Running the Stand-Alone Audit Loader	13-10
Managing Audit Policies	13-11
Manage Audit Policies for Java Components with Fusion Middleware Control.....	13-11
Manage Audit Policies for System Components with Fusion Middleware Control	13-14
Manage Audit Policies with WLST	13-17
View Audit Policies with WLST	13-17
Update Audit Policies with WLST	13-17
Example 1: Configuring an Audit Policy for Users with WLST	13-18
Example 2: Configuring an Audit Policy for Events with WLST	13-18
Custom Configuration is Retained when the Audit Level Changes.....	13-19
Manage Audit Policies Manually	13-19
Location of Configuration Files for Java Components.....	13-19
Audit Service Configuration Properties in jps-config.xml for Java Components	13-20
Switching from Database to File for Java Components.....	13-20
Manually Configuring Audit for System Components.....	13-20
Audit Logs.....	13-21
Location of Audit Logs.....	13-22
Audit Log Timestamps.....	13-22
Advanced Management of Database Store	13-22
Schema Overview	13-22
Table Attributes	13-23
Indexing Scheme	13-24
Backup and Recovery	13-24
Importing and Exporting Data.....	13-25
Partitioning.....	13-25
Partition Tables.....	13-25
Backup and Recovery of Partitioned Tables	13-27
Import, Export, and Data Purge	13-27
Tiered Archival.....	13-27
Metadata Schema Overview.....	13-28

14 Using Audit Analysis and Reporting

Setting up Oracle Business Intelligence Publisher for Audit Reports.....	14-1
About Oracle Business Intelligence Publisher	14-1
Install Oracle Business Intelligence Publisher	14-3
Set Up Oracle Reports in Oracle Business Intelligence Publisher.....	14-3
Set Up Audit Report Templates	14-4
Set Up Audit Report Filters	14-4
Configure Scheduler in Oracle Business Intelligence Publisher	14-5

Organization of Audit Reports	14-6
View Audit Reports	14-7
Example of Oracle Business Intelligence Publisher Reports	14-8
Audit Report Details	14-10
List of Audit Reports in Oracle Business Intelligence Publisher.....	14-11
Attributes of Audit Reports in Oracle Business Intelligence Publisher	14-13
Customizing Audit Reports	14-14
Using Advanced Filters on Pre-built Reports	14-14
Creating Custom Reports.....	14-15

Part IV Single Sign-On Configuration

15 Introduction to Single Sign-On in Oracle Fusion Middleware

Choosing the Right SSO Solution for Your Deployment	15-1
Introduction: OAM Authentication Provider for WebLogic Server	15-4
About Using the Identity Asserter Function with Oracle Access Manager	15-6
About Using the Authenticator Function with Oracle Access Manager	15-9
Choosing Applications for Oracle Access Manager SSO Scenarios and Solutions	15-10
Applications Using Oracle Access Manager for the First Time	15-10
Applications Migrating from Oracle Application Server to Oracle WebLogic Server	15-11
Applications Using OAM Security Provider for WebLogic SSPI	15-11
Implementation: Using the Provider with OAM 11g versus OAM 10g	15-12
Requirements for the Provider with Oracle Access Manager	15-13
Setting Up Debugging in the WebLogic Administration Console	15-14

16 Configuring Single Sign-On with Oracle Access Manager 11g

Introduction to Oracle Access Manager 11g SSO	16-1
Previewing Pre-Seeded OAM 11g Policies for Use by the 10g AccessGate	16-4
Deploying the Oracle Access Manager 11g SSO Solution	16-7
Installing the Authentication Provider with Oracle Access Manager 11g.....	16-8
Converting Oracle Access Manager Certificates to Java Keystore Format.....	16-9
Session Token: Provisioning an OAM Agent with Oracle Access Manager 11g.....	16-12
About WebGate Provisioning Methods for Oracle Access Manager 11g.....	16-12
Provisioning a WebGate with Oracle Access Manager 11g.....	16-13
Configuring Identity Assertion for SSO with Oracle Access Manager 11g.....	16-16
Establishing Trust with Oracle WebLogic Server	16-16
Configuring Providers in the WebLogic Domain	16-19
Trusted Header Assertion: Configuring Digital Signature Verification.....	16-25
Trusted Header Assertion: Configuring Policies	16-28
Testing Oracle Access Manager Identity Assertion for Single Sign-on	16-29
Configuring the Authenticator Function for Oracle Access Manager 11g	16-29
Configuring Providers for the Authenticator in a WebLogic Domain	16-30
Configuring the Application Authentication Method for the Authenticator	16-33
Mapping the Authenticated User to a Group in LDAP	16-34
Testing the Oracle Access Manager Authenticator Implementation	16-34
Configuring Identity Assertion for Oracle Web Services Manager and OAM 11g.....	16-35

Configuring Providers in a WebLogic Domain for Oracle Web Services Manager.....	16-36
Testing the Identity Asserter with Oracle Web Services Manager	16-38
Configuring Centralized Log Out for Oracle Access Manager 11g	16-38
Logout for 11g WebGate and OAM 11g	16-39
Logout for 10g WebGate with Oracle Access Manager 11g	16-39
Synchronizing the User and SSO Sessions: SSO Synchronization Filter	16-40
Troubleshooting Tips	16-42

17 Configuring Single Sign-On Using Oracle Access Manager 10g

Deploying SSO Solutions with Oracle Access Manager 10g	17-1
Installing and Setting Up Authentication Providers for OAM 10g	17-1
About Oracle Access Manager 10g Installation and Setup	17-2
Installing Components and Files for Authentication Providers and OAM 10g	17-4
Converting Oracle Access Manager Certificates to Java Keystore Format	17-6
Creating Resource Types in Oracle Access Manager 10g	17-9
Configuring Global Logout for Oracle Access Manager 10g and 10g WebGates.....	17-10
Recommended Process for Configuring Logout	17-10
Alternative Process for Configuring Logout.....	17-13
Oracle Access Manager Authentication Provider Parameter List	17-14
Introduction to OAMCfgTool	17-15
OAMCfgTool Process Overview	17-17
OAMCfgTool Parameters and Values	17-17
Create Mode Parameters and Values	17-18
Validate Mode Parameters and Values	17-27
Delete Mode Parameters and Values	17-28
Sample Policy Domain and AccessGate Profile Created with OAMCfgTool	17-29
Known Issues: JAR Files and OAMCfgTool	17-34
Configuring OAM Identity Assertion for SSO with Oracle Access Manager 10g	17-35
Establishing Trust with Oracle WebLogic Server.....	17-36
Setting Up the Application Authentication Method for SSO	17-36
Confirming mod_weblogic for Oracle Access Manager Identity Asserter	17-37
Establishing Trust between Oracle WebLogic Server and Other Entities	17-37
Configuring the Authentication Scheme for the Identity Asserter.....	17-39
Creating an Authentication Scheme, Policy Domain, and a WebGate Profile	17-39
Configuring Providers in the WebLogic Domain	17-41
About Oracle WebLogic Server Authentication and Identity Assertion Providers.....	17-42
About the Oracle WebLogic Scripting Tool (WLST).....	17-43
Setting Up Providers for Oracle Access Manager Identity Assertion.....	17-45
Setting Up the Login Form for the Identity Asserter and OAM 10g.....	17-47
Testing Identity Assertion for SSO with OAM 10g.....	17-48
Configuring the Authenticator for Oracle Access Manager 10g	17-49
Creating an Authentication Scheme for the Authenticator	17-50
Configuring a Policy Domain for the Oracle Access Manager Authenticator	17-50
About Creating a Policy Domain.....	17-50
Creating a Policy Domain and Access Policies for the Authenticator	17-51
Configuring Providers for the Authenticator in a WebLogic Domain.....	17-55
Configuring the Application Authentication Method for the Authenticator	17-58

Mapping the Authenticated User to a Group in LDAP.....	17-59
Testing the Oracle Access Manager Authenticator Implementation.....	17-59
Configuring Identity Assertion for Oracle Web Services Manager and OAM 10g.....	17-60
Creating an Policy Domain for Use with Oracle Web Services Manager	17-61
Configuring Providers in a WebLogic Domain for Oracle Web Services Manager.....	17-63
Testing the Identity Asserter with Oracle Web Services Manager	17-66
Synchronizing the User and SSO Sessions: SSO Synchronization Filter	17-66
Troubleshooting Tips for OAM Provider Deployments	17-68
About Using IPv6.....	17-69
Apache Bridge Failure: Timed Out	17-69
Authenticated User with Access Denied	17-70
Browser Back Button Results in Error.....	17-70
Cannot Reboot After Adding OAM and OID Authenticators	17-70
Client in Cluster with Load-Balanced WebGates.....	17-70
Error 401: Unable to Access the Application.....	17-72
Error 403: Unable to Access the Application.....	17-73
Error 404: Not Found ... Anything Matching the Request URI	17-73
Error Issued with the Action URL in Form Login Page.....	17-73
Error or Failure on Oracle WebLogic Server Startup.....	17-74
JAAS Control Flag.....	17-74
Login Form is Shown Repeatedly Upon Credential Submission: No Error.....	17-74
Logout and Session Time Out Issues	17-74
Not Found: The requested URL or Resource Was Not Found.....	17-75
Oracle WebLogic Server Fails to Start.....	17-75
Oracle ADF Integration and Cert Mode	17-76
About Protected_JSessionId_Policy	17-77

18 Configuring Single Sign-On using OracleAS SSO 10g

Deploying the OracleAS 10g Single Sign-On (OSSO) Solution	18-1
Using the OSSO Identity Asserter	18-1
Oracle WebLogic Security Framework.....	18-2
OSSO Identity Asserter Processing	18-2
Consumption of Headers with OSSO Identity Asserter	18-4
New Users of the OSSO Identity Asserter.....	18-4
Configuring mod_weblogic.....	18-6
Registering Oracle HTTP Server mod_osso with OSSO Server 10.1.4	18-7
Configuring mod_osso to Protect Web Resources.....	18-8
Adding Providers to a WebLogic Domain for OSSO	18-12
Establishing Trust Between Oracle WebLogic Server and Other Entities.....	18-14
Configuring the Application for the OSSO Identity Asserter	18-15
Synchronizing the User and SSO Sessions: SSO Synchronization Filter	18-16
Troubleshooting for an OSSO Identity Asserter Deployment.....	18-18
SSO-Related Problems.....	18-18
OSSO Identity Asserter-Related Problems.....	18-24
URL Rewriting and JSESSIONID.....	18-25
About mod_osso, OSSO Cookies, and Directives	18-25
New OssoHTTPOnly Directive in mod_osso	18-25

OssoSecureCookies Directive in mod_osso	18-26
Mod_osso Does Not Encode the Return URL.....	18-26
mod_osso: "Page Not found" error After Default Installation	18-26
About Using IPv6.....	18-27

Part V Developing with Oracle Platform Security Services APIs

19 Integrating Application Security with OPSS

Introduction	19-1
Security Integration Use Cases	19-2
Authentication	19-3
Java EE Application Requiring Authenticated Users	19-3
Java EE Application Requiring Programmatic Authentication	19-4
Java SE Application Requiring Authentication	19-4
Identities	19-4
Application Running in Two Environments	19-4
Application Accessing User Profiles in Multiple Stores	19-5
Authorization.....	19-5
Java EE Application Accessible by Specific Roles.....	19-5
ADF Application Requiring Fine-Grained Authorization.....	19-5
Web Application Securing Web Services	19-6
Java EE Application Requiring Codebase Permissions.....	19-6
Non-ADF Application Requiring Fine-Grained Authorization.....	19-6
Credentials	19-6
Application Requiring Credentials to Access System	19-6
Audit	19-7
Auditing Security-Related Activity	19-7
Auditing Business-Related Activity.....	19-7
Identity Propagation.....	19-8
Propagating the Executing User Identity	19-8
Propagating a User Identity	19-9
Propagating Identities Across Domains	19-9
Propagating Identities over HTTP	19-9
Administration and Management	19-9
Application Requiring a Central Store.....	19-9
Application Requiring Custom Management Tool	19-10
Application Running in a Multiple Server Environment	19-10
Integration	19-10
Application Running in Multiple Domains	19-11
Some Use Cases Details	19-11
Propagating Identities over HTTP.....	19-11
The OPSS Trust Service.....	19-11
Propagating Identities over the HTTP Protocol	19-11
Domains Using Both Protocols	19-17
A Custom Graphical User Interface	19-18
Imports Assumed.....	19-20

Code Sample 1	19-20
Code Sample 2	19-21
Code Sample 3	19-21
Code Sample 4	19-22
Code Sample 5	19-23
Code Sample 6	19-25
Appendix - Security Life Cycle of an ADF Application	19-25
Development Phase	19-25
Deployment Phase	19-26
Management Phase	19-26
Summary of Tasks per Participant per Phase	19-26
Appendix - Code and Configuration Examples	19-27
Code Examples	19-27
Configuration Examples	19-28
Full Code Example of a Java EE Application with Integrated Security	19-28

20 The OPSS Policy Model

The Security Policy Model	20-1
Authorization Overview	20-1
Introduction to Authorization	20-1
The Java EE Authorization Model	20-2
Declarative Authorization	20-2
Programmatic Authorization	20-2
Java EE Code Example	20-3
The JAAS Authorization Model	20-4
The JAAS/OPSS Authorization Model	20-4
The Resource Catalog	20-4
Managing Policies	20-5
Checking Policies	20-6
Using the Method checkPermission	20-7
Using the Methods doAs and doAsPrivileged	20-11
Using the Method checkBulkAuthorization	20-12
Using the Method getGrantedResources	20-12
The Class ResourcePermission	20-12

21 Manually Configuring Java EE Applications to Use OPSS

Configuring the Servlet Filter and the EJB Interceptor	21-1
Interceptor Configuration Syntax	21-7
Summary of Filter and Interceptor Parameters	21-7
Configuring the Application Stripe for Application MBeans	21-8
Choosing the Appropriate Class for Enterprise Groups and Users	21-9
Packaging a Java EE Application Manually	21-9
Packaging Policies with Application	21-10
Packaging Credentials with Application	21-10
Configuring Applications to Use OPSS	21-11
Parameters Controlling Policy Migration	21-11
Policy Parameter Configuration According to Behavior	21-15

To Skip Migrating All Policies	21-16
To Migrate All Policies with Merging.....	21-16
To Migrate All Policies with Overwriting.....	21-16
To Remove (or Prevent the Removal of) Application Policies.....	21-17
To Migrate Policies in a Static Deployment	21-19
Recommendations	21-19
Using a Wallet-Based Credential Store	21-19
Parameters Controlling Credential Migration.....	21-20
Credential Parameter Configuration According to Behavior.....	21-20
To Skip Migrating Credentials.....	21-21
To Migrate Credentials with Merging	21-21
To Migrate Credentials with Overwriting	21-21
Supported Permission Classes	21-21
Policy Store Permission.....	21-22
Credential Store Permission	21-23
Generic Permission	21-23
Specifying Bootstrap Credentials Manually.....	21-23
Migrating Identities with migrateSecurityStore.....	21-24
Example of Configuration File jps-config.xml.....	21-25
22 Authentication for Java SE Applicaitons	
Links to Authentication Topics for Java EE Applications	22-1
Authentication for Java SE Applications.....	22-2
The Identity Store.....	22-2
Configuring an LDAP Identity Store in Java SE Applications.....	22-2
Supported Login Modules for Java SE Applications.....	22-3
The Identity Store Login Module	22-3
Using the Identity Store Login Module for Authentication	22-4
Using the Identity Login Module for Assertion	22-6
Using the OPSS API LoginService in Java SE Applications.....	22-8
Configuration Examples	22-9
23 Authorization for Java SE Applications	
Configuring Policy and Credential Stores in Java SE Applications	23-1
Configuring File-Based Policy and Credential Stores	23-1
Configuring LDAP-Based Policy and Credential Stores.....	23-2
Configuring DB-Based OPSS Security Stores.....	23-3
Unsupported Methods for File-Based Policy Stores	23-4
24 Developing with the Credential Store Framework	
About the Credential Store Framework API.....	24-1
Overview of Application Development with CSF	24-2
Setting the Java Security Policy Permissions.....	24-2
Guidelines for Granting Permissions.....	24-3
Permissions Grant Example 1.....	24-3
Permissions Grant Example 2.....	24-3

Guidelines for the Map Name	24-4
Configuring the Credential Store	24-5
Steps for Using the API	24-5
Using the CSF API in a Standalone Environment	24-5
Using the CSF API in Oracle WebLogic Server	24-6
Examples	24-6
Code for CSF Operations	24-6
Example 1: Java SE Application with Wallet Store	24-8
Example 2: Java EE Application with Wallet Store.....	24-11
Example 3: Java EE Application with LDAP Store	24-13
Best Practices	24-14

25 Developing with the User and Role API

Introduction to the User and Role API Framework	25-1
User and Role API and the Oracle WebLogic Server Authenticators	25-2
Summary of Roles and Classes	25-2
Working with Service Providers	25-5
Understanding Service Providers.....	25-5
Setting Up the Environment.....	25-5
Selecting the Provider.....	25-6
Creating the Provider Instance.....	25-7
Properties for Provider Configuration	25-7
Start-time and Run-time Configuration	25-8
ECID Propagation.....	25-10
When to Pass Configuration Values	25-10
Configuring the Provider when Creating a Factory Instance	25-11
Oracle Internet Directory Provider.....	25-11
Using Existing Logger Objects	25-11
Supplying Constant Values	25-12
Configuring Connection Parameters	25-12
Configuring a Custom Connection Pool Class	25-13
Configuring the Provider when Creating a Store Instance	25-13
Runtime Configuration	25-13
Programming Considerations	25-13
Provider Portability Considerations	25-14
Considerations when Using IdentityStore Objects	25-15
Provider Life cycle	25-15
Searching the Repository	25-15
Searching for a Specific Identity.....	25-16
Searching for Multiple Identities	25-16
Specifying Search Parameters	25-16
Using Search Filters	25-17
Operators in Search Filters	25-17
Handling Special Characters when Using Search Filters.....	25-17
Search Filter for Logged-In User.....	25-17
Examples of Using Search Filters	25-18
Searching by GUID	25-20

User Authentication	25-20
Creating and Modifying Entries in the Identity Store	25-20
Handling Special Characters when Creating Identities	25-21
Creating an Identity	25-21
Modifying an Identity	25-21
Deleting an Identity	25-22
Examples of User and Role API Usage	25-22
Example 1: Searching for Users.....	25-22
Example 2: User Management in an Oracle Internet Directory Store	25-24
Example 3: User Management in a Microsoft Active Directory Store.....	25-25
SSL Configuration for LDAP-based User and Role API Providers	25-28
Out-of-the-box Support for SSL	25-28
System Properties.....	25-28
SSL configuration.....	25-28
Customizing SSL Support for the User and Role API	25-29
SSL configuration.....	25-29
The User and Role API Reference	25-29
Developing Custom User and Role Providers	25-29
SPI Overview	25-29
Types of User and Role Providers	25-30
Developing a Read-Only Provider	25-30
SPI Classes Requiring Extension	25-30
oracle.security.idm.spi.AbstractIdentityStoreFactory	25-31
oracle.security.idm.spi.AbstractIdentityStore	25-31
oracle.security.idm.spi.AbstractRoleManager.....	25-32
oracle.security.idm.spi.AbstractUserManager	25-32
oracle.security.idm.spi.AbstractRoleProfile.....	25-32
oracle.security.idm.spi.AbstractUserProfile	25-33
oracle.security.idm.spi.AbstractSimpleSearchFilter	25-34
oracle.security.idm.spi.AbstractComplexSearchFilter	25-34
oracle.security.idm.spi.AbstractSearchResponse	25-34
Developing a Full-Featured Provider	25-35
Development Guidelines	25-35
Testing and Verification	25-35
Example: Implementing an Identity Provider	25-36
About the Sample Provider	25-36
Overview of Implementation.....	25-36
Configure jps-config.xml to use the Sample Identity Provider.....	25-37
Configure Oracle WebLogic Server.....	25-38

26 Developing with the Identity Directory API

About the Identity Directory API	26-1
Feature Overview	26-1
Summary of Classes	26-2
Identity Directory Configuration	26-3
Working with the Identity Directory API	26-3
Getting an Identity Directory API Instance.....	26-4

Performing CRUD Operations on Users and Groups	26-4
User Operations	26-4
Group Operations	26-4
Examples of Identity Directory API	26-5
Initialize and Obtain Identity Directory Handle	26-5
Create a User	26-7
Get a User	26-7
Modify a User	26-8
Simple Search for a User	26-8
Complex Search for Users	26-8
Create a Group	26-9
Get a Group	26-10
Get Group Using a Search Filter	26-10
Delete a Group	26-11
Add a Member to a Group	26-11
Delete a Member from a Group	26-11
SSL Configuration	26-12

27 Developing with the Keystore Service

About the Keystore Service API	27-1
Overview of Application Development with the Keystore Service	27-2
Setting the Java Security Policy Permission	27-2
Guidelines for Granting Permissions	27-3
Permissions Grant Example 1	27-3
Permissions Grant Example 2	27-4
Permissions Grant Example 3	27-4
Configuring the Keystore Service	27-5
Steps for Using the Keystore Service API	27-5
Using the Keystore Service API in a Standalone Environment	27-5
Using the Keystore Service API in Oracle WebLogic Server	27-6
Example of Keystore Service API Usage	27-6
Java Program for Keystore Service Operations	27-6
Policy Store Setup	27-8
Configuration File	27-8
About Using the Keystore Service in the Java SE Environment	27-9
Best Practices	27-9

28 Developing with the Audit Service

Application Integration with Audit Flow	28-1
Audit Metadata Model	28-2
Attribute Groups	28-2
Audit Attribute Data Types	28-2
Common Attribute Groups	28-3
Generic Attribute Groups	28-3
Custom Attribute Groups	28-4
Event Categories and Events	28-4
System Categories and Events	28-4

Component/Application Categories	28-5
The Audit Metadata Store	28-6
Integrating the Application with the Audit Framework	28-6
Create Audit Definition Files	28-7
Understand Mapping and Versioning Rules	28-9
Version Numbers	28-9
Custom Attribute to Database Column Mappings	28-10
Register Application with the Registration Service	28-11
Add Application Code to Log Audit Events	28-12
Audit Client API.....	28-12
Set System Grants.....	28-12
Obtain Auditor Instance.....	28-13
Integrate with Oracle Business Intelligence Publisher	28-13
Update and Maintain Audit Definitions	28-14

Part VI Appendices

A OPSS Configuration File Reference

Top- and Second-Level Element Hierarchy.....	A-1
Lower-Level Elements	A-2

B File-Based Identity and Policy Store Reference

Hierarchy of Elements in system-jazn-data.xml	B-1
Elements and Attributes of system-jazn-data.xml	B-4

C Oracle Fusion Middleware Audit Framework Reference

Audit Events	C-1
What Components Can be Audited?	C-1
What Events can be Audited?	C-2
Oracle Directory Integration Platform Events and their Attributes	C-2
Oracle Platform Security Services Events and their Attributes	C-6
Oracle HTTP Server Events and their Attributes	C-8
Oracle Internet Directory Events and their Attributes	C-9
Oracle Identity Federation Events and their Attributes	C-11
Oracle Virtual Directory Events and their Attributes.....	C-16
OWSM-Agent Events and their Attributes	C-18
OWSM-PM-EJB Events and their Attributes	C-19
Reports Server Events and their Attributes	C-20
WS-Policy Attachment Events and their Attributes	C-21
Oracle Web Cache Events and their Attributes	C-21
Oracle Web Services Manager Events and their Attributes.....	C-24
Event Attribute Descriptions	C-24
Pre-built Audit Reports	C-28
Common Audit Reports	C-29
Component-Specific Audit Reports	C-29
The Audit Schema	C-31

WLST Commands for Auditing	C-44
getNonJava EEAuditMBeanName	C-45
Description.....	C-45
Syntax	C-45
Example.....	C-45
getAuditPolicy	C-45
Description.....	C-45
Syntax	C-46
Example.....	C-46
setAuditPolicy	C-46
Description.....	C-46
Syntax	C-46
Example.....	C-47
getAuditRepository	C-47
Description.....	C-47
Syntax	C-47
Example.....	C-47
setAuditRepository	C-47
Description.....	C-48
Syntax	C-48
Example.....	C-48
listAuditEvents.....	C-48
Description.....	C-48
Syntax	C-48
Example.....	C-48
exportAuditConfig.....	C-49
Description.....	C-49
Syntax	C-49
Example.....	C-49
importAuditConfig.....	C-49
Description.....	C-49
Syntax	C-50
Example.....	C-50
Audit Filter Expression Syntax	C-50
Naming and Logging Format of Audit Files	C-51

D User and Role API Reference

Mapping User Attributes to LDAP Directories	D-1
Mapping Role Attributes to LDAP Directories	D-3
Default Configuration Parameters	D-4
Secure Connections for Microsoft Active Directory	D-9

E Administration with WLST Scripting and MBean Programming

Configuring OPSS Service Provider Instances with a WLST Script	E-1
Configuring OPSS Services with MBeans	E-3
List of Supported OPSS MBeans.....	E-3
Invoking an OPSS MBean.....	E-3

Programming with OPSS MBeans	E-4
Access Restrictions	E-11
Annotation Examples	E-11
Mapping of Logical Roles to WebLogic Roles	E-12
Particular Access Restrictions.....	E-13
F OPSS System and Configuration Properties	
OPSS System Properties	F-1
OPSS Configuration Properties	F-4
Policy Store Properties	F-4
Policy Store Configuration	F-4
Runtime Policy Store Configuration	F-10
Credential Store Properties.....	F-14
LDAP Identity Store Properties	F-15
Properties Common to All LDAP-Based Instances.....	F-21
Anonymous and Authenticated Roles Properties.....	F-23
Trust Service Properties	F-24
Audit Service Properties	F-25
Keystore Service Properties	F-26
G Upgrading Security Data	
Upgrading with upgradeSecurityStore	G-1
Examples of Use	G-4
Example 1 - Upgrading Identities.....	G-4
Example 2 - Upgrading to File-Based Policies.....	G-5
Example 3 - Upgrading to Oracle Internet Directory LDAP-Based Policies.....	G-5
Example 4 - Upgrading File-Based Policies to Use the Resource Catalog.....	G-6
Upgrading Policies with upgradeOpss	G-12
Command Syntax	G-13
H References	
OPSS API References	H-1
I OPSS Scripts	
Policy-Related Scripts.....	I-1
Credential-Related Scripts.....	I-2
Other Security Scripts	I-2
Audit Scripts.....	I-3
J Using an OpenLDAP Identity Store	
Using an OpenLDAP Identity Store.....	J-1
K Adapter Configuration for Identity Virtualization	
About Split Profiles	K-1

Configuring a Split Profile.....	K-2
Deleting a Join Rule.....	K-3
Deleting a Join Adapter	K-3
Changing Adapter Visibility.....	K-4

L Troubleshooting Security in Oracle Fusion Middleware

Diagnosing Security Errors	L-2
Log Files and OPSS Loggers.....	L-2
Diagnostic Log Files.....	L-2
Generic Log Files.....	L-2
Authorization Loggers	L-3
Offline OPSS Scripts Loggers	L-4
Other OPSS Loggers	L-5
Audit Loggers.....	L-5
Managing Loggers with Fusion Middleware Control.....	L-6
System Properties.....	L-7
jps.auth.debug	L-7
jps.auth.debug.verbose	L-8
Debugging the Authorization Process.....	L-9
Solving Security Errors.....	L-11
Understanding Sample Log Entries	L-11
Searching Logs with Fusion Middleware Control	L-13
Identifying a Message Context with Fusion Middleware Control	L-13
Generating Error Listing Files with Fusion Middleware Control	L-14
Reassociation Failure	L-14
Missing Policies in Reassociated Policy Store.....	L-16
Unsupported Schema	L-18
Server Fails to Start	L-19
Missing Required LDAP Authenticator	L-19
Missing Administrator Account	L-20
Missing Permission.....	L-21
Server with NFS-Mounted Domain Directory Fails to Start.....	L-21
Other Causes.....	L-22
Failure to Grant or Revoke Permissions - Case Mismatch	L-24
Failure to Connect to an LDAP Server	L-25
Failure to Connect to the Embedded LDAP Authenticator	L-26
User and Role API Failure	L-27
Failure to Access Data in the Credential Store	L-28
Failure to Establish an Anonymous SSL Connection	L-29
Authorization Check Failure	L-29
User Gets Unexpected Permissions	L-30
Security Access Control Exception	L-31
Runtime Permission Check Failure	L-32
Permission Failure Before Server Starts	L-33
Policy Migration Failure	L-34
Characters in Policies	L-35
Use of Special Characters in Oracle Internet Directory 10.1.4.3.....	L-35

XML Policy Store that Contains Certain Characters.....	L-35
Characters in Application Role Names.....	L-36
Missing Newline Characters in XML Policy Store	L-36
Granting Permissions in Java SE Applications	L-36
Troubleshooting Oracle Business Intelligence Reporting	L-37
Audit Templates for Oracle Business Intelligence Publisher	L-37
Oracle Business Intelligence Publisher Time Zone	L-37
Search Failure when Matching Attribute in Policy Store	L-37
Search Failure with an Unknown Host Exception.....	L-40
Incompatible Versions of Binaries and Policy Store	L-41
Incompatible Versions of Policy Stores	L-42
Need Further Help?.....	L-42

Index

List of Examples

7-1	Single-LDAP Configuration in Oracle WebLogic Server.....	7-6
7-2	Multi-LDAP Configuration in Oracle WebLogic Server.....	7-6
7-3	Multi-LDAP Configuration in Third-Party Application Servers.....	7-9
7-4	Querying the LDAP Identity Store Programmatically.....	7-11
17-1	logout.html Script.....	17-11
17-2	OIM Integration-Related Parameter Usage.....	17-27
18-1	SSO Authentication with Dynamic Directives.....	18-10
18-2	SSO Logout with Dynamic Directives.....	18-11
25-1	Simple Filter to Retrieve Users by Name.....	25-18
25-2	Simple Filter to Find Users by Language Preference.....	25-18
25-3	Complex Filter for Names by Starting Letter.....	25-18
25-4	Complex Filter with Restrictions on Starting Letter.....	25-18
25-5	Complete Search with Output.....	25-19
25-6	Obtaining the Identity of the Logged-in User.....	25-19
25-7	Obtaining the Role/Group Name.....	25-20
B-1	<jazn-policy>.....	B-27
B-2	<jazn-policy>.....	B-28

List of Figures

1-1	The OPSS Architecture	1-3
2-1	Application Policy Logical Model	2-3
7-1	The OPSS Identity Store Service	7-2
10-1	The Create Key Dialog	10-5
10-2	The Crteate Key Dialog	10-6
12-1	Audit Event Flow	12-6
13-1	Audit Schema	13-23
15-1	Identity Asserter Configuration with Oracle Access Manager and WebGates	15-8
15-2	Authenticator for Web and non-Web Resources	15-9
16-1	Pre-seeded Resources in the User ID Assertion Authentication Policy	16-5
16-2	Pre-seeded Responses in the User ID Assertion Policy	16-5
16-3	Pre-seeded Application SSO Authentication Policy and Resources	16-6
16-4	Pre-seeded Responses for the Application SSO Authentication Policy	16-6
16-5	Pre-seeded Application SSO Authorization Policy and Resources	16-7
16-6	Sample Authorization Policy for Trusted Header Assertion	16-28
17-1	Sample OAMCfgTool Policy Domain General Tab	17-30
17-2	Sample OAMCfgTool Policy Domain Resources Tab	17-30
17-3	Sample OAMCfgTool Policy Domain Authorization Rules Tab	17-30
17-4	Sample OAMCfgTool Policy Domain Default Rules Tab	17-31
17-5	Sample OAMCfgTool Policy Domain Policies Tab	17-32
17-6	OAMCfgTool Policy Domain Delegated Access Admins Tab	17-32
17-7	Sample OAMCfgTool Host Identifiers	17-33
17-8	Sample OAMCfgTool AccessGate Profile	17-34
17-9	Default Login Form for Single Sign-On with 10g WebGates	17-48
17-10	Create Policy Domain Page in the Oracle Access Manager Policy Manager	17-50
18-1	Location of OSSO Components in the Oracle WebLogic Security Framework	18-2
18-2	OSSO Identity Asserter Processing	18-3
19-1	Applications, Security Stores, and Management Tools	19-2
19-2	Identity Propagation with HTTP Calls	19-12
19-3	Mapping of Application Roles to Users and Groups	19-19
19-4	Application Life Cycle Phases	19-26
28-1	Integrating Applications with the Audit Framework	28-2

List of Tables

2-1	Granted and Inherited Permissions	2-7
5-1	Basic Administrative Security Tasks and Tools	5-2
6-1	Tools to Deploy Applications after Development	6-3
7-1	Global LDAP Identity Store Parameters.....	7-3
8-1	SSO Provider Properties	8-22
11-1	Keystore Service Commands	11-4
13-1	Attributes of Base Table IAU_BASE	13-23
14-1	List of Audit Reports	14-11
14-2	Attributes of Audit Reports.....	14-14
15-1	Summary: Identity Assertion Mechanisms for Oracle Access Manager	15-5
15-2	Differences in Authentication Provider Implementation Tasks for OAM 11g versus OAM 10g	15-12
16-1	Options to Create DER Format Files from PEM.....	16-11
16-2	Provisioning Methods for OAM 11g.....	16-12
16-3	Required Registration Details for OAM Agents.....	16-13
16-4	Connection Filter Rules.....	16-18
16-5	addOAMSSOProvider Command-line Arguments.....	16-22
16-6	SSO Sync Filter Properties and Sync Behavior	16-42
17-1	Options to Create DER Format Files from PEM.....	17-8
17-2	Oracle Access Manager Authentication Provider Common Parameters	17-14
17-3	Provider-Specific Parameters	17-14
17-4	Provider-Specific Parameters: Oracle Access Manager Authenticator.....	17-15
17-5	OAMCfgTool CREATE Mode Parameters and Values	17-18
17-6	Additional OIM Integration-Related Parameters and Values.....	17-26
17-7	OAMCfgTool VALIDATE Mode Parameters and Values	17-27
17-8	OAMCfgTool DELETE Mode Parameters	17-28
17-9	OAMCfgTool Known Issues	17-34
17-10	Connection Filter Rules.....	17-38
17-11	addOAMSSOProvider Command-line Arguments.....	17-44
17-12	SSO Sync Filter Properties and Sync Behavior	17-68
18-1	Headers Sent by Oracle HTTP Server	18-4
18-2	ssoreg Parameters to Register Oracle HTTP Server mod_osso.....	18-7
18-3	Connection Filter Rules.....	18-14
18-4	SSO Sync Filter Properties and Sync Behavior	18-17
19-1	Security Tasks for the Application Architect.....	19-27
19-2	Security Tasks for the Application Developer.....	19-27
19-3	Security Tasks for the Application Security Administrator	19-27
20-1	Comparing Authorization in the Java EE Model	20-2
20-2	Behavior of checkPermission According to JAAS Mode	20-7
21-1	Summary of JpsFilter and JpsInterceptor Parameters	21-8
21-2	Settings to Skip Policy Migration	21-16
21-3	Settings to Migrate Policies with Merging	21-16
21-4	Settings to Migrate Policies with Overwriting	21-16
21-5	Settings to Remove Policies	21-17
21-6	Settings to Prevent the Removal of Policies.....	21-17
21-7	Settings to Migrate Policies with Static Deployments.....	21-19
21-8	Settings Not to Migrate Policies with Static Deployments	21-19
21-9	Settings to Skip Credential Migration.....	21-21
21-10	Settings to Migrate Credentials with Merging	21-21
21-11	Settings to Migrate Credentials with Overwriting	21-21
22-1	Idstore Types	22-11
25-1	Classes and Interfaces in the User and Role API.....	25-3
25-2	LDAP Identity Provider Classes.....	25-7

25-3	Start-time Identity Provider Configuration Properties	25-8
25-4	Runtime Identity Provider Configuration Properties	25-9
25-5	SPI Classes to Extend for Custom Provider	25-31
25-6	Methods of AbstractSimpleSearchFilter	25-46
25-7	Methods of Complex Search Filter	25-53
26-1	Classes in the Identity Directory API.....	26-2
28-1	Audit Attribute Data Types.....	28-3
28-2	Parameters for Audit Registration Service.....	28-11
A-1	First- and Second-Level Elements in jps-config.xml.....	A-2
A-2	Scenarios for <extendedProperty>	A-5
A-3	Scenarios for <property>	A-15
B-1	Hierarchy of Elements in system-jazn-data.xml	B-1
C-1	Oracle Directory Integration Platform Events	C-2
C-2	Oracle Platform Security Services Events.....	C-6
C-3	Oracle HTTP Server Events	C-8
C-4	Oracle Directory Integration Platform Events	C-9
C-5	Oracle Identity Federation Events.....	C-11
C-6	Oracle Virtual Directory Events.....	C-17
C-7	OWSM-Agent Events	C-19
C-8	OWSM-PM-EJB Events	C-20
C-9	Reports Server Events	C-21
C-10	WS-Policy Attachment Events	C-21
C-11	Oracle Web Cache Events	C-22
C-12	Oracle Web Services Manager Events.....	C-24
C-13	Attributes of Audited Events	C-25
C-14	The Audit Schema.....	C-31
C-15	Additional Audit Schema Tables.....	C-42
C-16	WLST Audit Commands	C-44
D-1	User Attributes in UserProfile.Property	D-1
D-2	Role Attribute Values in LDAP Directories	D-4
D-3	Default Values - Oracle Internet Directory and Microsoft Active Directory	D-4
D-4	Default Values - Oracle Directory Server Enterprise Edition and Novell eDirectory ..	D-5
D-5	Default Values - OpenLDAP and Oracle Virtual Directory	D-7
D-6	Default Values - Oracle WebLogic Server LDAP	D-8
E-1	List of OPSS MBeans	E-3
E-2	Mapping of Logical Roles to WebLogic Groups	E-12
E-3	Roles Required per Operation.....	E-13
F-1	Java System Properties Used by OPSS.....	F-2
F-2	Policy Store Properties	F-5
F-3	Runtime Policy Store Properties.....	F-10
F-4	Credential Store Properties.....	F-14
F-5	LDAP-Based Identity Store Properties	F-16
F-6	Generic LDAP Properties.....	F-22
F-7	Anonymous and Authenticated Roles Properties.....	F-23
F-8	Trust Service Properties	F-24
F-9	Audit Service Properties	F-25
F-10	Keystore Service Properties	F-26
L-1	Log Files for Audit Diagnostics	L-5

Preface

This manual explains the features and administration of the Oracle Platform Security Services.

This preface is divided into the following sections:

- [Audience](#)
- [Documentation Accessibility](#)
- [Related Documentation](#)
- [Conventions](#)

Audience

The intended audience of this guide are experienced Java developers, administrators, deployers, and application managers who want to understand and use Oracle Platform Security Services.

The overall structure of the guide is divided into parts, each of which groups related major topics. Parts I through III are relevant to administrators; parts IV contains information about the OPSS policy model and is intended for developers; and part V contains reference information.

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 information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website 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.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit <http://www.oracle.com/support/contact.html> or visit <http://www.oracle.com/accessibility/support.html> if you are hearing impaired.

Related Documentation

Additional information is found in the following documents:

- *Oracle Fusion Middleware Administrator's Guide*
- *Oracle Fusion Middleware 2 Day Administration Guide*
- *Oracle Fusion Middleware Security and Administrator's Guide for Web Services*
- *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*
- *Oracle Fusion Middleware Integration Guide for Oracle Identity Management*
- *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Federation*
- *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*
- *Oracle Fusion Middleware Security Overview*
- *Oracle Fusion Middleware Administrator's Guide for Oracle Entitlements Server*
- *Oracle Fusion Middleware Third-Party Application Server Guide*
- For links to API documentation, see [Section H.1, "OPSS API References."](#)

For a comprehensive list of Oracle documentation or to search for a particular topic within Oracle documentation libraries, see <http://www.oracle.com/technology/documentation/index.html>.

Conventions

The following text conventions are used in this document:

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

What's New in This Guide

This chapter describes the most important changes introduced in releases 11gR1, 11gR1 PS1, 11gR1 PS2, Oracle Identity Management 11gR1, 11gR1 PS3, Oracle Identity Management 11gR1 PS1, and 11gR1 PS5.

New Features in Release 11gR1 PS5

The features introduced in release 11gR1 PS5 include the following:

- Encrypting credentials. For details, see [Section 10.2, "Encrypting Credentials."](#)
- Trusted Header Assertion with the Oracle Access Manager Identity Assertion Provider. For details, see [Chapter 15, "Introduction to Single Sign-On in Oracle Fusion Middleware"](#) and [Chapter 16, "Configuring Single Sign-On with Oracle Access Manager 11g"](#).
- Integrating application security with OPSS. For details, see [Chapter 19, "Integrating Application Security with OPSS."](#)
- Developing applications using the Audit Service. For details, see [Chapter 28, "Developing with the Audit Service"](#).
- Using the Identity Directory API in your applications. For details, see [Chapter 26, "Developing with the Identity Directory API"](#).
- Administering the Keystore Service. For details, see [Chapter 11, "Managing Keys and Certificates with the Keystore Service"](#).
- Developing applications using the Keystore Service. For details, see [Chapter 27, "Developing with the Keystore Service"](#).
- Upgrading to PS5 with `upgradeOpss`. For details, see [Appendix G, "Upgrading Security Data."](#)

Documentation updates include the following:

- Updates to the discussion of the Common Audit Framework. For details, see [Chapter 12, "Introduction to Oracle Fusion Middleware Audit Framework"](#) and [Chapter 13, "Configuring and Managing Auditing"](#).
- Procedures to enable SSL for the Identity Store Service. See [Section 7.5](#).

New Features in Oracle Identity Management 11gR1 PS1

The features introduced in Oracle Identity Management 11gR1 PS1 include the following:

- Oracle Entitlements Server, a tool that supersedes Oracle Authorization Policy Manager. For details, see *Oracle Fusion Middleware Administrator's Guide for Oracle Entitlements Server*.
- The stand-alone audit loader.

New Features in Release 11gR1 PS3

The features introduced in release 11gR1 PS3 include the following:

- Support for DB-based stores.
- Support for the IBM WebSphere Application Server.
- Support for identity virtualization, which allows querying multiple identity stores.
- Support for security administrative scripts on IBM WebSphere Application Server.
- The OPSS script `upgradeOpss` to upgrade security data from 11gR1 PS1 or 11gR1 PS2 to 11gR1 PS3.
- Additional OPSS scripts.
- Improved Fusion Middleware Control security pages.
- Enhanced OAMCfgTool for OAM 10g SSO, with additional parameters.
- User and Role API support for IBM Tivoli and Microsoft ADAM directories.

New Features in Oracle Identity Management 11gR1

The features introduced in Oracle Identity Management 11gR1 include the following:

- Oracle Authorization Policy Manager, a tool to manage application security artifacts. The set of available tools to administer application security is expanded to Oracle WebLogic Administration Console, Oracle Enterprise Manager Fusion Middleware Control, WLST commands, and Oracle Authorization Policy Manager.

Additions to This Guide

New material in this guide includes:

- An appendix that lists all security-related WLST commands.

New Features in Release 11gR1 PS2

The features introduced in release 11gR1 PS2 include the following:

- The Resource Catalog, a way of specifying resource types, resources, actions, and entitlements in an application policy grant. Starting with this release, OPSS supports resource-based policies with the introduction of the resource catalog.
- Instructions for developing custom User and Role providers.
- Use of the class `ResourcePermission` in permissions.
- New WLST commands to manage resource types.
- The system property `jps.deployment.handler.disabled` of the Oracle WebLogic Server has been introduced.
- A new use of the command `upgradeSecurityStore`.

- A new argument to the command `migrateSecurityStore` to control the migration behavior upon encountering duplicate items. It applies only when migrating application policies.

New Features in Release 11gR1 PS1

The features introduced in release 11gR1 PS1 include the following:

- The class Resource Permission.
- Principal name comparison has been enhanced.
- Manual settings for policy migration have been simplified. In particular, versioning the application is no longer required.
- The WLST command `migrateSecurityStore` supports the embedded LDAP store as a target.
- The configuration of the identity store has been simplified. For example, previously required properties such as `username.attr` and `login.name.attr` are no longer needed when configuring an LDAP identity store.
- The WLST command `reassociateSecurityStore` supports an existing LDAP node as a target.
- New and improved Oracle Fusion Middleware Control pages. In particular, using these pages, one can specify the SSO service to use in a domain.

New Features in Release 11gR1

The single most important new feature in the 11gR1 release is the introduction of the Oracle WebLogic Server as the environment where applications run and where security is provisioned.

The features introduced in release 11gR1 include the following:

- Support for application policies and roles, and the authenticated and anonymous users and roles
- Credential Store Framework
- Auditing framework for Oracle Platform Security Services (OPSS) events for credential and policy management, and authorization checks
- Support for application lifecycle security integrated with JDeveloper
- Enhanced authorization framework
- Consolidation of code-based and subject-based policies in `system-jazn-data.xml`
- Management of security with Oracle Fusion Middleware and WLST commands
- New security-related WLST commands

Desupported Features from 10.1.3.x

The features de-supported in release 11gR1 include the following:

- Jazn is replaced with OPSS.
- Jazn Realm API is replaced by the User and Role API.
- Migration of OSDT toolkit from proprietary objects to JCE is desupported.

- The identity store, as previously configured in `system-jazn-data.xml`, is replaced by the use of WebLogic authenticators.
- The functions of Oracle Jazn Administration Tool are replaced as follows:
 - User and Role CRUD operations are replaced by the use of the Embedded LDAP configured and operated with the Oracle WebLogic Administration Console
 - The configuration of login modules is replaced with the use of the Oracle WebLogic Administration Console to configure authenticators
- JavaSSO is no longer supported. On a Oracle WebLogic Server domain, Single Sign-On (SSO) is automatic within clusters only when session replication is turned on.

Links to Upgrade Documentation

To upgrade from a previous release to the current, see any of the following documents:

- *Oracle Fusion Middleware Upgrade Planning Guide*
- *Oracle Fusion Middleware Upgrade Guide for Java EE*
- *Oracle Fusion Middleware Upgrade Guide for Oracle SOA Suite, WebCenter, and ADF*
- *Oracle Fusion Middleware Upgrade Guide for Oracle Portal, Forms, Reports, and Discoverer*
- *Oracle Fusion Middleware Upgrade Guide for Oracle Identity Management*

Part I

Understanding Security Concepts

This part contains the following chapters:

- [Chapter 1, "Introduction to Oracle Platform Security Services"](#)
- [Chapter 2, "Understanding Users and Roles"](#)
- [Chapter 3, "Understanding Identities, Policies, Credentials, Keys, and Certificates"](#)
- [Chapter 4, "About Oracle Platform Security Services Scenarios"](#)

Introduction to Oracle Platform Security Services

Oracle Platform Security Services (OPSS) is a security platform that can be used to secure applications deployed in any of the supported platforms or in standalone applications. This chapter introduces the main features of this platform in the following sections:

- [What is Oracle Platform Security Services?](#)
- [OPSS Architecture Overview](#)
- [Oracle ADF Security Overview](#)
- [OPSS for Administrators](#)
- [OPSS for Developers](#)

The scope of this document does *not* include Oracle Web Services security. For details about that topic, see *Oracle Fusion Middleware Security and Administrator's Guide for Web Services*.

For an overview of Oracle Fusion Middleware security topics, see *Oracle Fusion Middleware Security Overview*.

1.1 What is Oracle Platform Security Services?

OPSS provides enterprise product development teams, systems integrators, and independent software vendors with a standards-based, portable, integrated, enterprise-grade security framework for Java SE and Java EE applications.

OPSS is the underlying security platform that provides security to Oracle Fusion Middleware including WebLogic Server, Server Oriented Architecture (SOA) applications, Oracle WebCenter, Oracle Application Development Framework (ADF) applications, and Oracle Entitlement Server. OPSS is designed to be portable to third-party application servers, so developers can use OPSS as the single security framework for both Oracle and third-party environments, thus decreasing application development, administration, and maintenance costs.

OPSS provides an abstraction layer in the form of application programming interfaces (APIs) that insulate developers from security and identity management implementation details. With OPSS, developers do not need to know the details of, for example, cryptographic key management, repository interfaces, or other identity management infrastructures. Using OPSS, in-house developed applications, third-party applications, and integrated applications benefit from the same, uniform security, identity management, and audit services across the enterprise.

For OPSS-related news, including FAQs, a whitepaper, and code examples, and forum discussions, see http://www.oracle.com/technology/products/id_mgmt/opss/index.html.

1.1.1 OPSS Main Features

OPSS complies with the following standards: role-based-access-control (RBAC); Java Enterprise Edition (Java EE); and Java Authorization and Authentication Services (JAAS).

Built upon these standards, OPSS provides an integrated security platform that supports:

- Authentication
- Identity assertion
- Authorization, based on fine-grained JAAS permissions
- The specification and management of application policies
- Secure storage and access of system credentials through the Credential Store Framework
- Secure storage and access of keys and certificates through the Keystore Service
- Auditing
- Role administration and role mappings
- The User and Role API
- Identity Virtualization
- Security configuration and management
- SAML and XACML
- Oracle Security Developer Tools, including cryptography tools
- Policy Management API
- Java Authorization for Containers (JAAC)

Details about a given OPSS feature functionality are found in subsequent chapters of this guide.

For details about the WebLogic Auditing Provider, see section *Configuring the WebLogic Auditing Provider* in *Oracle Fusion Middleware Securing Oracle WebLogic Server*.

1.1.2 Supported Server Platforms

OPSS is supported in the following application server platforms:

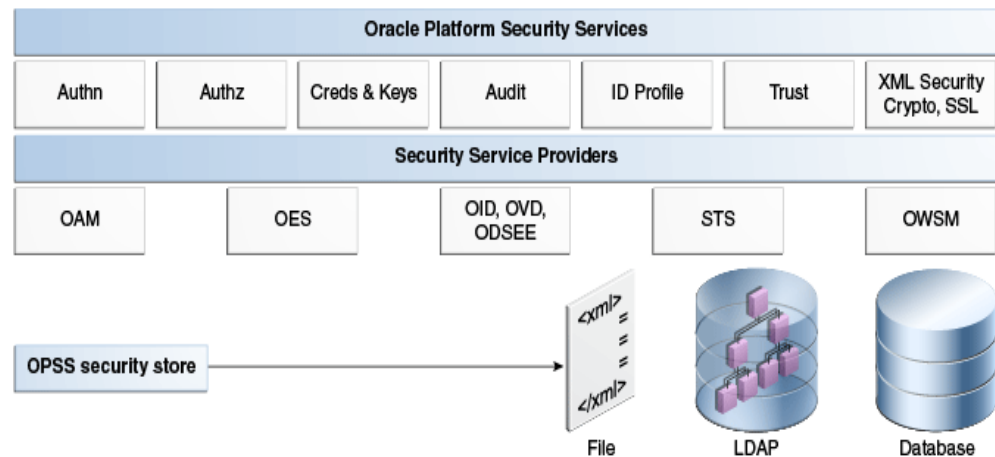
- Oracle WebLogic Server
- IBM WebSphere Application Server - Network Deployment (ND) 7.0
- IBM WebSphere Application Server 7.0

This guide documents OPSS features relevant to the Oracle WebLogic Server that apply uniformly to all other platforms. Those topics that apply specifically to third-party servers are found in *Oracle Fusion Middleware Third-Party Application Server Guide*.

1.2 OPSS Architecture Overview

OPSS comprises the application server's security and Oracle's Fusion Middleware security. [Figure 1-1](#) illustrates the layered architecture that combines these two security frameworks:

Figure 1-1 The OPSS Architecture



The top layer includes the OPSS security services; the next layer includes the service providers, and the bottom layer includes the OPSS security store with a repository of one of three kinds.

Security Services Providers

Security Services Provider Interface (SSPI) provides Java EE container security in permission-based (JACC) mode and in resource-based (non-JACC) mode, and resource-based authorization for the environment.

SSPI is a set of APIs for implementing pluggable security providers. A module implementing any of these interfaces can be plugged into SSPI to provide a particular type of security service, such as custom authentication or a particular role mapping.

For details, see section The Security Service Provider Interfaces (SSPIs) in *Oracle Fusion Middleware Understanding Security for Oracle WebLogic Server*.

Oracle Platform Security Services

Java Authorization (JAZN) functionality includes the Credential Store Framework (CSF), the Common Audit Framework (CAF), Keystore Service, and other components, and combined with SSPI as Oracle Platform Security Services (OPSS).

1.2.1 Benefits of Using OPSS

The benefits that OPSS offers include the following:

- Allows developers to focus on application and domain problems
- Supports enterprise deployments
- Supports several LDAP servers and SSO systems
- Is certified on the Oracle WebLogic Server
- Pre-integrates with Oracle products and technologies

- Offers a consistent security experience for developers and administrators
- Provides a uniform set of APIs for all types of applications
- Optimizes development time by offering abstraction layers (declarative APIs)
- Provides a simplified application security maintenance
- Allows changing security rules without affecting application code
- Eases the administrator's job
- Integrates with identity management systems
- Integrates with legacy and third-party security providers

OPSS combines SSPI and JPS to provide a framework where the application server and Oracle applications can seamlessly run in a single environment.

OPSS supports security for Java EE applications and for Oracle Fusion Middleware applications, such as Oracle WebCenter and Oracle SOA Suite.

Developers can use OPSS APIs to secure all types of applications and integrate them with other security artifacts, such as LDAP servers, RDBMS, and custom security components.

Administrators can use OPSS to deploy large enterprise applications with a small, uniform set of tools and administer all security in them. OPSS simplifies the maintenance of application security because it allows the modification of security configuration without changing the application code.

By default and out-of-the-box, Oracle WebLogic Server stores users and groups in its embedded LDAP repository. Domains can be configured, however, to use identity data in other kinds of LDAP repositories, such as Oracle Internet Directory, ActiveDirectory, Novell eDirectory, and OpenLDAP. In addition, Oracle WebLogic Server provides a generic, default LDAP authenticator that can be used with other LDAP servers not in the preceding list.

Out-of-the-box, policies and credentials are stored in file-based stores; these stores can be moved (or reassociated) to an LDAP repository backed by an Oracle Internet Directory.

Out-of-the-box, keys and certificates are stored in a file-based keystore, which can be reassociated with a database or an LDAP repository.

Note: This guide does not attempt to describe in detail WebLogic security features; wherever specific information about SSPI is used or assumed, the reader is referred to the appropriate document.

1.3 Oracle ADF Security Overview

Oracle ADF is an end-to-end Java EE framework that simplifies development by providing out-of-the-box infrastructure services and a visual and declarative development experience.

Oracle ADF Security is based on the JAAS security model, and it uses OPSS. Oracle ADF Security supports LDAP- or file-based policy and credential stores, uses permission-based fine-grained authorization provided by OPSS, and simplifies the configuration of application security with the aid of visual declarative editors and the Oracle ADF Security wizard, all of them available in Oracle JDeveloper 11g (any reference to this tool in this guide stands for its 11g release).

Oracle ADF Security authorization allows protecting components (flows and pages), is integrated with Oracle JDeveloper at design time, and is available at run time when the application is deployed to the integrated server where testing of security features is typically carried out.

During the development of an Oracle ADF application, the authenticators are configured with the Oracle WebLogic Server Administration Console for the particular domain where the application is deployed, and the policy store is file-based and stored in the file `jazn-data.xml`. For deployment details, see [Section 6.3.1, "Deploying to a Test Environment."](#)

To summarize, Oracle ADF Security provides:

- Control over granular declarative security
- Visual and declarative development of security artifacts
- Assignment of simplified permission through a role hierarchy
- Use of EL (expression language) to access Oracle ADF resources
- Integration with Oracle JDeveloper that allows quick development and test cycles
- Rich Web user interfaces and simplified database access

For related information, see [Scenario 2: Securing an Oracle ADF Application](#).

1.4 OPSS for Administrators

Depending on the application type, the guidelines to administer application security with Oracle WebLogic Administration Console, OPSS scripts, Fusion Middleware Control, or Oracle Entitlements Server are as follows:

- For Java EE applications, security is managed with Oracle WebLogic Administration Console, Oracle Entitlements Server, or OPSS scripts.
- For Oracle SOA, Oracle WebCenter, MDS, and Oracle ADF applications, authentication is managed with Oracle WebLogic Administration Console and authorization is managed with Fusion Middleware Control and Oracle Entitlements Server.
- For Java EE applications integrating with OPSS, authentication is managed using Oracle WebLogic Administration Console, and authorization is managed with Fusion Middleware Control and Oracle Entitlements Server.

For details about security administration, see [Chapter 5, "Security Administration."](#)

1.5 OPSS for Developers

This section summarizes the main OPSS features typically used when securing applications, in the following scenarios:

- [Scenario 1: Enhancing Security in a Java EE Application](#)
- [Scenario 2: Securing an Oracle ADF Application](#)
- [Scenario 3: Securing a Java SE Application](#)

For other use cases, see [Section 19.2, "Security Integration Use Cases."](#)

1.5.1 Scenario 1: Enhancing Security in a Java EE Application

A Java EE application can be enhanced to use OPSS APIs such as the CSF, User and Role, or Policy Management: user attributes, such as a user's email, phone, or address, can be retrieved using the Identity Governance Framework API or the User and Role API; external system credentials (stored in a wallet or in a LDAP-based store) can be retrieved using the CSF API; authorization policy data can be managed with the policy management APIs; and application keys and certificates can be managed with Keystore Service APIs.

Java EE applications, such as servlets, JSPs, and EJBs, deployed on Oracle WebLogic Server can be configured to use authentication and authorization declaratively, with specifications in the file `web.xml`, or programmatically, with calls to `isUserInRole` and `isCallerInRole`.

Custom authenticators include the standard basic, form, and client certification methods. Authentication between servlets and EJBs is controlled using user roles and enterprise groups, typically stored in an LDAP repository, a database, or a custom authenticators.

1.5.2 Scenario 2: Securing an Oracle ADF Application

Oracle Application Development Framework (ADF) is a Java EE development framework available in Oracle JDeveloper that simplifies the development of Java EE applications by minimizing the need to write code that implements the application's infrastructure, thus allowing developers to focus on the application features. Oracle ADF provides these infrastructure implementations as part of the Oracle JDeveloper framework, therefore enhancing the development experience with visual and declarative approaches to Java EE development.

Oracle ADF implicitly uses OPSS, and, for most part, the developer does not have to code directly to OPSS APIs; of course, the developer can nevertheless use direct calls to OPSS APIs.

Oracle ADF leverages container authentication and subsequently uses JAAS based authorization to control access to Oracle ADF resources. These authorization policies may include application-specific roles and JAAS authorization permissions. Oracle ADF connection credentials are stored securely in the credential store.

Oracle ADF and Oracle WebCenter applications deployed on Oracle WebLogic Server include WebLogic authenticators, such as the default WebLogic authenticator, and may include a single sign-on solution (Oracle Access Manager or Oracle Application Server Single Sign-On).

Usually, applications also use one or several of the following OPSS features: anonymous and authenticated role support, policy management APIs, and the Credential Store Framework.

For details about these topics, see the following sections:

- [Section 2.3, "The Authenticated Role"](#)
- [Section 2.4, "The Anonymous User and Role"](#)
- [Section 3.2, "Policy Store Basics"](#)
- [Section 3.3, "Credential Store Basics"](#)

For details on how to develop and secure Oracle ADF applications, see chapter 29 in *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

1.5.3 Scenario 3: Securing a Java SE Application

Most of the OPSS features that work in Java EE applications work in Java SE applications, but there are some differences, which are noted in this section.

Configuration

All OPSS-related configuration and data files are located under configuration directory in the domain home. For example, the configuration file for a Java SE environment is defined in the file `jps-config-jse.xml` by default installed in the following location:

```
$DOMAIN_HOME/config/fmwconfig/jps-config-jse.xml
```

To specify a different location, use the following switch:

```
-Doracle.security.jps.config=pathToConfigFile
```

The syntax of this file is identical to that of the file `jps-config.xml`. This file is used by code running in WebLogic containers. For details, see [Appendix A, "OPSS Configuration File Reference."](#)

For details about security configuration for Java SE applications, see [Section 22.2, "Authentication for Java SE Applications,"](#) and [Section 23.1, "Configuring Policy and Credential Stores in Java SE Applications."](#)

Required JAR in Class Path

To make OPSS services available to a Java SE application, ensure that the following JAR file is added to your class path, located in the modules area of the Oracle installation home:

```
$ORACLE_HOME/oracle_common/modules/oracle.jps_11.1.1/jps-manifest.jar
```

Login Modules

Java SE applications can use standard JAAS login modules. However, to use the same login module on WLS, implement a custom authentication provider that invokes the login module. The SSPI interfaces allow integrating custom authentication providers in WLS.

The login module recommended for Java SE applications is the IdentityStore login module.

For details, see section Authentication Providers in *Oracle Fusion Middleware Developing Security Providers for Oracle WebLogic Server*.

Understanding Users and Roles

This chapter describes various characteristics of users and roles, such as the anonymous role, the authenticated role, role mapping, and the role category. It also includes the definition of terms used throughout this guide and an overview of the User and Role API Framework.

OPSS delegates authentication to Oracle WebLogic Server authenticator providers managed with the WebLogic Administration Console.

This chapter is divided into the following sections:

- [Terminology](#)
- [Role Mapping](#)
- [The Authenticated Role](#)
- [The Anonymous User and Role](#)
- [Administrative Users and Roles](#)
- [Managing User Accounts](#)
- [Principal Name Comparison Logic](#)
- [The Role Category](#)

For further details about managing users and roles programmatically, see [Chapter 25](#), "Developing with the User and Role API."

2.1 Terminology

This section defines most of the OPSS security terms.

Users

A *user*, or *enterprise user*, is an end-user accessing a service. User information is stored in the identity store. An *authenticated user* is a user whose credentials have been validated.

An *anonymous user* is a user whose credentials have not been validated (hence unauthenticated) that is permitted access to only unprotected resources. This user is specific to OPSS and its use can be enabled or disabled by an application. For details about anonymous user support, see [Section 2.4](#), "The Anonymous User and Role."

Roles

An *enterprise role* or *enterprise group* is a collection of users and other groups. It can be hierarchical, that is, a group can include arbitrarily nested groups (other than itself).

A Java EE *logical role* is a role specified declaratively or programmatically by a Java EE application. It is defined in an application deployment descriptor and, typically, used in the application code. It can be mapped to only enterprise groups or users, and it cannot be mapped directly to application roles.

An *application role* is a collection of users, groups, and other application roles; it can be hierarchical. Application roles are defined by application policies and not necessarily known to a Java EE container. Application roles can be many-to-many mapped to external roles. For example, the external group `employee` (stored in the identity store) can be mapped to the application role `helpdesk service request` (in one stripe) and to the application role `self service HR` (in another stripe).

For details about the *anonymous role*, see [Section 2.4, "The Anonymous User and Role."](#)
For details about the *authenticated role*, see [Section 2.3, "The Authenticated Role."](#)

Principal

A *principal* is the identity to which the authorization in the policy is granted. A principal can be a user, an external role, or an application role. Most frequently, it is an application role.

Application Policy

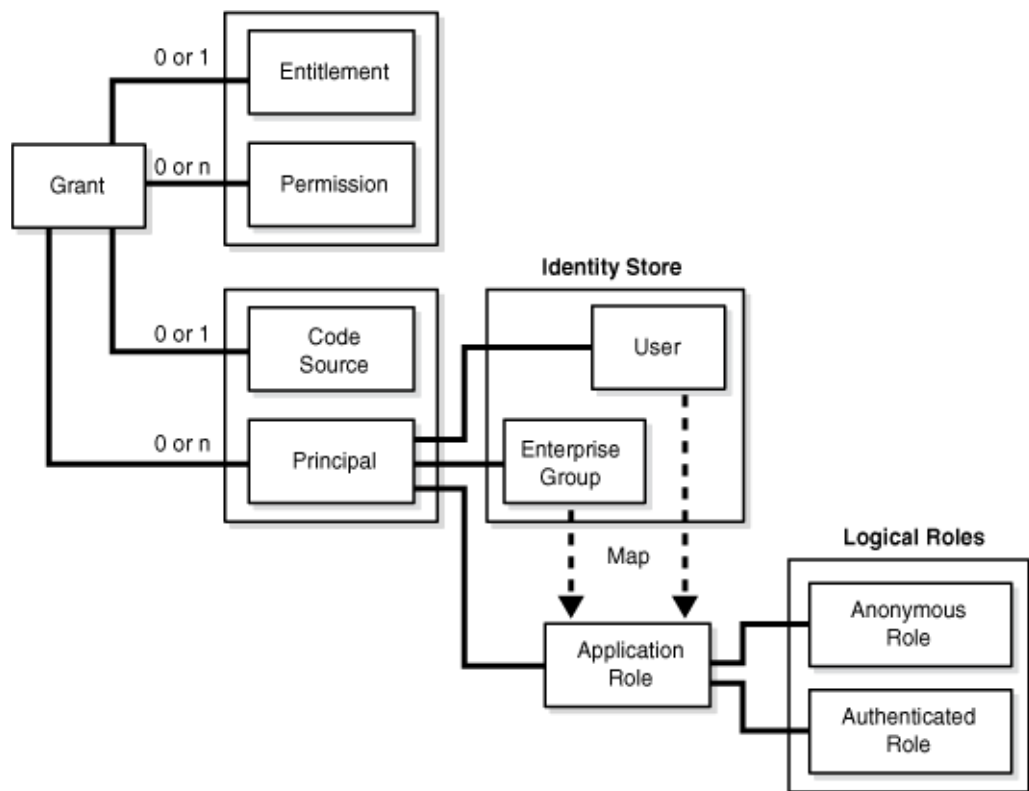
An *application policy* is a functional policy that specifies a set of permissions that an entity (the grantee, a principal or code source) is allowed within an application, such as viewing web pages or modifying reports. That is, it specifies who can do what in an application.

An application policy uses:

- Principals as grantees, and must have at least one principal.
- Either one or more permissions, or an entitlement, but not both.

Policies that use an entitlement are called *entitlement-based policies*; policies that use one or more permissions are called *resource-based policies*.

[Figure 2–1](#) illustrates the application policy model.

Figure 2–1 Application Policy Logical Model**OPSS Subject**

An OPSS *subject* is a collection of principals and, possibly, user credentials such as passwords or cryptographic keys. The server authentication populates the subject with users and groups, and then augments the subject with application roles. The OPSS Subject is key in identity propagation using other Oracle Identity Management products such as OAM, for example. For details about how anonymous data is handled, see [Section 2.4.1, "Anonymous Support and Subject."](#)

Security Stores

The *identity store* is the repository of enterprise users and groups and must be LDAP-based. Out-of-the-box the identity store is the WebLogic LDAP DefaultAuthenticator. Other types of identity stores include Oracle Internet Directory, Sun Directory Server, and Oracle Virtual Directory.

The *policy store* is the repository of application and system policies. This store is administered with Oracle Enterprise Manager Fusion Middleware Control.

The *credential store* is the repository of credentials. This store is administered with Oracle Enterprise Manager Fusion Middleware Control.

The *OPSS security store* is the logical repository of system and application-specific policies, credentials, and keys. The only type of LDAP-based OPSS security store supported is Oracle Internet Directory.

For details, see [Chapter 3, "Understanding Identities, Policies, Credentials, Keys, and Certificates."](#)

Other Terms

A *system component* is a manageable process that is not a WebLogic component. Examples include Oracle Internet Directory, WebCache, and Java SE components.

A *Java component* is a peer of a system component, but managed by an application server container. Generally it refers to a collection of applications and resources in one-to-one relationship with a domain extension template. Examples include Oracle SOA applications, Oracle WebCenter Spaces.

2.2 Role Mapping

OPSS supports many-to-many mapping of application roles in the policy store to enterprise groups in the identity store, which allows users in enterprise groups to access application resources as specified by application roles. Since this mapping is many-to-many, it is alternatively referred to as the role-to-group mapping or as the group-to-role mapping.

Notes: Oracle JDeveloper allows specifying this mapping when the application is being developed in that environment. Alternatively, the mapping can be also specified, after the application has been deployed, using OPSS scripts, Fusion Middleware Control, or Oracle Entitlements Server, as explained in [Section 9.2.2, "Managing Application Roles."](#)

The mapping of an application role to an enterprise group rewrites the privilege of the enterprise group as the union of its privileges and those of the mapped application role. Therefore, it (possibly) augments the privileges of the enterprise group but never removes any from it.

2.2.1 Permission Inheritance and the Role Hierarchy

OPSS roles can be structured hierarchically by the relation "is a member of." Thus a role can have as members users or *other* roles.

Important: When building a role hierarchy, ensure that you do not introduce circular dependencies to prevent unwanted behavior. For example, setting roleA to be a member of roleB, and roleB to be a member of roleA would create such a circular dependency.

In a role hierarchy, role members inherit permissions from the parent role. Thus, if roleA is a member of roleB, then all permissions granted to roleB are also permissions granted to roleA. Of course, roleA may have its own particular permissions, but, just by being a member of roleB, roleA inherits all the permissions granted to roleB.

For details about managing an application role hierarchy with OPSS scripts, see [Section 9.3.4, "grantAppRole,"](#) and [Section 9.3.5, "revokeAppRole."](#)

For details about managing an application role hierarchy with Oracle Entitlements Server, see *Oracle Fusion Middleware Administrator's Guide for Oracle Entitlements Server*.

The following example illustrates a role hierarchy consisting of the following nested application users and roles:

- The role `developerAppRole` has the following members:

```

developer
developer_group
managerAppRole
directorAppRole

```

- In addition, the role `directorAppRole` has the following members:

```

developer
developer_group

```

Here is the relevant portions of the file `jazn-data.xml` specifying the above hierarchy:

```

<policy-store>
  <applications>
    <application>
      <name>MyApp</name>
      <app-roles>
        <app-role>
          <name>developerAppRole</name>
          <class>oracle.security.jps.service.policystore.ApplicationRole</class>
          <display-name>Application developer role</display-name>
          <description>Application developer role</description>
          <guid>61FD29C0D47E11DABF9BA765378CF9F5</guid>
          <members>
            <member>
              <class>weblogic.security.principal.WLSUserImpl</class>
              <name>developer</name>
            </member>
            <member>
              <class>weblogic.security.principal.WLSGroupImpl</class>
              <name>developer_group</name>
            </member>
            <member>
              <class>
oracle.security.jps.service.policystore.ApplicationRole</class>
              <name>managerAppRole</name>
            </member>
          </members>
        </app-role>
        <app-role>
          <name>directorAppRole</name>
          <class>oracle.security.jps.service.policystore.ApplicationRole</class>
          <display-name>Application director role </display-name>
          <description>Application director role</description>
          <guid>61FD29C0D47E11DABF9BA765378CF9F8</guid>
          <members>
            <member>
              <class>weblogic.security.principal.WLSUserImpl</class>
              <name>developer</name>
            </member>
            <member>
              <class>weblogic.security.principal.WLSGroupImpl</class>
              <name>developer_group</name>
            </member>
          </members>
        </app-role> ...
      </app-roles>
    </application>
  </applications>
</policy-store>
<jazn-policy>
  <grant>

```

```

<grantee>
  <principals>
    <principal>
      <class>
oracle.security.jps.service.policystore.ApplicationRole</class>
      <name>developerAppRole</name>
    </principal>
  </principals>
</grantee>
<permissions>
<permission>
  <class>java.io.FilePermission</class>
  <name>/tmp/oracle.txt</name>
  <actions>write</actions>
</permission>
</permissions>
</grant>

<grant>
  <grantee>
    <principals>
      <principal>
        <class>
oracle.security.jps.service.policystore.ApplicationRole</class>
        <name>managerAppRole</name>
      </principal>
    </principals>
  </grantee>
  <permissions>
    <permission>
      <class>java.util.PropertyPermission</class>
      <name>myProperty</name>
      <actions>read</actions>
    </permission>
  </permissions>

</grant>
<grant>
  <grantee>
    <principals>
      <principal>
        <class>
oracle.security.jps.service.policystore.ApplicationRole</class>
        <name>directorAppRole</name>
      </principal>
    </principals>
  </grantee>
  <permissions>
    <permission>
      <class>foo.CustomPermission</class>
      <name>myProperty</name>
      <actions>*</actions>
    </permission>
  </permissions>
</grant>
</jaza-policy>
</policy-store>

```

Table 2–1 summarizes the permissions that each of the five users and roles in the above hierarchy gets according the inheritance rule:

Table 2–1 *Granted and Inherited Permissions*

Role	Permission Granted	Actual Permissions
developerAppRole	P1=java.io.FilePermission	P1
managerAppRole	P2= java.util.PropertyPermission	P2 and (inherited) P1
directorAppRole	P3=foo.CustomPermission	P3 and (inherited) P1
developer		P1 and P3 (both inherited)
developer_group		P1 and P3 (both inherited)

2.3 The Authenticated Role

OPSS supports the use of a special role: the authenticated role. This role has the following characteristics:

- It need not be declared in any configuration file.
- It is always represented by a principal attached to a subject after a successful authentication. In another words: it is granted by default to any authenticated user.
- Its presence, within a subject, is mutually exclusive with the anonymous role, that is, either (a) a subject has *not* gone through authentication, in which case it contains a principal with the anonymous role as explained in [Anonymous Support and Subject](#) or (b) the subject has gone through authentication successfully, in which case it contains the authenticated role and, depending on the configuration, the anonymous role.
- It is an application role and, therefore, it can be used by any application and participate in the application's role hierarchy.

The permissions granted to the authenticated role need not be specified explicitly but are implicitly derived from the enterprise groups and application roles of which it is a member.

A typical use of the authenticated role is to allow authenticated users access to common application resources, that is, to resources available to a user that has been authenticated.

For details on how an application can manually configure the use of the authenticated role, see [Section 21.1, "Configuring the Servlet Filter and the EJB Interceptor."](#)

2.4 The Anonymous User and Role

OPSS supports the use of two special entities: the anonymous user and the anonymous role. Like the authenticated role, these entities need not be declared and applications configure their use in the `JpsFilter` or `JpsInterceptor`. Any of them can be used by an application in the application's role hierarchy.

When enabled, before the user is authenticated and while the user is accessing unprotected resources, the user is represented by a subject populated with just the anonymous user and the anonymous role. Eventually, if that subject attempts access to a *protected* resource, then authorization handles the subject as explained in [Anonymous Support and Subject](#).

The permissions granted to the anonymous user and role need not be specified explicitly but are implicitly derived from the enterprise groups and application roles of which they are a member.

A typical use of the anonymous user and role is to allow unauthenticated users to access public, unprotected resources.

For details on how an application can manually configure the use of the anonymous user and role, see [Section 21.1, "Configuring the Servlet Filter and the EJB Interceptor."](#)

2.4.1 Anonymous Support and Subject

Throughout this section, it is assumed that the use of the anonymous user and anonymous role are enabled.

When an end-user first accesses an unprotected resource, the system creates a subject and populates it with two principals corresponding with the anonymous user and the anonymous role. While unprotected resources are involved, that subject is not modified and authentication does not take place.

When a protected resource is accessed, then authentication kicks in, and the subject (which thus far contained just the anonymous role) is modified according to the result of the authentication process, as follows.

If authentication is successful, then:

1. The anonymous user is removed from the subject and replaced, as appropriate, by an authenticated user.
2. The anonymous role is removed and the authenticated role is added.
3. Other roles are added to the subject, as appropriate.

Notice that a successful authentication results then in a subject that has exactly one principal corresponding to a non-anonymous user, one principal corresponding to the authenticated role, and possibly other principals corresponding to enterprise or application roles.

If authentication is not successful, then the anonymous user is retained, the anonymous role is removed or retained (according to how the application has configured the `JpsFilter` or `JpsInterceptor`), and no other principals are added. By default, the anonymous role is removed from the subject.

2.5 Administrative Users and Roles

A (WebLogic) administrator is any user member of the group Administrators, and any user that exists in a security realm can be added to this group.

For details about the default groups that exist in a security realm, see section *Users, Groups, And Security Roles in Oracle Fusion Middleware Securing Resources Using Roles and Policies for Oracle WebLogic Server*.

Generally, there is no default name for an administrator, with just one exception: when you install the examples, you get a default user name and password for the administrator of the sample domain. It is recommended, however, that these examples not be used in any production environment.

For details, see section *Install WebLogic Server in a Secure Manner in Oracle Fusion Middleware Securing a Production Environment for Oracle WebLogic Server*.

Once a domain is configured, users that have been created in the security realm can be added or removed from the Administrators group at anytime by any member of the Administrators group. The two basic tools for managing these accounts are the Oracle WebLogic Administration Console and the Oracle WebLogic Scripting Tool (WLST).

For details, see section *Add Users to Groups* in *Oracle Fusion Middleware Oracle WebLogic Server Administration Console Help*, and section *Using the WebLogic Scripting Tool* in *Oracle Fusion Middleware Oracle WebLogic Scripting Tool*.

2.6 Managing User Accounts

This section provides several links to information about creating user accounts and protecting their passwords.

- For general guidelines on creating passwords, see section *Manage Users and Groups* in *Oracle Fusion Middleware Oracle WebLogic Server Administration Console Help*. The default authentication provider requires a minimum password length of 8 characters, but this is configurable.

A few recommendations regarding password creation are explained in section *Securing the WebLogic Server Host* in *Oracle Fusion Middleware Securing a Production Environment for Oracle WebLogic Server*.

- In general, passwords are stored in either an LDAP server or an RDBMS. The particular location in which they are stored is determined by the specific authentication provider that is configured in the environment (or more precisely, the security realm of a domain). For details about out-of-the-box authentication providers, see section *Managing the Embedded LDAP Server* in *Oracle Fusion Middleware Securing Oracle WebLogic Server*.
- For information about how to configure the optional Password Validation provider, which is automatically called whenever you create a password and that enforces a set of customizable password composition rules, see section *Configuring the Password Validation Provider* in *Oracle Fusion Middleware Securing Oracle WebLogic Server*.
- When adding or deleting a user, consider the recommendations explained in [Section L.11, "User Gets Unexpected Permissions."](#)

2.7 Principal Name Comparison Logic

This section explains how principal comparison affects OPSS authorization and describes the system parameters that control the principal name comparison logic, in the following sections:

- [How Does Principal Comparison Affect Authorization?](#)
- [System Parameters Controlling Principal Name Comparison](#)

2.7.1 How Does Principal Comparison Affect Authorization?

Upon a successful user authentication, the system populates a Subject with principals whose names accord with the user and enterprise group names (of enterprise groups the user is included in) stored in the identity store.

On the other hand, when the user (or enterprise group) needs to be authorized, the system considers how application roles have been mapped to enterprise groups, and builds another set of principals from names in application grants stored in the policy store.

In order to authorize a principal, the principal names populated in the Subject (from names found in the identity store) and those built from names in the policy store are compared. The user (or group) is authorized if and only if a match of principal names is found.

It is therefore crucial that principal names be compared properly for the authorization provider to work as expected.

Suppose, for instance, a scenario where the identity store contains the user name "jdoe", but, in grants, that user is referred to as "Jdoe". Then one would want the principal name comparison to be case *insensitive*, for otherwise the principals built from the names "jdoe" and "Jdoe" will not match (that is, they will be considered distinct) and the system will not authorize "jdoe" as expected.

2.7.2 System Parameters Controlling Principal Name Comparison

The following two WebLogic Server system parameters control the way principal names are compared in a domain and allow, furthermore, to compare principals using DN and GUID data:

PrincipalEqualsCaseInsensitive (True or False; False by default)
PrincipalEqualsCompareDnAndGuid (True or False; False by default)

To set these parameters using the WebLogic Server Console, proceed as follows:

1. In the left pane of the Console, under **Domain Structure**, select the domain for which you intend to set the parameters above.
2. Select **Configuration > Security** and click **Advanced**.
3. Check (to set to true) or uncheck (to set to false) the box next to the following entries:
 - **Principal Equals Case Insensitive**
 - **Principal Equals Compare DN and GUID**
4. Restart the server. Changes do not take effect until the server is restarted.

These parameters can alternatively be set using OPSS scripts. For more details about configuring the WebLogic server, see section [Configuring a Domain to Use JAAS Authorization](#) in *Oracle Fusion Middleware Securing Oracle WebLogic Server*.

The name comparison logic chosen at runtime is described by the following pseudo-code fragment:

```
if PrincipalEqualsCompareDnAndGuid is true
//use GUID and DN to compare principals
{
  when GUID is present in both principals {
    use case insensitive to compare GUIDs
  }
  when DN is present in both principals {
    use case insensitive to compare DNs
  }
}

if PrincipalEqualsCaseInsensitive is true
//use just name to compare principals
{
  use case insensitive to compare principal names
}
else
{
  use case sensitive to compare principal names
}
```


Since by default both `PrincipalEqualsCompareDnAndGuid` and `PrincipalEqualsCaseInsensitive` are false, name principal comparison defaults to case sensitive.

2.8 The Role Category

The role category allows a security administrator to organize application roles. Rather than displaying the flat list of roles in an application, an administrator can organize them arbitrarily in flat sets or categories.

For details about managing an application role category with Oracle Entitlements Server, see *Oracle Fusion Middleware Administrator's Guide for Oracle Entitlements Server*.

The following fragment illustrates the configuration of a role category:

```
<role-categories>
  <role-category>
    <name>RC_READONLY</name>
    <display-name>RC_READONLY display name</display-name>
    <description>RC_READONLY description</description>
    <members>
      <role-name-ref>AppRole1</role-name-ref>
      <role-name-ref>AppRole2</role-name-ref>
      <role-name-ref>AppRole3</role-name-ref>
    </members>
  </role-category>
</role-categories>
```

The role category name is case insensitive. The role category can be managed with the interface `RoleCategoryManager`.

For details about this interface, see the Javadoc document *Oracle Fusion Middleware Java API Reference for Oracle Platform Security Services*.

Understanding Identities, Policies, Credentials, Keys, and Certificates

Applications use the identity, policy, credential stores and keystores configured in the domain in which they run. This chapter introduces the basic concepts regarding identity, policy, credential, and keystore data, and it is divided into the following sections:

- [Authentication Basics](#)
- [Policy Store Basics](#)
- [Credential Store Basics](#)
- [Keystore Service Basics](#)

For definitions of the terms used in this chapter, see [Section 2.1, "Terminology."](#)

For scenarios illustrating the use of stores, see [Chapter 4, "About Oracle Platform Security Services Scenarios."](#)

3.1 Authentication Basics

OPSS uses server authentication providers, components that validate user credentials or system processes based on a user name-password combination or a digital certificate. Authentication providers also make user identity information available to other components in a domain (through subjects) when needed.

Java EE applications must use LDAP-based authentication providers; Java SE applications use file-based identity stores out-of-the-box, but the identity store can be configured to be LDAP-based.

For further details, see section Authentication in *Oracle Fusion Middleware Understanding Security for Oracle WebLogic Server*.

Note: OPSS does not support automatic migration of users and groups used in application development to a remote WebLogic Server where an application may be deployed. Instead, one must independently create the necessary application identities using the Oracle WebLogic Administration Console, OPSS scripts, or the appropriate tool depending on the authentication provider(s) configured in your domain.

This section covers the following topics:

- [Supported LDAP Identity Store Types](#)

- [Oracle WebLogic Authenticators](#)
- [WebSphere Identity Stores](#)

3.1.1 Supported LDAP Identity Store Types

The following list enumerates the LDAP repositories supported for an identity store:

- Oracle Internet Directory 11g
- Oracle Virtual Directory
- Oracle Directory Server Enterprise Edition 11.1.1.3.0
- Active Directory 2008
- Novell eDirectory 8.8
- OpenLDAP 2.2. For the special configuration required for this type, see [Appendix J, "Using an OpenLDAP Identity Store."](#)
- Tivoli Access Manager
- Sun DS 6.3, 7.0
- Oracle DB 10g, 11gR1, 11gR2
- iPlanet Directory Server
- Custom Authenticator

For information about Oracle Fusion Middleware Certification and Supported Configurations, visit

http://www.oracle.com/technology/software/products/ias/files/fusion_certification.html.

In regards to support for reference integrity in Oracle Internet Directory servers, see Important note [Section 8.2, "Using an LDAP-Based OPSS Security Store."](#)

3.1.2 Oracle WebLogic Authenticators

For a list of WebLogic authenticator providers, see chapter 4, Authentication Providers in *Oracle Fusion Middleware Developing Security Providers for Oracle WebLogic Server*.

For details about the available authenticators, and choosing and configuring one, see section Configuring Authentication Providers in *Oracle Fusion Middleware Securing Oracle WebLogic Server*, and section Configure Authentication and Identity Assertion providers in *Oracle Fusion Middleware Oracle WebLogic Server Administration Console Help*.

By default and out-of-the-box, Oracle WebLogic Server stores users and groups in the DefaultAuthenticator. This authenticator is setup to use cn as the default attribute.

The data stored in any LDAP authenticator can be accessed by the User and Role API to query user profile attributes. For details about WebLogic LDAP authenticators, see the following sections:

- [Using an LDAP Authenticator](#)
- [Configuring the LDAP Identity Store Service](#)
- [Additional Authentication Methods](#)

Important: If your domain uses the DefaultAuthenticator, then the domain administration server *must* be running for an application to query data using the User and Role API.

OPSS requires that a domain have at least one LDAP-based authenticator configured in a domain.

For details about X.509 identity assertion, see section How an LDAP X509 Identity Assertion Provider Works in *Oracle Fusion Middleware Securing Oracle WebLogic Server*.

For details about authentication using the SAML 1.1 or SAML 2.0 identity assertion provider, see section Configuring the SAML Authentication Provider in *Oracle Fusion Middleware Securing Oracle WebLogic Server*.

3.1.2.1 Using an LDAP Authenticator

Oracle WebLogic Server offers several LDAP-based authenticators. For a choice of available LDAP servers for the identity store, see [Supported LDAP Identity Store Types](#). The Weblogic DefaultAuthenticator is the default authenticator configured and ready to use out-of-the-box after installation. Other authenticators can be configured using the WebLogic Administration Console.

For details about the use of authenticators in Java SE applications, see [Section 22.2.2, "Configuring an LDAP Identity Store in Java SE Applications."](#)

3.1.2.2 Configuring the LDAP Identity Store Service

Oracle WebLogic Server allows the configuration of multiple authenticators in a given context, each of which has a control flag set. One of them must be an LDAP-based authenticator.

OPSS initializes the identity store service with the LDAP authenticator chosen from the list of configured LDAP authenticators according to the following algorithm:

1. Consider the subset of LDAP authenticators configured. Note that, since the context is assumed to contain at least one LDAP authenticator, this subset is not empty.
2. Within that subset, consider those that have set the maximum flag. The flag ordering used to compute this subset is the following:

REQUIRED > REQUISITE > SUFFICIENT > OPTIONAL

Again, this subset (of LDAPs realizing the maximum flag) is not empty.

3. Within that subset, consider the first configured in the context.

The LDAP authenticator singled out in step 3 is the one chosen to initialize the identity store service. For details about host name verification when establishing an SSL connection with an LDAP authenticator, see *Oracle Fusion Middleware Securing Oracle WebLogic Server*.

For details about the default values that OPSS uses to initialize the various supported LDAP authenticators, see javadoc User and Role API documentation in [Section H.1, "OPSS API References."](#) If a service instance initialization value is provided by default and also (explicitly) in the service instance configuration, the value configured takes precedence over the default one.

Important: Any LDAP-based authenticator used in a domain, other than the `DefaultAuthenticator`, requires that the flag `UseRetrievedUserNameAsPrincipal` be set. Out-of-the-box, this flag is set in the `DefaultAuthenticator`.

3.1.2.3 Additional Authentication Methods

The WebLogic Identity Assertion providers support certificate authentication using X.509 certificates, SPNEGO tokens, SAML assertion tokens, and CORBA Common Secure Interoperability version 2 (CSIv2) identity assertion.

The Negotiate Identity provider is used for SSO with Microsoft clients that support the SPNEGO protocol. This provider decodes SPNEGO tokens to obtain Kerberos tokens, validates the Kerberos tokens, and maps Kerberos tokens to WebLogic users.

For general information about identity assertion providers, see section Identity Assertion Providers in *Oracle Fusion Middleware Understanding Security for Oracle WebLogic Server*.

For an overview of SSO with Microsoft clients, see section Overview of Single Sign-On with Microsoft Clients in *Oracle Fusion Middleware Securing Oracle WebLogic Server*.

For details about Kerberos identification, see section Creating a Kerberos Identification for WebLogic Server in *Oracle Fusion Middleware Securing Oracle WebLogic Server*.

3.1.3 WebSphere Identity Stores

On WebSphere, OPSS supports LDAP-based registries only; in particular, it does not support WebSphere's built-in file-based user registry.

For details about configuration and seeding a registry, see *Oracle Fusion Middleware Third-Party Application Server Guide*

3.2 Policy Store Basics

A Java 2 policy specifies the permissions granted to signed code loaded from a given location.

A JAAS policy extends Java 2 grants by allowing an optional list of principals; the semantics of the permissions are granted to only code from a given location, possibly signed, and run by a user represented by those principals.

JACC extends the Java 2 and JAAS permission-based policy to EJBs and Servlets by defining an interface to plug custom authorization providers, that is, pluggable components that allow the control and customizing of authorizations granted to running Java EE applications.

An application policy is a collection of Java 2 and JAAS policies, which is applicable to just that application (in contrast to a Java 2 policy, which are applicable to the whole JVM).

The policy store is a repository of system and application-specific policies and roles. Application roles can include enterprise users and groups specific to the application (such as administrative roles). A policy can use any of these groups or users as principals.

In the case of applications that manage their own roles, Java EE application roles (configured in files `web.xml` or `ejb-jar.xml`) get mapped to enterprise users and groups and used by application-specific policies.

Important: As long as a domain is pointing to a policy store, that policy store cannot be deleted from the environment.

Policy Store Types

A policy store can be file-, LDAP-, or DB-based. A file-based policy store is an XML file, and this store is the out-of-the-box policy store provider. The only LDAP-based policy store type supported is Oracle Internet Directory. The only DB-based policy store type supported is Oracle RDBMS (releases 10.2.0.4 or later; releases 11.1.0.7 or later; and releases 11.2.0.1 or later).

Policy Store Scope, Migration, and Reassociation

There is exactly one policy store per domain. During development, application policies are file-based and specified in the file `jazn-data.xml`.

When the application is deployed on WebLogic with Fusion Middleware Control, they can be automatically migrated into the policy store. For details about this feature, see [Section 8.6.1, "Migrating with Fusion Middleware Control."](#) By default, the policy store is file-based.

When the application is deployed on WebSphere, the behavior of migration at deployment can be manually specified as described in [Section 21.4.1, "Parameters Controlling Policy Migration,"](#) and [Section 21.4.4, "Parameters Controlling Credential Migration."](#)

For reassociation details, see [Section 8.5, "Reassociating the OPSS Security Store."](#)

Note: All permission classes must be specified in the system class path.

For details about the resource catalog support within a policy store, see [Section 20.3.1, "The Resource Catalog."](#)

3.3 Credential Store Basics

A credential store is a repository of security data (credentials) that certify the authority of users, Java components, and system components. A credential can hold user name and password combinations, tickets, or public key certificates. This data is used during authentication, when principals are populated in subjects, and, further, during authorization, when determining what actions the subject can perform.

OPSS provides the Credential Store Framework, a set of APIs that applications can use to create, read, update, and manage credentials securely.

Credential Store Types

A credential store can be file-, LDAP-, or DB-based. A file-based credential store, also referred to as wallet-based and represented by the file `cwallet.sso`, is the out-of-the-box credential store. The only LDAP-based credential store type supported is Oracle Internet Directory. The only DB-based credential store type supported is Oracle RDBMS (releases 10.2.0.4 or later; releases 11.1.0.7 or later; and releases 11.2.0.1 or later).

Credential Store Scope, Migration, and Reassociation

An application can use either the domain credential store or its own wallet-based credential store. The domain credential store can be wallet-based (by default), LDAP-, or DB-based. The only LDAP-based credential store type supported is Oracle Internet Directory.

The migration of application credentials to the credential store can be configured to take place automatically when the application is deployed. For details, see [Section 8.6.1, "Migrating with Fusion Middleware Control."](#)

Credentials can also be reassociated from one type of store to another. For details, see [Section 8.5, "Reassociating the OPSS Security Store."](#)

3.4 Keystore Service Basics

The Keystore Service provides a central repository for keystores and trust stores containing all the keys and certificates used by a domain's components and applications. This eliminates the need to associate keystores with individual applications.

The administrator works with a single user interface providing a unified way to view and manage all keystores.

3.4.1 Keystore Repository Types

The central repository can be any of the following:

- XML file-based

This is the out-of-the-box keystore repository, and it is named file-keystores.xml.

Note: This file is not present immediately after installation; rather, it is generated later.

- Oracle Database
- LDAP directory

3.4.2 Keystore Repository Scope and Reassociation

Keys and certificates in the domain keystore repository can be reassociated from one type to another. For details, see [Section 8.5, "Reassociating the OPSS Security Store"](#).

About Oracle Platform Security Services Scenarios

This chapter describes some typical security scenarios supported by Oracle Platform Security Services. It also includes the list of LDAP, DB, and XML servers supported, the management tools that an administrator would use to administer security data in each scenario, and the package requirements for policies and credentials.

These topics are explained in the following sections:

- [Supported LDAP-, DB-, and File-Based Services](#)
- [Management Tools](#)
- [Packaging Requirements](#)
- [Example Scenarios](#)
- [Other Scenarios](#)

4.1 Supported LDAP-, DB-, and File-Based Services

Oracle Platform Security Services supports the following LDAP-, DB-, and file-based repositories:

- For the OPSS security store:
 - If file-based, XML for the policy store and cwallet for the credential store.
 - If LDAP-based, Oracle Internet Directory (versions 10.1.4.3 or 11g) for the policy store and credential store.
 - If DB-based, Oracle RDBMS (releases 10.2.0.4 or later; releases 11.1.0.7 or later; and releases 11.2.0.1 or later).
- For the identity store, any of the LDAP authenticators supported by the Oracle WebLogic Server. An XML identity store is supported in only Java SE applications.
- For keystores:
 - XML file
 - LDAP (Oracle Internet Directory)
 - RDBMS (Oracle Database)

Important: If using Oracle Internet Directory 10.1.4.3 with OPSS, a mandatory one-off patch for bug number 8351672 is recommended on top of Oracle Internet Directory 10.1.4.3. Download the patch for your platform from Oracle Support at <http://myoraclesupport.oracle.com>.

To ensure optimal performance, the following Oracle Internet Directory tuning is recommended:

```
ldapmodify -D cn=orcladmin -w <password> -v <<EOF
dn: cn=dsconfig,cn=configsets,cn=oracle internet directory
changetype: modify
add: orclinmemfiltprocess
orclinmemfiltprocess: (objectclass=orcljaznpermission)
orclinmemfiltprocess: (objectclass=orcljazngranttee)
EOF
```

For details about LDAP authenticators, see section Configuring LDAP Authentication Providers in *Oracle Fusion Middleware Securing Oracle WebLogic Server*. In particular, the DefaultAuthenticator is available out-of-the-box, but its use is recommended only in developing environments for no more than ten thousand entries, for users, and for no more than twenty five hundred entries, for groups.

Policies, credentials, and keys stored in an LDAP-based store must use the same physical persistent repository. For details, see the following chapters:

- [Chapter 9, "Managing the Policy Store"](#)
- [Chapter 10, "Managing the Credential Store"](#)
- [Chapter 11, "Managing Keys and Certificates with the Keystore Service"](#)

The Oracle WebLogic Server requires that a domain DB-based OPSS security store be up and running for the server to start.

4.2 Management Tools

The tools available to a security administrator are the following:

- WebLogic Administration Console
- Oracle Enterprise Manager Fusion Middleware Control
- Oracle Entitlements Server
- OPSS scripts (available on all supported platforms)
- LDAP server-specific utilities

The tool to manage security data depends on the type of data stored and the kind of store used to keep that data. For applications deployed on WebSphere Application Server, there is also the WebSphere Application Server Administration Console; for details, see WebSphere Application Server documentation. Note that OPSS scripts are available for both platforms: WebLogic and WebSphere.

Users and Groups

If a domain uses the DefaultAuthenticator to store identities, then use the Oracle WebLogic Server Administration Console to manage the stored data. The data stored in the DefaultAuthenticator can also be accessed by the User and Role API to query

user profile attributes. To insert *additional* attributes to users or groups in the DefaultAuthenticator, an applications also uses the User and Role API.

Important: If your domain uses the DefaultAuthenticator, then the domain administration server *must* be running for an application to operate on identity data using the User and Role API.

For details about configuring this authenticator, see [Section 3.1.2.1, "Using an LDAP Authenticator."](#)

Otherwise, if authentication uses any other LDAP server different from the default authenticator or a DB, then, to manage users and groups, use the services of that LDAP server.

Policies, Credentials, Keys, and Certificates

Policies, keys, and credentials must use the same kind of storage (file-, LDAP-, or DB-based), and if LDAP-based, the same LDAP server (Oracle Internet Directory only).

To manage policies and credentials use Fusion Middleware Control as explained in [Section 9.2, "Managing Policies with Fusion Middleware Control"](#) and [Section 10.4, "Managing Credentials with Fusion Middleware Control,"](#) or the OPSS scripts, as explained in [Section 9.3, "Managing Application Policies with OPSS Scripts"](#) and [Section 10.5, "Managing Credentials with OPSS Scripts."](#)

Alternatively, to manage policy data, use Oracle Entitlements Server as explained in *Oracle Fusion Middleware Administrator's Guide for Oracle Entitlements Server*.

Keys and certificates are managed with Fusion Middleware Control and WLST. For details, see [Chapter 11, "Managing Keys and Certificates with the Keystore Service"](#).

The following list summarizes the tools used to manage security data:

- Identity data
 - Default Authenticator: use Administration Console
 - Other LDAP or DB stores: use utilities provided by the LDAP server or DB
- Policy and Credential data
 - File-based: use Fusion Middleware Control or WLST
 - LDAP-based: use Fusion Middleware Control, WLST, or Oracle Entitlements Server to manage policies.
- Keys and Certificates
 - Use WLST

Changes to policies, credentials, or keys do not require server restart; changes to the file `jps-config.xml` *do* require server restart.

Note: In general, domain configuration changes require the server to be restarted; however, changes to the domain data do not require the server to be restarted. An example of a domain configuration change is the reassociation of domain stores.

For details about the automatic migration of application policies and credentials to the domain stores when the application is deployed, see [Section 8.6, "Migrating the OPSS Security Store."](#)

For details about managing tools on WebSphere Application Server, see *Oracle Fusion Middleware Third-Party Application Server Guide*.

4.3 Packaging Requirements

File-based application policies are defined in the file `jazn-data.xml`. The only supported way to package this file with an application is to place it in the directory `META-INF` of an EAR file.

File-based application credentials are defined in a file that must be named `cwallet.sso`. The only supported way to package this file with an application is to place it in the directory `META-INF` of an EAR file. For details, see [Section 21.3, "Packaging a Java EE Application Manually."](#)

For information about deployment on WebLogic, see [Chapter 6, "Deploying Secure Applications."](#)

On WebSphere, the behavior at deployment is controlled by properties specified in the file `META-INF/opss-application.xml`. For details about policy migration, see *Oracle Fusion Middleware Third-Party Application Server Guide*. For details about credential migration, see *Oracle Fusion Middleware Third-Party Application Server Guide*.

Note: Oracle JDeveloper automatically packages the EAR file for a secured Oracle ADF application with all the required files (and with the appropriate security configurations), when the EAR file is produced within that environment.

4.4 Example Scenarios

The scenarios explained in this section describe the security features adopted by most Oracle ADF applications, Oracle WebCenter, and Web Services Manager Control.

They assume that the application employs a security scheme that has the following characteristics:

- **Authentication:** it uses the WebLogic Default Authenticator to store users and groups.
- **Authorization:** it uses fine-grained JAAS authorization supported by file-based policies and credentials packaged with the application and by policy and credential stores (file- or LDAP-based).

One of these security schemes is typically employed by applications, such as Oracle ADF or Oracle SOA applications, that require fine-grained JAAS authorization. The various security components in these cases are managed with the appropriate tool.

Based on these assumptions, the following scenarios are typical variations on the basic theme; note, however, that the list of variations is not exhaustive.

Related Documentation

For details about configuring the Default Authenticator, see section [Configure Authentication and Identity Assertion Providers](#) in *Oracle Fusion Middleware Oracle WebLogic Server Administration Console Help*.

For details about configuring the OPSS security store, see [Chapter 8, "Configuring the OPSS Security Store."](#)

For details about managing policies, see [Chapter 9, "Managing the Policy Store."](#)

For details about managing credentials, see [Chapter 10, "Managing the Credential Store."](#)

For details about managing Oracle Fusion Middleware on WebSphere Application Server, see *Oracle Fusion Middleware Third-Party Application Server Guide*.

For details about managing the Keystore Service, see the chapter on keystore management.

Common Scenario 1

This scenario describes a Java EE application during development.

Authentication: The application uses the Default Authenticator, typical in development environments.

Authorization: The policy and credential stores are file-based.

Variation: The application uses the WebLogic support for SSO and Java EE security.

For details about WebLogic support for SSO, see section [Configuring Single Sign-On with Web Browsers and HTTP Clients](#) in *Oracle Fusion Middleware Securing Oracle WebLogic Server*.

Common Scenario 2

This scenario describes a Java EE application during development.

Authentication: The application uses the Default Authenticator, typical in development environments.

Authorization: The policy and credential stores are LDAP-based using the services of the same instance of an Oracle Internet Directory LDAP server.

Variation: JAAS is enabled and policies include permissions for the anonymous and the authenticated roles.

For details about configuring support for the anonymous and authenticated roles, see [Section 2.3, "The Authenticated Role,"](#) and [Section 2.4, "The Anonymous User and Role."](#)

Common Scenario 3

This scenario describes a Java EE application during development.

Authentication: The application uses the Default Authenticator, typical in development environments.

Authorization: The policy and credential stores are LDAP-based using the services of the same instance of an Oracle Internet Directory LDAP server.

Variation: The application uses Java EE security, JAAS is enabled, and policies include permissions for the anonymous and the authenticated role. It also uses the Credential Store Framework (CSF) APIs to query, retrieve, and manage policies.

For details about configuring support for the anonymous and authenticated roles, see [Section 2.3, "The Authenticated Role,"](#) and [Section 2.4, "The Anonymous User and Role."](#)

For details about CSF APIs, see [Section 24.1, "About the Credential Store Framework API."](#)

4.5 Other Scenarios

The following scenarios differ from the common scenarios in that the application uses an authenticator other than the DefaultAuthenticator (typically used in the application development phase) or some API to access security data.

Scenario 4

Authentication: The application uses an LDAP authenticator (other than the DefaultAuthenticator).

Authorization: Both, the policy and credential use the same Oracle Internet Directory LDAP-based store.

Variation: The application uses the User and Role API to access user profiles in the DB and the Credential Store Framework (CSF) APIs to access credentials.

For details about User and Role API, see [Chapter 25, "Developing with the User and Role API."](#)

For details about CSF APIs, see [Section 24.1, "About the Credential Store Framework API."](#)

Scenario 5

Authentication: The application uses the Oracle Internet Directory LDAP authenticator, typical in test and production environments.

Authorization: The policy and credential stores are file-based and packaged with the application. These data is automatically mapped to domain security data at deployment.

Variation: Post-deployment, the policy and credential stores are reassociated to an LDAP-based store configured through one-way SSL transmission channel.

For details about automatic migration of application security data at deployment, see [Section 8.6, "Migrating the OPSS Security Store."](#)

For details about reassociation, see [Section 8.5, "Reassociating the OPSS Security Store."](#)

For details about SSL configuration and related topics, see the following:

- [Section Configuring SSL in *Oracle Fusion Middleware Securing Oracle WebLogic Server*.](#)
- [Oracle Fusion Middleware Administrator's Guide.](#)
- [Section Set up SSL in *Oracle Fusion Middleware Oracle WebLogic Server Administration Console Help*.](#)
- [Section Using SSL Authentication in Java Clients in *Oracle Fusion Middleware Programming Security for Oracle WebLogic Server*.](#)

Scenario 6

This scenario describes a Java SE application using OPSS APIs.

Authentication: The application uses the LoginService API.

Authorization: The application uses the method CheckPermission.

In addition, the application uses the User and Role API to query attributes into the domain authenticator, and the Credential Store Framework API to query the credential store.

Part II

Basic OPSS Administration

This part describes basic OPSS administration features in the following chapters:

- [Chapter 5, "Security Administration"](#)
- [Chapter 6, "Deploying Secure Applications"](#)

Security Administration

This chapter introduces the tools available to an administrator and the typical tasks to manage application security; it is divided into the following sections:

- [Choosing the Administration Tool According to Technology](#)
- [Basic Security Administration Tasks](#)
- [Typical Security Practices with Fusion Middleware Control](#)
- [Typical Security Practices with the Administration Console](#)
- [Typical Security Practices with Oracle Entitlements Server](#)
- [Typical Security Practices with OPSS Scripts](#)

For advanced administrator tasks, see [Appendix E, "Administration with WLST Scripting and MBean Programming."](#)

5.1 Choosing the Administration Tool According to Technology

The four basic tools available to a security administrator are Oracle Enterprise Manager Fusion Middleware Control, Oracle WebLogic Administration Console, Oracle Entitlements Server, and the Oracle WebLogic Scripting Tool (WLST). For further details on these and other tools, see chapter 3, *Getting Started Managing Oracle Fusion Middleware in Oracle Fusion Middleware Administrator's Guide*.

The main criterion that determines the tool to use to administer application security is whether the application uses just container-managed security (Java EE application) or it includes Oracle ADF security (Oracle ADF application).

Oracle-specific applications, such as Oracle Application Development Framework (Oracle ADF) applications, Oracle Server-Oriented Architecture (SOA) applications, and Web Center applications, are deployed, secured, and maintained with Fusion Middleware Control and Oracle Entitlements Server.

Other applications, such as those developed by third parties, Java SE, and Java EE applications, are typically deployed, secured, and administered with Oracle WebLogic Administration Console or with WLST.

The recommended tool to develop Java applications is Oracle JDeveloper 11g. This tool helps the developer configure file-based identity, policy, and credential stores through specialized graphical editors. In particular, when developing Oracle ADF applications, the developer can run a wizard to configure security for web pages associated with Oracle ADF resources (such as Oracle ADF task flows and page definitions), and define security artifacts using a specialized, visual editor for the file `jazn-data.xml`.

For details about procedures and related topics, see the following sections in the Oracle JDeveloper online help documentation:

- Securing a Web Application Using Oracle ADF Security
- Securing a Web Application Using Java EE Security
- About Oracle ADF Security as an Alternative to Security Constraints
- About Securing Web Applications

For further details about Oracle ADF Security and its integration with Oracle JDeveloper, see *Accessing the Oracle ADF Security Design Time Tools*, in *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

For further details about Oracle Entitlements Server, see *Oracle Fusion Middleware Administrator's Guide for Oracle Entitlements Server*.

5.2 Basic Security Administration Tasks

Table 5–1 lists some basic security tasks and the tools used to execute them. Recall that the tool chosen to configure and manage application security depends on the type of the application: for Java EE applications, which use just container-managed security, use the Oracle WebLogic Administration Console; for Oracle ADF applications, which use OPSS authorization, use Fusion Middleware Control and Oracle Entitlements Server.

Manual settings without the aid of the tools listed below are not recommended. For information about using the Oracle WebLogic Administration Console, see the list of links following the table below. For details about Oracle Entitlements Server, see *Oracle Fusion Middleware Administrator's Guide for Oracle Entitlements Server*.

Table 5–1 Basic Administrative Security Tasks and Tools

Task	Use Fusion Middleware Control Security Menu	Use Other Tool
Configure WebLogic Domains		WebLogic Admin Console
Configure WebLogic Security Realms		WebLogic Admin Console
Manage WebLogic Domain Authenticators		WebLogic Admin Console
Enable SSO for MS clients, Web Browsers, and HTTP clients.		WebLogic Admin Console
Manage Domain Administrative Accounts		WebLogic Admin Console
Configuring the identity store service		WebLogic Admin Console or the WebSphere command <code>configureIdentityStore</code>
Manage Credentials for Oracle ADF Application	Credentials	
Enable anonymous role in Oracle ADF Application	Security Provider Configuration	
Enable authenticated role in Oracle ADF Application	Security Provider Configuration	

Table 5–1 (Cont.) Basic Administrative Security Tasks and Tools

Task	Use Fusion Middleware Control Security Menu	Use Other Tool
Enable JAAS in Oracle ADF Application	Security Provider Configuration	
Map application to enterprise groups for Oracle ADF Application	Application Roles or Application Policies	Oracle Entitlements Server
Manage system-wide policies for Oracle ADF Applications	System Policies	
Configure OPSS Properties	Security Provider Configuration	
Reassociate Policy and Credential Stores	Security Provider Configuration	

Details about using the Oracle WebLogic Administration Console for the tasks above are found in the following documents:

- For general use of the Administration Console, see Oracle Fusion Middleware Oracle WebLogic Server Administration Console Help.
- To configure WebLogic domains, see *Oracle Fusion Middleware Understanding Domain Configuration for Oracle WebLogic Server*.
- To configure WebLogic security realms, see section Creating and Configuring a New Security Realm: Main Steps in *Oracle Fusion Middleware Securing Oracle WebLogic Server*.
- To manage WebLogic domain authenticators, see chapter 5 in *Oracle Fusion Middleware Securing Oracle WebLogic Server*.
- To configure SSO with MS clients, see chapter 6 in *Oracle Fusion Middleware Securing Oracle WebLogic Server*.
- To manage domain administrative accounts, see chapter 6 in *Oracle Fusion Middleware Securing Resources Using Roles and Policies for Oracle WebLogic Server*.
- For details about configuring an LDAP identity store, see [Section 3.1.2, "Oracle WebLogic Authenticators,"](#) and [Section 3.1.3, "WebSphere Identity Stores."](#)

Note: OPSS does not support automatic backup or recovery of server files. It is recommended that the server administrator periodically back up all server configuration files, as appropriate.

For details about backing up and recovering Oracle Fusion Middleware, see chapter 15, *Introducing Backup and Recovery*, in *Oracle Fusion Middleware Administrator's Guide*.

5.2.1 Setting Up a Brand New Production Environment

A new production environment based on an existing environment can be set up in either of the following ways:

- Replicating an established environment using Oracle Cloning utilities. For details, see section 9.5, *Cloning Oracle Fusion Middleware Entities*, in *Oracle Fusion Middleware Administrator's Guide*.

- Reinstalling software and configuring the environment, as it was done to set up the established environment.

5.3 Typical Security Practices with Fusion Middleware Control

Fusion Middleware Control is a Web-based tool that allows the administration of a network of applications from a single point. Fusion Middleware Control is used to deploy, configure, monitor, diagnose, and audit Oracle SOA applications, Oracle ADF applications, Oracle WebCenter, and other Oracle applications using OPSS. Note that this section mentions only security-related operations.

In regards to security, it provides several administration tasks; using this tool, an administrator can:

- Post-installation and before deploying applications, reassociate the policy and credential stores; for details, see [Section 8.5.1, "Reassociating with Fusion Middleware Control."](#)
- Post-installation and before deploying applications, define OPSS properties. For details, see [Section 8.7, "Configuring the Identity Provider, Property Sets, and SSO."](#)

- At deploy time, configure the automatic migration of file-based application policies and credentials to LDAP-based domain policies and credentials.

For details see:

- [Section 6.3, "Deploying Oracle ADF Applications to a Test Environment."](#)
- [Section 8.6, "Migrating the OPSS Security Store."](#)

- For each application after it is deployed:
 - Manage application policies. For details, see [Section 9.1, "Managing the Policy Store."](#)
 - Manage credentials; for details, see [Section 10.3, "Managing the Credential Store."](#)
 - Specify the mapping from application roles to users, groups, and application roles. For details, see [Section 9.2.2, "Managing Application Roles."](#)
- For the domain, manage system policies; for details see [Section 9.2.3, "Managing System Policies."](#)
- For the domain, manage OPSS properties; for details see [Section 8.7, "Configuring the Identity Provider, Property Sets, and SSO."](#)

For a summary of security administrative tasks and the tools used to execute them, see [Basic Security Administration Tasks](#).

For further details about other functions, see the Fusion Middleware Control online help documentation.

For details about managing Oracle Fusion Middleware on WebSphere Application Server, see *Oracle Fusion Middleware Third-Party Application Server Guide*.

5.4 Typical Security Practices with the Administration Console

The Oracle WebLogic Administration Console is a Web-based tool that allows, among other functions, application deployment and redeployment, domain configuration, and monitoring of application status. Note that this section mentions only security-related operations.

Typical tasks performed with the Oracle WebLogic Administration Console include the following:

- Starting and stopping Oracle WebLogic Servers; for details see section Starting and Stopping Servers in *Oracle Fusion Middleware Managing Server Startup and Shutdown for Oracle WebLogic Server*.
- Configuring Oracle WebLogic Servers and Domains; for details see section Configuring Existing Domains in *Oracle Fusion Middleware Oracle WebLogic Scripting Tool*.
- Deploying applications; for details, see *Oracle Fusion Middleware Deploying Applications to Oracle WebLogic Server*.
- Configuring fail over support; for details see section Failover and Replication in a Cluster in *Oracle Fusion Middleware Using Clusters for Oracle WebLogic Server*.
- Configuring WebLogic domains and WebLogic realms.
- Managing users and groups in domain authenticators.
- Enabling the use of Single Sign-On for MS clients, Web browsers, and HTTP clients.
- Managing administrative users and administrative policies.

For details about Oracle WebLogic Administration Console, see *Oracle Fusion Middleware Oracle WebLogic Server Administration Console Help*.

5.5 Typical Security Practices with Oracle Entitlements Server

Typical security tasks performed with Oracle Entitlements Server include the following:

- Searching application security artifacts.
- Managing application security artifacts, including policies.
- Viewing the external role hierarchy.
- Managing the application role hierarchy.

For a list of some of the most frequent security tasks to administer application security with Oracle Entitlements Server, see *Oracle Fusion Middleware Administrator's Guide for Oracle Entitlements Server*.

5.6 Typical Security Practices with OPSS Scripts

Most of the operations available in the Oracle WebLogic Administration Console can be effected with OPSS scripts, a set of command-line interface that allows the scripting and automation of administration tasks, including domain configuration and application deployment.

For the list of security-related OPSS scripts, see [Appendix I, "OPSS Scripts."](#) For the complete list of WLST scripts, see *Oracle Fusion Middleware WebLogic Scripting Tool Command Reference*.

For details about managing Oracle Fusion Middleware on WebSphere Application Server, see *Oracle Fusion Middleware Third-Party Application Server Guide*.

Deploying Secure Applications

An application can be deployed to an Oracle WebLogic Server using any of the following tools: the Oracle WebLogic Server Administration Console, Oracle Enterprise Manager Fusion Middleware Control, Oracle JDeveloper, or the WebSphere Application Server console. An application can also be started by setting the its bits in a location known to the WebLogic server, without the need to restart the server; this kind of application start is known as *hot deployment*.

The recommended way to deploy an application depends on the platform, the application type, and whether the application is in the developing phase or in a post-development phase. For example, in the post-development phase, typically, the application is started in a production environment by means of a hot deployment.

The recommendations stated in this chapter apply to Oracle ADF applications and to Java EE applications using OPSS.

During development, the application is typically deployed with Oracle JDeveloper to the embedded Oracle WebLogic Server. Once the application transitions to test or production environments, it is typically deployed with Fusion Middleware Control or the Oracle WebLogic Server Administration Console or by a hot deployment.

This chapter focuses on administrative tasks performed at deployment of an Oracle ADF or pure Java EE application. The last section explains the packaging requirements to secure Java EE applications, a topic relevant only when the application is packaged manually.

This chapter is divided into the following sections:

- [Overview](#)
- [Selecting the Tool for Deployment](#)
- [Deploying Oracle ADF Applications to a Test Environment](#)
- [Deploying Standard Java EE Applications](#)
- [Migrating from a Test to a Production Environment](#)

Additional Documentation

For further details about deployment, see Chapter 8, *Deploying Applications*, in *Oracle Fusion Middleware Administrator's Guide*.

For an overview of the entire security life-cycle of an application, from development to production, see *Oracle Fusion Middleware Security Overview*.

For details about securing an Oracle ADF application during development, see *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

For details about the application life cycle, see [Section 19.4, "Appendix - Security Life Cycle of an ADF Application."](#)

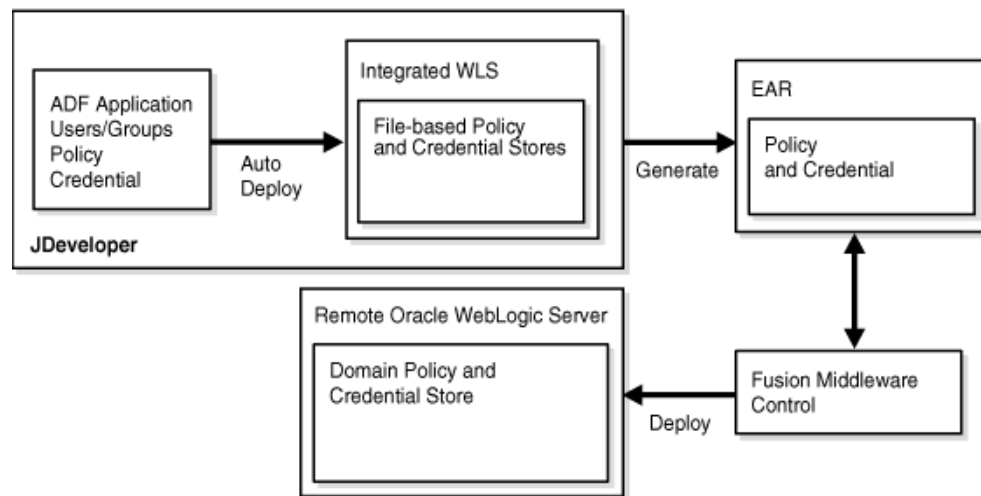
For details about the files in an EAR file relevant to application security management and configuration, such as `web.xml` and `weblogic-application.xml`, see [Chapter 21, "Manually Configuring Java EE Applications to Use OPSS."](#)

6.1 Overview

The steps that lead to the deployment of an Oracle ADF application into a remote Oracle WebLogic Server are, typically, as follows:

- Using Oracle JDeveloper, a developer develops an Oracle ADF application into which Oracle ADF security is included with the Oracle ADF Security Wizard.
- Application users and groups, authorization policies, and credentials are copied by Oracle JDeveloper to the integrated WebLogic Server, into which the application is auto-deployed during the test cycles in that environment.
- The developer creates an application EAR file which packs policies and credentials.
- The domain administrator deploys the EAR file to a remote Oracle WebLogic Server using Fusion Middleware Control.

This flow is illustrated in the following graphic:



6.2 Selecting the Tool for Deployment

The types of application we consider in this chapter are Java EE applications, which are further categorized into pure Java EE applications and Oracle Fusion Middleware ADF applications. The distinction of these two kinds of Java EE applications is explained in sections [Section 1.5.1, "Scenario 1: Enhancing Security in a Java EE Application,"](#) and [Section 1.5.2, "Scenario 2: Securing an Oracle ADF Application."](#)

[Table 6–1](#) lists the tool used to deploy a developed application according to its type.

Table 6–1 Tools to Deploy Applications after Development

Application Type	Tool to Use
Pure Java EE Application	Oracle WebLogic Administration Console, Fusion Middleware Control, WebSphere Application Server Administrator Console, WebSphere Application Server WASAdmin commands. The recommended tool is Oracle WebLogic Administration Console.
Oracle ADF Application	Fusion Middleware Control or OPSS script. The recommended tool is Fusion Middleware Control.

6.2.1 Deploying Java EE and Oracle ADF Applications with Fusion Middleware Control

This section focuses on the security configurations available when deploying an application that uses Oracle ADF security or a Java EE application that uses OPSS with Fusion Middleware Control on the WebLogic server.

Specifically, it describes the options you find in the page **Configure Application Security** at the third stage of the deploy settings.

The appearance of this page varies according to what is packaged in the EAR file, as follows:

- If the EAR file packages `jazn-data.xml` with application policies, the application policy migration section is shown.
- If the EAR file packages credentials in `cwallet.sso`, the credential migration section is shown.
- If the EAR file does not include any of the above, then the page displays the default Java EE security options.

This page, showing the policy migration sections, is partially illustrated in the following graphic:

The screenshot shows the 'Configure Application Security' page. At the top, there are navigation tabs: 'Deployment Settings', 'Configure Application Security' (which is active), and another 'Deployment Settings'. Below the tabs, the main heading is 'Configure Application Security'. A descriptive text says: 'Use this page to configure application authorization policy and credential migration behavior.' There are two main sections: 'Application Policy Migration' and 'Advanced Options'. Under 'Application Policy Migration', there are three radio buttons: 'Append' (selected), 'Overwrite', and 'Ignore'. Below these is a checkbox labeled 'Migrate only application roles and grants. Ignore identity store artifacts.' Under 'Advanced Options', there is a text input field for 'Application Stripe ID' with a 'Select' button next to it. At the bottom, there is a checkbox labeled 'Remove the policies during application undeployment.'

The settings in this page concern the migration of application policies and credentials (packed in application EAR file) to the corresponding domain store, and they are explained next.

Application Policy Migration Settings

These settings control of the policy migration in the following scenarios:

- If you are deploying the application for the first time, you typically want application policies to be migrated to the policy store. Therefore, select **Append** in the **Application Policy Migration** area.

If for some reason you do not want the migration to take place, select instead **Ignore**. The option **Overwrite** is also supported.

- If you are redeploying the application, and assuming that the migration of application policies has taken place in a previous deployment, you can choose **Append**, to merge the packed policies with the existing ones in the domain, or **Ignore**, to prevent policy migration.

The option **Ignore** is typically selected when an application is redeployed and you want to leave the current application policies in the domain unchanged, that is, when you want to preserve changes to the policy store made during previous deployments.

- When you choose **Append**, you can further specify which grants and roles should be migrated; the basic distinction is between ADF application roles and grants (needed in a production environment), and development-time only roles and grants (not needed in a production environment).

To migrate ADF application roles and grants, and not to migrate development-time only security roles and grants, check the box **Migrate only application roles and grants. Ignore identity store artifacts**. Typically, this box is checked when deploying to a production environment. Note that when this box is checked, you will need to map application roles to enterprise groups once the application has been deployed.

- When you choose **Append**, you can further specify a particular stripe (different from the default stripe, which is the application name) into which the application policies should be migrated, by entering the name of that stripe in the box **Application Stripe Id**.

About Application Stripes: The policy store is logically partitioned in stripes, one for each application name specified in the file `system-jazn-data.xml` under the element `<applications>`. Each stripe identifies the subset of domain policies pertaining to a particular application.

Typical Use Cases: This page supports specifying the migration of policies in the following two most common scenarios:

- Resolving inconsistent specifications found in the EAR file - The specifications in the EAR file are validated; if specifications regarding the application stripe found in the files `web.application.xml`, `web.xml`, and `ejb-jar.xml` (packed in the EAR file) are inconsistent (that is, do not match), you can enter a new stripe to use or select one from the drop-down list. The specified value trumps any other specified value in the EAR file and it is used as the target of the migration and in the runtime environment.
 - Allowing two or more applications to share an application stripe - If your application is to share an existing stripe (populated originally by some other application), you can specify that stripe. The **Overwrite** option should be used carefully when sharing an existing application stripe.
-

- If nothing is specified, the default settings are **Append** (in deployment) and **Ignore** (in redeployment).

Application Credential Migration Settings

These settings control of the credential migration in the following scenarios:

- If you are deploying the application for the first time, you typically want application credentials to be migrated to the credential store. Therefore, select **Append** in the **Application Credential Migration** area.
- In any case (first or succeeding deployment), if for some reason you do not want the migration to take place, select instead **Ignore**.

Note: Application code using credentials may not work if the credential migration is ignored. Typically, one would choose the **Ignore** option under the assumption that the credentials are manually created with the same map and key, but with different values.

- The option **Overwrite** is supported *only* when the WebLogic server is running in development mode.
- If nothing is entered, the default is **Ignore**.

6.3 Deploying Oracle ADF Applications to a Test Environment

An Oracle ADF application is a Java EE application using JAAS authorization, and it is typically developed and tested using Oracle JDeveloper; this environment allows a developer to package the application and deploy it in the Embedded Oracle WebLogic Server integrated with the tool. When transitioning to a test or production environment, the application is deployed using Oracle Fusion Middleware Control to leverage all the Oracle ADF security features that the framework offers. For details, see [Overview](#).

For step-by-step instructions on how to deploy an Oracle ADF application with Fusion Middleware Control, see:

- Section Deploy an Application Using Fusion Middleware Control in the Oracle Fusion Middleware Control online help system.
- Section 8.4, Deploying and Undeploying Oracle ADF Applications, in *Oracle Fusion Middleware Administrator's Guide*.

This section is divided into the following topics:

- [Deploying to a Test Environment](#)
- [Migrating from a Test to a Production Environment](#)

6.3.1 Deploying to a Test Environment

The security options available at deployment are explained in [Deploying Java EE and Oracle ADF Applications with Fusion Middleware Control](#).

When deploying an Oracle ADF application to a test environment with Fusion Middleware Control, the following operations take place:

Policy Management

- Application-specific policies packed with the application are automatically migrated to the policy store when the application is deployed.

Oracle JDeveloper automatically writes the necessary configuration for this migration to occur.

Note: Before migrating a file-based policy store (that is, the file `jazn-data.xml`) to a production environment, verify that any grant contains no duplicate permissions. If a duplicate permission (one that has the same name and class) appears in a grant, the migration runs into an error and it is halted. In this case, manually edit the `jazn-data.xml` file to remove any duplicate permissions from a grant definition, and invoke the migration again.

Credential Management

- Application-specific credentials packed with the application are automatically migrated to the credential store when the application is deployed.

Oracle JDeveloper automatically writes the necessary configuration for this migration to occur.

- The bootstrap credentials necessary to access LDAP repositories during migration are automatically produced by Fusion Middleware Control. For details about a manual setup, see [Section 21.4.7, "Specifying Bootstrap Credentials Manually."](#)

Identity Management

Identities packed with the application are not migrated. The domain administrator must configure the domain authenticator (with the Administration Console), update identities (enterprise users and groups) in the environment, as appropriate, and map application roles to enterprise users and groups (with Fusion Middleware Control).

Other Considerations

- When deploying to a domain with LDAP-based security stores and to preserve application data integrity, it is recommended that the application be deployed at the cluster level or, otherwise, to just one managed server.

- When deploying an application to multiple managed servers, be sure to include the administration server so that data is migrated as expected.
- The reassociation of domain stores is an infrequent operation and, typically, takes place when the domain is set up before applications are deployed. For procedure details, see [Section 8.5.1, "Reassociating with Fusion Middleware Control."](#)

6.3.1.1 Typical Administrative Tasks after Deployment in a Test Environment

At any time after an application is deployed in a test environment, an administrator can perform the following tasks using Fusion Middleware Control or the Administration Console:

- Map application roles to enterprise groups. Until this mapping is accomplished, security does not work as expected. For procedure details, see [Section 9.2.2, "Managing Application Roles."](#)
- Create additional application roles or customize existing ones. For details, see [Section 9.2.2, "Managing Application Roles."](#)
- Manage system policies. For procedure details, see [Section 9.2.3, "Managing System Policies."](#)
- Manage credentials. For procedure details, see [Section 10.3, "Managing the Credential Store."](#)

Notes: If the application is undeployed with Fusion Middleware Control from a server running in production mode, then the application-specific policies are automatically removed from the policy store. Otherwise, if you use any other tool to undeploy the application, then the removal of application-specific policies must be performed manually.

Credentials are not deleted upon an application undeployment. A credential may have started its life as being packaged with an application, but when the application is undeployed credentials are *not* removed.

6.4 Deploying Standard Java EE Applications

There are two ways to secure Java EE applications that do not use OPSS but that use standard Java authorization: administratively, with the Administration Console or a OPSS script; or programmatically, with deployment descriptors.

A Java EE application deployed to the Oracle WebLogic Server *is* a WebLogic resource. Therefore, an administrator would set security for the deployed application the same way that he would for any other resource.

For details about deployment procedures, see section 8.3, *Deploying and Undeploying Java EE Applications*, in *Oracle Fusion Middleware Administrator's Guide*.

For details about deploying applications with WLST commands, see section *Deployment Commands* in *Oracle Fusion Middleware WebLogic Scripting Tool Command Reference*.

For an overview of WebLogic Server deployment features, see chapter *Understanding WebLogic Server Deployment* in *Oracle Fusion Middleware Deploying Applications to Oracle WebLogic Server*.

Related Documentation

Further information about securing application resources, can be found in the following documents:

In *Oracle Fusion Middleware Securing Resources Using Roles and Policies for Oracle WebLogic Server*

- Section Application Resources
- Section Options for Securing Web Application and EJB Resources

In *Oracle Fusion Middleware Oracle WebLogic Server Administration Console Help*:

- Section Use Roles and Policies to Secure Resources

In *Oracle Fusion Middleware Securing WebLogic Web Services for Oracle WebLogic Server*:

- Section Overview of Web Services Security

In *Oracle Fusion Middleware Programming Security for Oracle WebLogic Server*:

- Section Securing Web Applications. Particularly relevant is the subsection Using Declarative Security with Web Applications
- Section Securing Enterprise JavaBeans (EJBs)
- Section Using Java Security to Protect WebLogic Resources

6.5 Migrating from a Test to a Production Environment

The recommendations that follow apply only to Java EE applications using JAAS authorization, such as Oracle Application Development Framework, Oracle SOA, and Oracle WebCenter applications, and they do not apply to Java EE applications using standard authorization. For deploying the latter, see [Deploying Standard Java EE Applications](#).

The recommended tool to deploy applications is Fusion Middleware Control, and the user performing the operations described in the following sections must have the appropriate privileges, including the privilege to seed a schema in an LDAP repository.

It is assumed that a production has been set up as explained in [Section 5.2.1, "Setting Up a Brand New Production Environment."](#)

Important Note: File-based stores are not recommended in production environments.

The migration to a new production environment is divided into three major portions: migrating providers other than policy or credential providers, migrating policy and credential providers, and migrating audit policies, as explained in the following sections:

- [Migrating Providers other than Policy and Credential Providers](#)
- [Migrating Policies and Credentials at Deployment](#)
- [Migrating Audit Policies](#)
- [Migrating Keystore Service Keys and Certificates](#)

Migration can be used for backup and recovery security data: to backup security data, migrate to an XML-based store; to recover security data, migrate from a saved XML-based store to the target security store.

6.5.1 Migrating Providers other than Policy and Credential Providers

The configuration of providers (other than policy and credential providers) in the production environment must be repeated as it was done in the test environment. This task may include:

- The identity store configuration, including the provisioning of required users and groups using the WebLogic Administrator Console or the OPSS script `configureIdentityStore`. For details about this last command, see [Migrating Identities Manually](#).
- Any particular provider configuration that you have performed in the test environment.

Note: Oracle WebLogic Server provides several tools to facilitate the creation of domains, such as the `pack` and `unpack` commands. For details, see *Oracle Fusion Middleware Creating Templates and Domains Using the Pack and Unpack Commands*.

6.5.1.1 Migrating Identities Manually

Identity data can be migrated manually from a source repository to a target repository using the OPSS script `migrateSecurityStore`. This migration is needed, for example, when transitioning from a test environment that uses a file-based identity store to a production environment that uses an LDAP-based identity store.

This script is **offline**, that is, it does not require a connection to a running server to operate; therefore, the configuration file passed to the argument `configFile` need not be an actual domain configuration file, but it can be assembled *just* to specify the source and destination repositories of the migration.

This script can be run in interactive mode or in script mode. In interactive mode, you enter the script at a command-line prompt and view the response immediately after. In script mode, you write scripts in a text file (with a `py` file name extension) and run it without requiring input, much like the directives in a shell script.

For platform-specific requirements to run an OPSS script, see [Important Note](#).

Script and Interactive Modes Syntaxes

To migrate identities on WebLogic, use the script (first) or interactive (second) syntaxes (arguments are written in separate lines for clarity):

```
migrateSecurityStore -type idStore
                    -configFile jpsConfigFileLocation
                    -src srcJpsContext
                    -dst dstJpsContext
                    [-dstLdifFile LdifFileLocation]
```

```
migrateSecurityStore(type="idStore", configFile="jpsConfigFileLocation",
src="srcJpsContext", dst="dstJpsContext", [dstLdifFile="LdifFileLocation"])
```

The migration of identities on WebSphere is accomplished with a similar script. For details, see *Oracle Fusion Middleware Third-Party Application Server Guide*.

The meaning of the arguments (all required except `dstLdifFile`) is as follows:

- `configFile` specifies the location of a configuration file `jps-config.xml` relative to the directory where the script is run.
- `src` specifies the name of a `jps-context` in the configuration file passed to the argument `configFile`, where the source store is specified.
- `dst` specifies the name of another `jps-context` in the configuration file passed to the argument `configFile`, where the destination store is specified. The destination store must be an LDAP-based identity store. For list of supported types, see [Section 3.1.1, "Supported LDAP Identity Store Types."](#)
- `dstLdifFile` specifies the relative or absolute path to the LDIF file created. Required only if destination is an LDAP-based Oracle Internet Directory store. Notice that the LDIF file is not imported into the LDAP server.

The contexts passed to `src` and `dst` must be defined in the passed configuration file and must have distinct names. From these two contexts, the script determines the locations of the source and the target repositories involved in the migration.

After an LDIF file is generated, the next step typically involves manual editing this file to customize the attributes of the LDAP repository where the LDIF file would, eventually, be imported.

6.5.2 Migrating Policies and Credentials at Deployment

In a production environment, it is strongly recommended that the OPSS security store (policy, credential, and key stores) be reassociated to an LDAP-based Oracle Internet Directory; if the test policy and credential stores were also LDAP, the production LDAP is assumed to be distinct from the test LDAP; if the test policy store was file-based, verify that no grant has duplicate permissions; see note in [Policy Management](#).

For details on how to reassociate stores, see [Section 8.5.1, "Reassociating with Fusion Middleware Control."](#)

The migration of policies and credentials can take place in the following ways: automatically, when an application is deployed; or manually, before or after the application is deployed.

Important Note: If the application is hot deployed, that is without stopping and restarting the server, the migration of data in the file `jazn-data.xml` to the domain security store is carried out *provided* the security store does not contain a stripe with the same name as the application. In particular, if the application is hot re-deployed (that is, hot deployed for a second or later time), any changes introduced in the file `jazn-data.xml` are *not* migrated over the domain security store.

To disable the automatic migration of policies and credentials for *all* applications deployed in a WebLogic Server (regardless of the application migration particular settings), set the system property `jps.deployment.handler.disabled` to `TRUE`.

When deploying an application to a production environment, an administrator should know the answer the following question:

Have policies or credentials packed in the application EAR been modified in the test environment?

Assuming that you know the answer to the above question, to deploy an application to a production environment, proceed as follows:

1. Use Fusion Middleware Control to deploy the application EAR file to the production environment using the following options:
 - If policies (application or system) have been modified in the test environment, then disable the option to migrate policies at deploy time by selecting the option **Ignore** under the **Application Policy Migration** area in Fusion Middleware Control's page **Configuration Application Security**; otherwise, select **Append**.

Note: You can select **Append** (that is, to migrate application policies) *in combination with* checking the box **Migrate only application roles and grants. Ignore identity store artifacts**, even when application roles have been modified in the test environment to the extent of mapping them to test enterprise groups.

Selecting this combination migrates application policies but disregards the maps to test enterprise groups. Later on, in step 3 below, you must remap application roles to production enterprise groups.

- If credentials have been modified in the test environment, then disable the option to migrate credentials at deploy time by selecting the option **Ignore** under the **Application Credential Migration** area in Fusion Middleware Control's page **Configuration Application Security**; otherwise, select **Append**.
2. Use the script `migrateSecurityStore` to migrate modified data, as follows:
 - If you chose to **Ignore** application policy migration, then migrate application and system policies from the test to the production LDAP. See example in [Migrating Policies Manually](#).
 - If you chose to **Ignore** application credential migration, then migrate credentials from the test to the production LDAP. See example in [Migrating Credentials Manually](#).
 3. In any case, use Fusion Middleware Control to map application roles to production enterprise groups, as appropriate.
 4. Use Fusion Middleware Control to verify that administrative credentials in the production environment are valid; in particular, test passwords versus production passwords; if necessary, modify the production data, as appropriate.

Note: There is a way to configure the application so that, at deployment, the migration of policies preserves GUIDs (instead of recreating them).

This setting can only be configured manually. For details, see parameter `jps.approle.preserveguid` in [Section 21.4.1, "Parameters Controlling Policy Migration."](#)

6.5.2.1 Migrating Policies Manually

By default, the script `migrateSecurityStore` recreates GUIDs and may take a long time to migrate large volume of policies; for these reasons, during the transition from a

test to a production environment, you may want to consider migrating policies and credentials with an alternate procedure that uses Oracle Internet Directory bulk operations. For details, see [Migrating Large Volume Policy and Credential Stores](#).

Migrating policies manually with the script `migrateSecurityStore` requires assembling a configuration file where the source and destination are specified.

Here is a complete sample of a configuration file, named `t2p-policies.xml`, illustrating the specification of policy sources in LDAP, DB, and XML storages, and of policy destinations in LDAP and DB storages:

```
<?xml version="1.0" encoding="UTF-8" standalone='yes'?>
<jpsConfig xmlns="http://xmlns.oracle.com/oracleas/schema/11/jps-config-11_1.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.oracle.com/oracleas/schema/11/jps-config-11_1.xsd" schema-major-version="11" schema-minor-version="1">

  <serviceProviders>
    <serviceProvider
class="oracle.security.jps.internal.policystore.xml.XmlPolicyStoreProvider"
name="policystore.xml.provider" type="POLICY_STORE">
      <description>XML-based policy store provider</description>
    </serviceProvider>

    <serviceProvider
class="oracle.security.jps.internal.policystore.ldap.LdapPolicyStoreProvider"
name="ldap.policystore.provider" type="POLICY_STORE">
      <property value="OID" name="policystore.type"/>
      <description>LDAP-based policy store provider</description>
    </serviceProvider>

    <serviceProvider
class="oracle.security.jps.internal.policystore.ldap.LdapPolicyStoreProvider"
name="db.policystore.provider" type="POLICY_STORE">
      <property value="DB_ORACLE" name="policystore.type"/>
      <description>DB-based policy store provider</description>
    </serviceProvider>
  </serviceProviders>

  <serviceInstances>
    <!-- Source XML-based policy store instance -->
    <serviceInstance location="./system-jazn-data.xml"
provider="policystore.xml.provider" name="policystore.xml.source">
      <description>Replace location with the full path of the folder where the
system-jazn-data.xml is located in the source file system </description>
    </serviceInstance>

    <!-- Source LDAP-based policy store instance -->
    <serviceInstance provider="ldap.policystore.provider"
name="policystore.ldap.source">
      <description>Replace: A. mySourceDomain and mySourceRootName to appropriate
values according to your source LDAP directory structure; B. OID with OVD,
if your source LDAP is OVD; C. ldap://mySourceHost.com:3060 with the URL
and port number of your source LDAP</description>
      <property value="OID" name="policystore.type"/>
      <property value="bootstrap" name="bootstrap.security.principal.key"/>
      <property value="cn=mySourceDomain" name="oracle.security.jps.farm.name"/>
      <property value="cn=mySourceRootName" name="oracle.security.jps.ldap.root.name"/>
      <property value="ldap://mySourceHost.com:3060" name="ldap.url"/>
    </serviceInstance>
  </serviceInstances>
</jpsConfig>
```

```

<!-- Source DB-based policy store instance -->
<serviceInstance provider="db.policystore.provider" name="policystore.db.source">
  <description>Replace: mySourceDomain and mySourceRootName to appropriate
    values according to your source DB policy store structure
  </description>
  <property value="DB_ORACLE" name="policystore.type"/>
  <property value="cn=mySourceDomain" name="oracle.security.jps.farm.name"/>
  <property value="cn=mySourceRootName" name="oracle.security.jps.ldap.root.name"/>
  <property value="jdbc:oracle:thin:@mySourceHost.com:1722:orcl" name="jdbc.url"/>
  <!-- the value of jdbc.url should be the value entered when the source
    datasource was set up -->
  <property value="oracle.jdbc.driver.OracleDriver" name="jdbc.driver"/>
  <property name="bootstrap.security.principal.key" value="mySourceKeyName" />
  <property name="bootstrap.security.principal.map" value="mySourceMapName" />
  <!-- the values of bootstrap.security.principal.key and
    bootstrapp.security.principal.map
    should be the values entered when the bootstrap credential was set up -->
</serviceInstance>

<!-- Destination LDAP-based policy store instance -->
<serviceInstance provider="ldap.policystore.provider"
name="policystore.ldap.destination">
<description>Replace: A. myDestDomain and myDestRootName to appropriate values
according to your destination LDAP directory structure; B. OID with OVD, if your
destination LDAP is OVD; C. ldap://myDestHost.com:3060 with the URL and port
number of your destination LDAP</description>
  <property value="OID" name="policystore.type"/>
  <property value="bootstrap" name="bootstrap.security.principal.key"/>
  <property value="cn=myDestDomain" name="oracle.security.jps.farm.name"/>
  <property value="cn=myDestRootName" name="oracle.security.jps.ldap.root.name"/>
  <property value="ldap://myDestHost.com:3060" name="ldap.url"/>
</serviceInstance>

<!-- Destination DB-based policy store instance -->
<serviceInstance provider="db.policystore.provider"
name="policystore.db.destination">
<description>Replace: myDestDomain and myDestRootName to appropriate values
according to your destination DB policy store structure</description>
  <property value="DB_ORACLE" name="policystore.type"/>
  <property value="cn=myDestDomain" name="oracle.security.jps.farm.name"/>
  <property value="cn=myDestRootName" name="oracle.security.jps.ldap.root.name"/>
  <property value="jdbc:oracle:thin:@myDestHost.com:1722:orcl" name="jdbc.url"/>
  <!-- the value of jdbc.url should be the value entered when the destination
    datasource was set up -->
  <property value="oracle.jdbc.driver.OracleDriver" name="jdbc.driver"/>
  <property name="bootstrap.security.principal.key" value="myDestKeyName" />
  <property name="bootstrap.security.principal.map" value="myDestMapName" />
  <!-- the value of bootstrap.security.principal.key and
    bootstrapp.security.principal.map
    should be the value entered when the bootstrap credential was set up -->
</serviceInstance>

<!-- Bootstrap credentials to access source and destination LDAPS or DBs-->
<serviceInstance location="./bootstrap" provider="credstoressp"
name="bootstrap.cred">
  <description>Replace location with the full path of the directory where the
bootstrap file cwallet.sso is located; typically found in
destinationDomain/config/fmwconfig/</description>
</serviceInstance>
</serviceInstances>

```

```

<jpsContexts>
<jpsContext name="XMLsourceContext">
<serviceInstanceRef ref="policystore.xml.source"/>
</jpsContext>

<jpsContext name="LDAPsourceContext">
<serviceInstanceRef ref="policystore.ldap.source"/>
</jpsContext>

<jpsContext name="DBsourceContext">
<serviceInstanceRef ref="policystore.db.source"/>
</jpsContext>

<jpsContext name="LDAPdestinationContext">
<serviceInstanceRef ref="policystore.ldap.destination"/>
</jpsContext>

<jpsContext name="DBdestinationContext">
<serviceInstanceRef ref="policystore.db.destination"/>
</jpsContext>

<!-- Do not change the name of the next context -->
<jpsContext name="bootstrap_credstore_context">
<serviceInstanceRef ref="bootstrap.cred"/>
</jpsContext>
</jpsContexts>
</jpsConfig>

```

Note that since the migration involves LDAP and DB stores, the file includes a `jps-context` named `bootstrap_credstore_context` that specifies the directory where the bootstrap credential file `cwallet.sso` is located. Furthermore, for each pair of map name and key name in the sample above, you must provide the corresponding bootstrap credentials using the WLST script `addBootStrapCredential` as illustrated in the following example:

```

wls:/offline> addBootStrapCredential(jpsConfigFile='jps-config.xml',
map='myMapName', key='myKeyName', username='myUserName',
password='myPassword')

```

where `myUserName` and `myPassaword` specify the user account name and password to access the target database.

The following examples of use of `migrateSecurityStore` assume that:

- The file `t2p-policies.xml` is located on the target system in the directory where the script is run.
- The directory structure of LDAP or DB system policies in the test and production environments should be *identical*. If this is not the case, before using the script, restructure manually the system policy directory in the production environment to match the corresponding structure in the test environment.

Under these assumptions, to migrate policies from a test (or source) LDAP store to a production (or destination) LDAP store, invoke `migrateSecurityStore` in the target system as follows:

```

>migrateSecurityStore(type="policyStore",configFile="t2p-policies.xml",src="LDAPso
urceContext",dst="LDAPdestinationContext")

```

To migrate policies from a test (or source) XML store to a production (or destination) LDAP store, invoke `migrateSecurityStore` in the target system as follows:

```
>migrateSecurityStore(type="policyStore", configFile="t2p-policies.xml", src="XMLsourceContext", dst="LDAPdestinationContext")
```

To migrate policies from a test (or source) DB store to a production (or destination) DB store, invoke `migrateSecurityStore` in the target system as follows:

```
>migrateSecurityStore(type="policyStore", configFile="t2p-policies.xml", src="DBsourceContext", dst="DBdestinationContext")
```

6.5.2.2 Migrating Credentials Manually

The script `migrateSecurityStore` recreates GUIDs and may take a long time to migrate large volume of credentials; for these reasons, during the transition from a test to a production environment, you may want to consider migrating policies and credentials with an alternate procedure that uses Oracle Internet Directory bulk operations. For details, see [Migrating Large Volume Policy and Credential Stores](#).

Migrating credentials manually with `migrateSecurityStore` requires assembling a configuration file where the source and destination are specified.

Since `migrateSecurityStore` recreates GUIDs and takes a long time to migrate large volume of data, you may want to consider migrating stores with an alternate procedure that uses Oracle Internet Directory bulk operations. For details, see [Migrating Large Volume Policy and Credential Stores](#).

Here is a complete sample of a configuration file, named `t2p-credentials.xml`, illustrating the specification of credential sources in LDAP, DB, and XML storages, and of credential destinations in LDAP or DB storages:

```
<?xml version="1.0" encoding="UTF-8" standalone='yes'?>
<jpsConfig xmlns="http://xmlns.oracle.com/oracleas/schema/11/jps-config-11_1.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.oracle.com/oracleas/schema/11/jps-config-11_1.xsd" schema-major-version="11" schema-minor-version="1">

<serviceProviders>
  <serviceProvider
class="oracle.security.jps.internal.credstore.ssp.SspCredentialStoreProvider"
name="credstoressp" type="CREDENTIAL_STORE">
  <description>File-based credential provider</description>
</serviceProvider>

  <serviceProvider
class="oracle.security.jps.internal.credstore.ldap.LdapCredentialStoreProvider"
name="ldap.credentialstore.provider" type="CREDENTIAL_STORE">
  <description>LDAP-based credential provider</description>
</serviceProvider>

  <serviceProvider
class="oracle.security.jps.internal.credstore.rdbms.DbmsCredentialStoreProvider"
name="db.credentialstore.provider" type="CREDENTIAL_STORE">
  <description>DB-based credential provider</description>
</serviceProvider>
</serviceProviders>

<serviceInstances>
  <!-- Source file-based credential store instance -->
  <serviceInstance location="myFileBasedCredStoreLocation" provider="credstoressp"
```

```

name="credential.file.source">
  <description>Replace location with the full path of the folder where the
file-based source credential store cwallet.sso is located in the source file
system; typically located in sourceDomain/config/fmwconfig/
</description>
</serviceInstance>

<!-- Source LDAP-based credential store instance -->
<serviceInstance provider="ldap.credentialstore.provider"
name="credential.ldap.source">
  <description>Replace: A. mySourceDomain and mySourceRootName to appropriate
values according to your source LDAP directory structure; B.
ldap://mySourceHost.com:3060 with the URL and port number of your source
LDAP</description>
  <property value="bootstrap" name="bootstrap.security.credential.key"/>
  <property value="cn=mySourceDomain" name="oracle.security.jps.farm.name"/>
  <property value="cn=mySourceRootName" name="oracle.security.jps.ldap.root.name"/>
  <property value="ldap://mySourceHost.com:3060" name="ldap.url"/>
</serviceInstance>

<!-- Source DB-based credential store instance -->
<serviceInstance provider="db.credentialstore.provider"
name="credential.db.source">
  <description>Replace: A. mySourceDomain and mySourceRootName to appropriate
values according to your source DB credential store</description>
  <property value="cn=mySourceDomain" name="oracle.security.jps.farm.name"/>
  <property value="cn=mySourceRootName" name="oracle.security.jps.ldap.root.name"/>
  <property value="jdbc:oracle:thin:@mySourceHost:1722:orcl" name="jdbc.url"/>
  <!-- the value of jdbc.url should be the value entered when the source datasource
was set up -->
  <property value="oracle.jdbc.driver.OracleDriver" name="jdbc.driver"/>
  <property name="bootstrap.security.principal.key" value="mySourceKeyName" />
  <property name="bootstrap.security.principal.map" value="mySourceMapName" />
  <!-- the values of bootstrap.security.principal.key and
bootstrap.security.principal.map
should be the values entered when the bootstrap credential was set up -->
</serviceInstance>

<!-- Destination LDAP-based credential store instance -->
<serviceInstance provider="ldap.credentialstore.provider"
name="credential.ldap.destination">
  <description>Replace: A. myDestDomain and myDestRootName to appropriate values
according to your destination LDAP directory structure; B.
ldap://myDestHost.com:3060 with the URL and port number of your destination
LDAP</description>
  <property value="bootstrap" name="bootstrap.security.credential.key"/>
  <property value="cn=myDestDomain" name="oracle.security.jps.farm.name"/>
  <property value="cn=myDestRootName" name="oracle.security.jps.ldap.root.name"/>
  <property value="ldap://myDestHost.com:3060" name="ldap.url"/>
</serviceInstance>

<!-- Destination DB-based credential store instance -->
<serviceInstance provider="db.credentialstore.provider"
name="credential.db.destination">
  <description>Replace: myDestDomain and myDestRootName to appropriate values
according to your destination DB credential store</description>
  <property value="cn=myDestDomain" name="oracle.security.jps.farm.name"/>
  <property value="cn=myDestRootName" name="oracle.security.jps.ldap.root.name"/>
  <property value="jdbc:oracle:thin:@myDestHost.com:1722:orcl" name="jdbc.url"/>

```



```

    <!-- the value of jdbc.url should be the value entered when the destination
datasource was set up -->
    <property value="oracle.jdbc.driver.OracleDriver" name="jdbc.driver" />
    <property name="bootstrap.security.principal.key" value="myDestKeyName" />
    <property name="bootstrap.security.principal.map" value="myDestMapName" />
    <!-- the values of bootstrap.security.principal.key and
        bootstrap.security.principal.map
        should be the values entered when the bootstrap credential was set up -->
</serviceInstance>

<!-- Bootstrap credentials to access source and destination LDAPs and DBs -->
    <serviceInstance location="./bootstrap" provider="credstoressp"
name="bootstrap.cred">
    <description>Replace location with the full path of the directory where the
bootstrap file cwallet.sso is located; typically found in
destinationDomain/config/fmwconfig/</description>
    </serviceInstance>
</serviceInstances>

    <jpsContexts>
    <jpsContext name="FileSourceContext">
    <serviceInstanceRef ref="credential.file.source" />
    </jpsContext>

    <jpsContext name="LDAPsourceContext">
    <serviceInstanceRef ref="credential.ldap.source" />
    </jpsContext>

    <jpsContext name="DBsourceContext">
    <serviceInstanceRef ref="credential.db.source" />
    </jpsContext>

    <jpsContext name="LDAPdestinationContext">
    <serviceInstanceRef ref="credential.ldap.destination" />
    </jpsContext>

    <jpsContext name="DBdestinationContext">
    <serviceInstanceRef ref="credential.db.destination" />
    </jpsContext>

    <!-- Do not change the name of the next context -->
    <jpsContext name="bootstrap_credstore_context">
    <serviceInstanceRef ref="bootstrap.cred" />
    </jpsContext>
    </jpsContexts>
</jpsConfig>

```

Note that since the migration involves LDAP and/or DB stores, the file includes a `jps-context` named `bootstrap_credstore_context` that specifies the directory where the bootstrap credential file `cwallet.sso` is located.

The following examples of use of `migrateSecurityStore` assume that the file `t2p-credentials.xml` is located on the target system in the directory where the script is run.

Under that assumption, to migrate credentials from a test (or source) LDAP store to a production (or destination) LDAP store, invoke `migrateSecurityStore` in the target system as follows:

```
>migrateSecurityStore(type="credStore", configFile="t2p-credentials.xml", src="LDAPs
sourceContext", dst="LDAPdestinationContext")
```

To migrate credentials from a test (or source) XML store to a production (or destination) LDAP store, invoke `migrateSecurityStore` in the target system as follows:

```
>migrateSecurityStore(type="credStore", configFile="t2p-credentials.xml", src="FileSourceContext", dst="LDAPdestinationContext")
```

To migrate credentials from a test (or source) DB store to a production (or destination) DB store, invoke `migrateSecurityStore` in the target system as follows:

```
>migrateSecurityStore(type="credStore", configFile="t2p-credentials.xml", src="DBSourceContext", dst="DBdestinationContext")
```

6.5.2.3 Migrating Large Volume Policy and Credential Stores

Migrating stores with the alternate procedure explained in this section is suitable to preserve source GUIDs or for large volume stores (where migrating with the script `migrateSecurityStore` would take an unacceptable amount of time).

Note: Large volume migration of stores is supported for LDAP-based stores only. It is not supported for DB-based stores.

For illustration purpose, assume that the policy store LDAP to be migrated is configured in the file `jps-config.xml` with a service instance as in the following fragment:

```
<serviceInstance provider="ldap.policystore.provider" name="policystore.ldap">
  <property name="policystore.type" value="OID" />
  <property name="bootstrap.security.principal" value="bootstrap"/>
  <property name="oracle.security.jps.farm.name" value="cn=base_domain"/>
  <property name="oracle.security.jps.ldap.root.name" value="cn=mySrcRootName"/>
  <property name="ldap.url" value="ldap://myCompany.com:7766"/>
</serviceInstance>
```

Important: If you intend to use the procedure that follows with a destination Oracle Internet Directory version 10.1.4.3.0, then you must first apply a patch for bug number 8417224. To download this patch for your platform, visit Oracle Support at <http://myoraclesupport.oracle.com>.

To migrate a source Oracle Internet Directory store to a destination Oracle Internet Directory store using bulk commands, proceed as follows:

1. In the system where the source Oracle Internet Directory is located, produce an LDIF file by running `ldifwrite` as illustrated in the following line:

```
>ldifwrite connect="srcOidDbConnectStr" baseDN="cn=jpsnode, c=us"
ldiffile="srcOid.ldif"
```

This command writes all entries under the node `cn=jpsnode, c=us` to the file `srcOid.ldif`. Once generated, move this file, as appropriate, to the destination Oracle Internet Directory file system so it is available to the commands that follow.

2. In the destination Oracle Internet Directory node, ensure that the JPS schema has been seeded.

3. In the destination Oracle Internet Directory system, verify that there are no schema errors or bad entries by running `bulkload` as illustrated in the following line:

```
>bulkload connect="dstOidDbConnectStr" check=true generate=true restore=true
file="fullPath2SrcOidLdif"
```

If duplicated DNs (common entries between the source and destination directories) are detected, review them to prevent unexpected results.

4. Backup the destination DB. If the next steps fails (and corrupts the DB), the DB must be restored.
5. Load data into the destination Oracle Internet Directory, by running `bulkload` as illustrated in the following line:

```
>bulkload connect="dstOidDbConnectStr" load=true file="fullPath2SrcOidLdif"
```

For details about the above commands, see chapter 14, *Performing Bulk Operations*, in *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*.

6.5.3 Migrating Audit Policies

To migrate audit policies, use the export and import operations as explained next.

First, export the audit configuration from a test environment to a file using one of the following tools:

- Fusion Middleware Control: navigate to *Domain* > **Security** > **Audit Policy**, and then click **Export**.
- The OPSS script `exportAuditConfig`. For details, see [Appendix C.4.7, "exportAuditConfig."](#)

Then, import that file into the production environment using one of the following tools:

- Fusion Middleware Control: navigate to *Domain* > **Security** > **Audit Policy**, and then click **Import**.
- The OPSS script `importAuditConfig`. For details, see [Appendix C.4.8, "importAuditConfig."](#)

The import/export operations above migrate audit policies only, and they do not migrate the audit data store settings. If you had configured an audit data source in your test environment, repeat the steps to configure a data source in the production environment. For details, see [Section 13.2.2, "Set Up Audit Data Sources."](#)

Normally, you would not want audit data records from a test environment to be migrated to production; however, to do so, use the database import/export utilities for that purpose. For details, see [Section 13.5.5, "Importing and Exporting Data."](#)

6.5.4 Migrating Keystore Service Keys and Certificates

To migrate keys and certificates manually with `migrateSecurityStore`, create a configuration file to specify the source and destination service instances. Next use the `migrateSecurityStore` command with appropriate options as shown in the examples at the end of this section.

Here is a complete example of a configuration file, named `t2p-keys.xml`, illustrating the specification of keystore service sources in LDAP, DB, and XML storages, and of keystore service destinations in LDAP or DB storages:

```

<?xml version="1.0" encoding="UTF-8" standalone='yes'?>
<jpsConfig xmlns="http://xmlns.oracle.com/oracleas/schema/11/jps-config-11_1.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.oracle.com/oracleas/schema/11/jps-config-11_
1.xsd" schema-major-version="11" schema-minor-version="1">

<serviceProviders>

<serviceProvider
class="oracle.security.jps.internal.credstore.ssp.SspCredentialStoreProvider"
name="credstoressp" type="CREDENTIAL_STORE">
  <description>File-based credential provider</description>
</serviceProvider>

<!--The following service provider configuration serves file-based, LDAP based
      And DB based keystore service instance -->
<serviceProvider type="KEY_STORE" name="keystore.provider"
class="oracle.security.jps.internal.keystore.KeyStoreProvider">
<description>PKI Based Keystore Provider</description>
</serviceProvider>

</serviceProviders>

<serviceInstances>

<!-- Source XML-based keystore service instance -->
  <serviceInstance location="." provider="keystore.provider"
name="keystore.file.source">
<property name="keystore.provider.type" value="file"/>
<property name="keystore.file.path" value="."/>
<description>Replace keystore.file.path with the full path of the folder where the
file-based source keystore service keystores.xml is located in the source file
system; typically located in sourceDomain/config/fmwconfig/</description>
  </serviceInstance>

<!-- Source LDAP-based keystore service instance -->
<serviceInstance provider="keystore.provider" name="keystore.ldap.source">
  <description>Replace: A. mySourceDomain and mySourceRootName to appropriate
values according to your source LDAP directory structure; B.
ldap://mySourceHost.com:3060 with the URL and port number of your source
LDAP</description>
  <property value="bootstrap" name="bootstrap.security.credential.key"/>
  <property value="cn=mySourceDomain" name="oracle.security.jps.farm.name"/>
  <property value="cn=mySourceRootName" name="oracle.security.jps.ldap.root.name"/>
  <property value="ldap://mySourceHost.com:3060" name="ldap.url"/>
  <property name="keystore.provider.type" value="ldap"/>
</serviceInstance>

<!-- Source DB-based keystore service instance -->
<serviceInstance provider="keystore.provider" name="keystore.db.source">
  <description>Replace: A. mySourceDomain and mySourceRootName to appropriate
values according to your source DB </description>
  <property value="cn=mySourceDomain" name="oracle.security.jps.farm.name"/>
  <property value="cn=mySourceRootName" name="oracle.security.jps.ldap.root.name"/>
  <property value="jdbc:oracle:thin:@mySourceHost:1722:orcl" name="jdbc.url"/>
  <!-- the value of jdbc.url should be the value entered when the source datasource

```

```

was set up -->
  <property value="oracle.jdbc.driver.OracleDriver" name="jdbc.driver"/>
  <property name="bootstrap.security.principal.key" value="mySourceKeyName" />
  <property name="bootstrap.security.principal.map" value="mySourceMapName" />
  <property name="keystore.provider.type" value="db"/>
  <!-- the values of bootstrap.security.principal.key and
        bootstrap.security.principal.map
        should be the values entered when the bootstrap credential was set up -->
</serviceInstance>

<!-- Destination LDAP-based keystore service instance -->
  <serviceInstance provider="keystore.provider" name="keystore.ldap.destination">
  <description>Replace: A. myDestDomain and myDestRootName to appropriate values
according to your destination LDAP directory structure; B.
ldap://myDestHost.com:3060 with the URL and port number of your destination
LDAP</description>
  <property value="bootstrap" name="bootstrap.security.credential.key"/>
  <property value="cn=myDestDomain" name="oracle.security.jps.farm.name"/>
  <property value="cn=myDestRootName" name="oracle.security.jps.ldap.root.name"/>
  <property value="ldap://myDestHost.com:3060" name="ldap.url"/>
  <property name="keystore.provider.type" value="ldap"/>
</serviceInstance>

<!-- Destination DB-based keystore service instance -->
  <serviceInstance provider="keystore.provider" name="keystore.db.destination">
  <description>Replace: myDestDomain and myDestRootName to appropriate values
according to your destination DB </description>
  <property value="cn=myDestDomain" name="oracle.security.jps.farm.name"/>
  <property value="cn=myDestRootName" name="oracle.security.jps.ldap.root.name"/>
  <property value="jdbc:oracle:thin:@myDestHost.com:1722:orcl" name="jdbc.url"/>
  <!-- the value of jdbc.url should be the value entered when the destination
datasource was set up -->
  <property value="oracle.jdbc.driver.OracleDriver" name="jdbc.driver"/>
  <property name="bootstrap.security.principal.key" value="myDestKeyName" />
  <property name="bootstrap.security.principal.map" value="myDestMapName" />
  <property name="keystore.provider.type" value="db"/>
  <!-- the values of bootstrap.security.principal.key and
        bootstrap.security.principal.map
        should be the values entered when the bootstrap credential was set up -->
</serviceInstance>

<!-- Bootstrap credentials to access source and destination LDAPs and DBs -->

  <serviceInstance location="./bootstrap" provider="credstoressp"
name="bootstrap.cred">
  <description>Replace location with the full path of the directory where the
bootstrap file cwallet.sso is located; typically found in
destinationDomain/config/fmwconfig/bootstrap</description>
  </serviceInstance>
</serviceInstances>

  <jpsContexts>
  <jpsContext name="FileSourceContext">
  <serviceInstanceRef ref="keystore.file.source"/>
  </jpsContext>

  <jpsContext name="LDAPsourceContext">

```

```
<serviceInstanceRef ref="keystore.ldap.source"/>
</jpsContext>

<jpsContext name="DBsourceContext">
  <serviceInstanceRef ref="keystore.db.source"/>
</jpsContext>

<jpsContext name="LDAPdestinationContext">
  <serviceInstanceRef ref="keystore.ldap.destination"/>
</jpsContext>

<jpsContext name="DBdestinationContext">
  <serviceInstanceRef ref="keystore.db.destination"/>
</jpsContext>

<!-- Do not change the name of the next context -->
<jpsContext name="bootstrap_credstore_context">
  <serviceInstanceRef ref="bootstrap.cred"/>
</jpsContext>
</jpsContexts>
</jpsConfig>
```

Note that since the migration involves LDAP and/or DB stores, the file includes a `jps-context` named `bootstrap_credstore_context` that specifies the directory where the bootstrap credential file `cwallet.sso` is located.

Examples

Note: The following `migrateSecurityStore` examples assume that the file `t2p-keys.xml` is located on the target system in the directory where the script is run.

To migrate all keys and certificates from a test (source) LDAP store to a production (destination) LDAP store, invoke `migrateSecurityStore` in the target system as follows:

```
>migrateSecurityStore(type="keyStore", configFile="t2p-keys.xml",
src="LDAPsourceContext", dst="LDAPdestinationContext")
```

To migrate all keys and certificates from a test (source) XML store to a production (destination) LDAP store, invoke `migrateSecurityStore` in the target system as follows:

```
>migrateSecurityStore(type="keyStore", configFile="t2p-keys.xml",
src="FileSourceContext", dst="LDAPdestinationContext")
```

To migrate keys and certificates for a specific application stripe from a test (source) database store to a production (destination) database store, invoke `migrateSecurityStore` in the target system as follows:

```
>migrateSecurityStore(type="stripeKeyStore", configFile="t2p-keys.xml",
src="DBSourceContext", dst="DBdestinationContext", srcStripe="application1",
dstStripe="application2")
```

Part III

Advanced OPSS Administration

This part describes advanced OPSS administration features in the following chapters:

- [Chapter 7, "Configuring the Identity Store Service"](#)
- [Chapter 8, "Configuring the OPSS Security Store"](#)
- [Chapter 9, "Managing the Policy Store"](#)
- [Chapter 10, "Managing the Credential Store"](#)
- [Chapter 11, "Managing Keys and Certificates with the Keystore Service"](#)
- [Chapter 12, "Introduction to Oracle Fusion Middleware Audit Framework"](#)
- [Chapter 13, "Configuring and Managing Auditing"](#)
- [Chapter 14, "Using Audit Analysis and Reporting"](#)

Configuring the Identity Store Service

This chapter explains how to use the identity store service in OPSS. Topics include:

- [Introduction to the Identity Store Service](#)
- [Configuring the Identity Store Provider](#)
- [Configuring the Identity Store Service](#)
- [Querying the Identity Store Programmatically](#)
- [SSL for the Identity Store Service](#)

7.1 Introduction to the Identity Store Service

This section describes key concepts of the OPSS identity store service:

- [About the Identity Store Service](#)
- [Service Architecture](#)
- [Application Server Support](#)

7.1.1 About the Identity Store Service

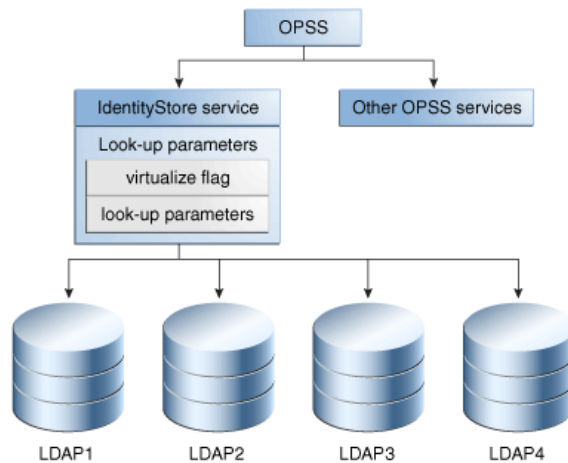
The identity store service enables you to query the identity store for user and role (group) information.

By default, a service instance supports querying against a single LDAP identity store. You can configure the service to support a virtualized identity store which queries multiple LDAP identity stores. This feature, known as identity virtualization, is described in [Section 7.3, "Configuring the Identity Store Service"](#).

7.1.2 Service Architecture

[Figure 7-1](#) shows the architecture of the identity store service. Depending on the configuration, the service can support either an XML file or one or more LDAP servers as the identity store.

When the service is configured for LDAP, it queries a single LDAP store by default. You can also configure the service to query multiple LDAP stores.

Figure 7–1 The OPSS Identity Store Service

7.1.3 Application Server Support

The identity store service supports:

- Oracle WebLogic Server
- Third-party application servers

The service configuration depends on the application server; you must specify the provider that supports the service.

7.1.4 Java SE Support

The identity store service is available in a stand-alone Java SE environment.

For more information, see [Section 7.3.6, "Java SE Environments"](#).

7.2 Configuring the Identity Store Provider

Before you can make use of the identity store service, you need to configure the identity store provider. OPSS support both XML- and LDAP-based providers.

This fragment from the `jps-config.xml` file shows the configuration of both XML and LDAP providers. The `serviceProvider` elements are children of the `serviceProviders` element.

```

<serviceProvider type="IDENTITY_STORE" name="idstore.ldap.provider"
  class="oracle.security.jps.internal.idstore.ldap.LdapIdentityStoreProvider">
  <description>LDAP-based IdentityStore Provider</description>
</serviceProvider>

<serviceProvider type="IDENTITY_STORE" name="idstore.xml.provider"
  class="oracle.security.jps.internal.idstore.xml.XmlIdentityStoreProvider">
  <description>XML-based IdentityStore Provider</description>
</serviceProvider>
  
```

For details, see [Section 8.7.1, "Configuring the Identity Store Provider"](#).

7.3 Configuring the Identity Store Service

This section describes how to configure the identity store service to LDAP-based stores. Topics include:

- [What is Configured?](#)
- [Configuration in WebLogic Server](#)
- [Configuring Split Profiles](#)
- [Configuring Custom Authenticators](#)
- [Configuration in Other Application Servers](#)
- [Java SE Environments](#)

See Also: [Appendix F, "OPSS System and Configuration Properties"](#).

7.3.1 What is Configured?

This section explains the different configuration parameters for the identity store service. It includes:

- [Configuring Multi-LDAP Lookup](#)
- [Global/Connection Parameters](#)
- [Back-End/Connection Parameters](#)

7.3.1.1 Configuring Multi-LDAP Lookup

You use the following parameters to configure the service for multi-LDAP queries:

- The `virtualize` property - This property can be either `true` (multi-LDAP lookup) or `false` (single-LDAP lookup). The default is `false`.
- Global Connection Parameters (if `virtualize` is enabled) - The calling application uses these parameters to specify global LDAP configuration such as the search base, create base, and so on. If any of these parameters are not configured, OPSS uses default values.
- Back-end Connection Parameters - These parameters are specific to each LDAP store. One set of back-end parameters is specified for each LDAP. You do not need to set these parameters unless you wish to overwrite existing values.

7.3.1.2 Global/Connection Parameters

[Table 7–1](#) shows the global parameters and their default values, if applicable.

Table 7–1 Global LDAP Identity Store Parameters

Parameter	Default Value
<code>group.create.bases</code>	same as <code>user.create.bases</code>
<code>group.filter.object.classes</code>	<code>groupofuniquenames</code> The global value is used if explicitly provided.
<code>group.mandatory.attrs</code>	-
<code>group.member.attrs</code>	<code>uniquemember</code>
<code>group.object.classes</code>	<code>groupofuniquenames</code>
<code>group.search.bases</code>	-

Table 7–1 (Cont.) Global LDAP Identity Store Parameters

Parameter	Default Value
group.selected.create.base	-
group.selected.search.base	-
groupname.attr	cn If the global value is explicitly given, it is used.
max.search.filter.length	-
search.type	-
user.create.bases	If only one authenticator, uses it as the create base value. If multiple authenticators, no default value is set; user must explicitly set the global value.
user.filter.object.classes	inetorgperson
user.login.attr	uid
user.mandatory.attrs	-
user.object.classes	inetorgperson If the global value is explicitly given, it is used.
user.search.bases	Same as group.search.bases
username.attr	cn The global value is used if explicitly provided.

See Also: [Section F–6, "Generic LDAP Properties"](#)

7.3.1.3 Back-End/Connection Parameters

As mentioned earlier, these are specific to the back-end LDAP store. For details, see:

- [Table F–5, "LDAP-Based Identity Store Properties"](#)
- [Section F.2.1, "Policy Store Properties"](#)

7.3.2 Configuration in WebLogic Server

You configure LDAP authenticators in Oracle WebLogic Server using either the WebLogic console or WLST command-line. At runtime, Oracle WebLogic Server passes the configuration details to OPSS. Oracle WebLogic Server allows you to configure multiple authenticators in a given context, and selects the first authenticator to initialize the identity store service by default. This process is explained in [Section 3.1.2.2, "Configuring the LDAP Identity Store Service"](#).

After the authenticators are configured, the identity store service can be set up to query one LDAP identity store or multiple stores. Configuring for multiple stores requires setting up the `virtualize` property.

This section explains how to set up these options.

7.3.2.1 Configuring the Service for Single LDAP

You can configure the identity store service to query only one LDAP store. See [Example 7-1](#) which displays a fragment of the `jps-config.xml` file with a single LDAP service instance.

7.3.2.2 Configuring the Service for Multiple LDAP using Fusion Middleware Control

As in the single LDAP setup, you start by configuring the authentication providers in Oracle WebLogic Server.

Next, take these steps in Fusion Middleware Control:

1. Select the WebLogic domain in the navigation pane on the left.
2. Navigate to Security, then Security Provider Configuration.
3. Expand the Identity Store Provider section of the page.
4. Click **Configure** (corresponding to "Configure parameters for User and Role APIs to interact with identity store").
5. The Identity Store Configuration page appears.
6. Under Custom Properties, click **Add**.
7. Add the new property as follows:

```
Property Name=virtualize
Value=true
```

Note: Be sure to add the property to the identity store service instance in the default context.

8. Click **OK**.

7.3.2.3 Configuring the Service for Multiple LDAP using WLST

To configure the virtualize property using WLST, take these steps:

1. Create a `py` script file to connect to the administration server in the domain of interest. You need to specify the `userName`, `userPass`, `localHost`, and `portNumber` for the operation.

See [Appendix E.1, "Configuring OPSS Service Provider Instances with a WLST Script"](#) for details about this script.

2. Navigate to `$ORACLE_HOME/common/bin`.
3. Run the `wlst.sh` command to execute the script.

For example, if the domain configuration file contains an authenticator named `idstore.ldap`, the following command:

```
wlst.sh /tmp/updateServiceInstanceProperty.py -si idstore.ldap
-key "virtualize" -value "true"
```

configures the provider for multi-LDAP lookup.

See Also: [Section E.1, "Configuring OPSS Service Provider Instances with a WLST Script"](#).

7.3.2.4 Configuring Other Parameters

If desired, you can update `jps-config.xml` to set query parameters listed in [Section 7.3.1, "What is Configured?"](#). These parameters are optional; default values are provided.

7.3.2.5 Restarting Servers

After configuring for multi-LDAP query, restart Weblogic admin and managed servers.

7.3.2.6 Examples of the Configuration File

[Example 7-1](#) shows a sample `jps-config.xml` file configured for single-LDAP queries in the Oracle WebLogic Server environment:

Example 7-1 Single-LDAP Configuration in Oracle WebLogic Server

```
<!-- JPS WLS LDAP Identity Store Service Instance -->
  <serviceInstance name=idstore.ldap provider=idstore.ldap.provider>
    <property name=idstore.config.provider
      value=oracle.security.jps.wls.internal.idstore.
      WlsLdapIdStoreConfigProvider/>
    <property name=CONNECTION_POOL_CLASS
      value=oracle.security.idm.providers.stdldap.JNDIPool/>
  </serviceInstance>
```

[Example 7-2](#) shows a sample `jps-config.xml` file configured for multi-LDAP queries in the Oracle WebLogic Server environment:

Example 7-2 Multi-LDAP Configuration in Oracle WebLogic Server

```
<jpsConfig xmlns="http://xmlns.oracle.com/oracleas/schema/11/jps-config-11_1.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.oracle.com/oracleas/schema/11/jps-config-11_
1.xsd" schema-major-version="11" schema-minor-version="1">

  <serviceProviders>
    <serviceProvider type="IDENTITY_STORE" name="idstore.ldap.provider"
class="oracle.security.jps.internal.idstore.ldap.LdapIdentityStoreProvider">
      <description>LDAP-based IdentityStore Provider</description>
    </serviceProvider>
  </serviceProviders>

  <serviceInstances>
    <!-- IDstore instance connecting to multiple ldap -->
    <serviceInstance name="idstore.virtualize"
provider="idstore.ldap.provider">

      <!-- following property indicates using WLS ldap Authenticators -->
      <property name="idstore.config.provider"

value="oracle.security.jps.wls.internal.idstore.WlsLdapIdStoreConfigProvider"/>

      <!-- following property enables virtualization i.e., support for multiple
stores -->
      <property name="virtualize" value="true"/>

      <!-- Front end ldap properties (if not supplied, will use default
values) -->
```

```

    <extendedProperty>
      <name>user.create.bases</name>
      <values>
        <value>cn=users_front,dc=us,dc=oracle,dc=com</value>
      </values>
    </extendedProperty>
    <extendedProperty>
      <name>group.create.bases</name>
      <values>
        <value>cn=groups_front,dc=us,dc=oracle,dc=com</value>
      </values>
    </extendedProperty>
  </serviceInstance>
</serviceInstances>

<jpsContexts default="default">

  <!-- the identity store uses multiple ldaps -->
  <jpsContext name="default">
    <!-- use multiple ldap -->
    <serviceInstanceRef ref="idstore.virtualize"/>
    <!-- .....other services -->
  </jpsContext>
</jpsContexts>

</jpsConfig>

```

Note that:

- the `virtualize` property of the service instance is set to `true`, enabling multi-LDAP queries.
- the `extendedProperty` element enables you to set front-end parameters if desired to override default values.

For more information, see "Front-End Parameters" in [Section 7.3.1, "What is Configured?"](#).

7.3.3 Configuring Split Profiles

Identity Virtualization supports a "split profile," where an application makes use of attributes for a single identity that are stored on two different sources.

This feature requires additional configuration beyond that described in this chapter. For details, see [Appendix K, "Adapter Configuration for Identity Virtualization"](#).

7.3.4 Configuring Custom Authenticators

OPSS supports the set of LDAP-based Oracle WebLogic Server authentication providers (WebLogic authenticators) for access to identity stores. If the out-of-the-box WebLogic authenticators are not applicable to your LDAP server type, you can customize a generic authenticator for this task.

This section explains how you can configure such an authenticator when the 'virtualize' flag is enabled for the identity store service.

Note the following points in this context:

- When using a generic LDAP authenticator, you need to tell the Identity Store Service of the exact LDAP type so that it can find the proper LDAP plug-in. You do this by overriding the `'idstore.type'` property in `jps-config.xml`.

- As the 'virtualize' flag is enabled, and the Oracle WebLogic Server domain has two or more authenticators (for example, the defaultAuthenticator and genericLDAP), you need to tell the Identity Store Service which LDAP server's 'idstore.type' is to be overridden.

You provide this information as follows in `jps-config.xml`:

```
<serviceInstance name="idstore.ldap" provider="idstore.ldap.provider">
  <property name="idstore.config.provider"
value="oracle.security.jps.wls.internal.idstore.WlsLdapIdStoreConfigProvider"
/>
  <property name="CONNECTION_POOL_CLASS"
value="oracle.security.idm.providers.stdlldap.JNDIPool" />

  <property value="true" name="virtualize" />

  <!-- the following refer to the Generic ldap name configured on the WLS
authenticator provider, you get this name from WebLogic config.xml file or admin
console -->
  <serviceInstanceRef ref="myGenericLDAPName"/>

</serviceInstance>

<!-- the following provide the overriding to the Generic ldap (e.g. AD)
-->
<!-- "myGenericLDAPName" is the name used on WLS for the generic LDAP
authenticator provider -->
<serviceInstance name="myGenericLDAPName" provider="idstore.ldap.provider">

  <!-- the following overrides the 'idstore.type' property to
"ACTIVE_DIRECTORY" -->
  <property name="idstore.type" value="ACTIVE_DIRECTORY" />
</serviceInstance>
```

If you need to override an additional LDAP provider instance, simply add another similar entry to the file.

7.3.5 Configuration in Other Application Servers

Topics in this section include:

- [Configuring the Service for Single LDAP](#)
- [Configuring the Service for Multiple LDAP](#)

7.3.5.1 Configuring the Service for Single LDAP

See the example in [Section 22.2.2, "Configuring an LDAP Identity Store in Java SE Applications,"](#) for details.

7.3.5.2 Configuring the Service for Multiple LDAP

To configure the identity store service to handle multiple LDAPs in third-party application servers:

1. Modify the `jps-config.xml` file to configure service instances for each supported LDAP directory
2. Restart the application server to make the changes effective.

Example 7-3 shows a sample `jps-config.xml` file configured to run multi-LDAP queries for third-party application servers:

Example 7-3 Multi-LDAP Configuration in Third-Party Application Servers

```
<jpsConfig xmlns="http://xmlns.oracle.com/oracleas/schema/11/jps-config-11_1.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.oracle.com/oracleas/schema/11/jps-config-11_1.xsd" schema-major-version="11" schema-minor-version="1">

  <serviceProviders>
    <serviceProvider type="IDENTITY_STORE" name="idstore.ldap.provider"
class="oracle.security.jps.internal.idstore.ldap.LdapIdentityStoreProvider">
      <description>LDAP-based IdentityStore Provider</description>
    </serviceProvider>
  </serviceProviders>

  <serviceInstances>
    <!-- instance 'idstore.oid' to represent an ldap server 'oid' -->
    <serviceInstance name="idstore.oid" provider="idstore.ldap.provider">
      <property name="subscriber.name" value="dc=us,dc=oracle,dc=com"/>
      <property name="idstore.type" value="OID"/>
      <property name="security.principal.key" value="oid.ldap.credentials"/>
      <property name="security.principal.alias" value="JPS"/>
      <property name="ldap.url"
value="ldap://oid1.us.oracle.com:389,ldap://oid2.us.oracle.com:389"/>
      <extendedProperty>
        <name>user.search.bases</name>
        <values>
          <value>cn=users,dc=us,dc=oracle,dc=com</value>
        </values>
      </extendedProperty>
      <extendedProperty>
        <name>group.search.bases</name>
        <values>
          <value>cn=groups,dc=us,dc=oracle,dc=com</value>
        </values>
      </extendedProperty>
      <extendedProperty>
        <name>username.attr</name>
        <values>
          <value>uid</value>
        </values>
      </extendedProperty>
      <extendedProperty>
        <name>groupname.attr</name>
        <values>
          <value>cn</value>
        </values>
      </extendedProperty>
    </serviceInstance>

    <!-- instance 'idstore.ad' to represent an ldap server 'ad' -->
    <serviceInstance name="idstore.ad" provider="idstore.ldap.provider">
      <property name="subscriber.name" value="dc=us,dc=oracle,dc=com"/>
      <property name="idstore.type" value="ACTIVE_DIRECTORY"/>
      <property name="security.principal.key"
value="msad.ldap.credentials"/>
      <property name="security.principal.alias" value="JPS"/>
```

```

        <property name="ldap.url"
value="ldap://msad1.us.oracle.com:389,ldap://msad2.us.oracle.com:389"/>
        <extendedProperty>
            <name>user.search.bases</name>
            <values>
                <value>cn=users,dc=us,dc=oracle,dc=com</value>
            </values>
        </extendedProperty>
        <extendedProperty>
            <name>group.search.bases</name>
            <values>
                <value>cn=groups,dc=us,dc=oracle,dc=com</value>
            </values>
        </extendedProperty>
        <extendedProperty>
            <name>username.attr</name>
            <values>
                <value>cn</value>
            </values>
        </extendedProperty>
        <extendedProperty>
            <name>groupname.attr</name>
            <values>
                <value>cn</value>
            </values>
        </extendedProperty>
    </serviceInstance>

    <!-- IDStore service "idservice.virtualize" to connect to multiple ldaps
( 'oid' and 'ad') using libOVD-->
    <serviceInstance name="idservice.virtualize"
        provider="idstore.ldap.provider">

        <!--following property enables virtualization i.e., support for multiple
stores -->
        <property name="virtualize" value="true"/>
        <!-- backend ldap instance "idstore.oid"-->
        <serviceInstanceRef ref="idstore.oid"/>
        <!-- backend ldap instance "idstore.ad"-->
        <serviceInstanceRef ref="idstore.ad"/>
        <!-- Front end ldap properties (if not supplied, will use default
values) -->
        <extendedProperty>
            <name>user.create.bases</name>
            <values>
                <value>cn=users_front,dc=us,dc=oracle,dc=com</value>
            </values>
        </extendedProperty>
        <extendedProperty>
            <name>group.create.bases</name>
            <values>
                <value>cn=groups_front,dc=us,dc=oracle,dc=com</value>
            </values>
        </extendedProperty>
    </serviceInstance>
</serviceInstances>

<jpsContexts default="default">

    <!-- IdStore service connect to multiple ldaps ('oid'+ 'ad') through

```

```

libOVD-->
    <jpsContext name="default">
<!-- use multiple ldaps ('oid'+ad') through libOVD-->
        <serviceInstanceRef ref="idservice.virtualize"/>
        <!-- .....other services -->
    </jpsContext>

</jpsContexts>

</jpsConfig>

```

Note that:

- the first service instance defines the provider for Oracle Internet Directory.
- the second service instance defines the provider for Microsoft Active Directory.
- the `virtualize` property of the service instance is set to `true`, enabling multi-LDAP queries.
- the `extendedProperty` elements enable you to set front-end parameters if desired to override default values.

For more information, see "Front-End Parameters" in [Section 7.3.1, "What is Configured?"](#).

7.3.6 Java SE Environments

In the Java SE environment, you directly modify the `jps-config.xml` file as follows:

1. Define a new identity store service instance.
2. Add the new service instance to the JPS context, replacing any previously defined `IdentityStore` instance.
3. Refer to [Example 7-3](#) to enable the 'virtualize' flag in the identity store service.

See [Section 22.2.2, "Configuring an LDAP Identity Store in Java SE Applications"](#) for details.

7.4 Querying the Identity Store Programmatically

To programmatically query the LDAP identity store, you use OPSS to obtain the JPS context; this acts like a bridge to obtain the store instance. Subsequently you use the User and Role API to query the store.

Example 7-4 Querying the LDAP Identity Store Programmatically

```

try {
    //find the JPS context
    JpsContextFactory ctxFactory = JpsContextFactory.getContextFactory();
    JpsContext ctx = ctxFactory.getContext();

    //find the JPS IdentityStore service instance
    //(assuming the backend is ldap type)
    LdapIdentityStore idstoreService =
    (LdapIdentityStore)ctx.getServiceInstance(IdentityStoreService.class)

    //get the User/Role API's Idmstore instance
    oracle.security.idm.IdentityStore idmIdentityStore =
    idstoreService.getIdmStore();

```

```
//use the User/Role API to query id store
//

//alternatively, instead of using IdentityStore, use the
//IdentityDirectory to access LDAP
oracle.igf.ids.IdentityDirectory ids = idstoreService.getIdentityStore();
// ref. chapter "Developing with the Identity Directory API"
// on how to use IdentityDirectory

} catch (Exception e) {
    e.printStackTrace()
}
```

To see how to enable the 'virtualize' property in the identity store service, refer to [Example 7-3](#).

For additional information about using MBeans, see [Section E.2, "Configuring OPSS Services with MBeans"](#).

7.5 SSL for the Identity Store Service

Connections between the identity store and an LDAP server can be SSL-enabled. This section explains how the connections are configured in the various scenarios.

- [Connections from Oracle WebLogic Server to Identity Store](#)
- [One-way SSL in a Multi-LDAP Scenario](#)
- [Two-way SSL in a Multi-LDAP Scenario](#)
- [Connections in a Single-LDAP Scenario](#)

7.5.1 Connections from Oracle WebLogic Server to Identity Store

When the connection to the identity store originates at a client residing in Oracle WebLogic Server, SSL configuration is handled by Oracle WebLogic Server. For details, see [Section 8.2.3](#).

7.5.2 One-way SSL in a Multi-LDAP Scenario

Both the Identity Directory API and the User and Role API can operate in a multi-LDAP identity store configuration (`virtualize = true`).

In this scenario, you can SSL-enable the connection from the identity store to the LDAP servers.

The procedure is as follows:

1. Create a keystore to contain the LDAP server certificate(s) for use by the service. You will need to provide passwords for the WebLogic Admin Server and the keystore, respectively.

Create the keystore using the script `$MW_HOME/oracle_common/bin/libovdconfig.sh` with the `"-createKeyStore"` option:

```
libovdconfig.sh -host wls_host -port wls_adminserver_port -userName
wls_user_name -domainPath full_path_domain_home -createKeyStore
```

where:

- `host` is the Oracle WebLogic Server host

- `port` is the Oracle WebLogic Server Admin Server port
 - `username` is the Oracle WebLogic Server admin user name
 - `domainPath` is the complete path to the domain home
2. Export the certificate from the LDAP directory using the appropriate export command.
 3. Import this certificate into the keystore you created in Step 1.

Import the certificate to the keystore using the `keytool` command. The syntax is as follows, for a keystore named `adapters.jks`:

```
$JAVA_HOME/bin/keytool -importcert
-keystore $DOMAIN_HOME/config/fmwconfig/ovd/default/keystores/adapters.jks
-storepass keystore_password_used_in_libovdconfig.sh
-alias alias_name
-file full_path_to_LDAPCert_file
-noprompt
```

4. Restart Oracle WebLogic Server.

7.5.3 Two-way SSL in a Multi-LDAP Scenario

To implement two-way SSL in a multi-LDAP identity store configuration, take these steps:

1. Perform the procedure described in [Section 7.5.2](#).
2. In the keystore that was created by Step 1 of [Section 7.5.2](#), generate a new key pair, signed by a CA.
3. Export this certificate to a file.
4. Import the certificate into the server's keystore.

7.5.4 Connections in a Single-LDAP Scenario

Both the Identity Directory API and the User and Role API can operate in a single-LDAP identity store configuration (`virtualize = false`).

For this scenario, SSL between the identity store and the LDAP server is configured with the same basic steps outlined in [Section 7.5.2](#). However, there is no need to create a keystore using `libovdconfig.sh`. Instead, the trusted certificate must be imported into the application server's trust-store.

For example:

```
keytool -import -v -trustcacerts -alias mytrust -file oidServerTrust.cert
-keystore myTrustStore.jks -storepass trustStorePassword
```

Configuring the OPSS Security Store

The OPSS security store is the repository of system and application-specific policies, credentials, and keys. For an introduction to policies, credentials, keys and certificates, see the following sections:

- [Section 3.2, "Policy Store Basics"](#)
- [Section 3.3, "Credential Store Basics"](#)
- [Section 3.4, "Keystore Service Basics"](#)

This chapter explains the features of the OPSS security store common to policies, credentials, and keys, and it is divided into the following sections:

- [Introduction to the OPSS Security Store](#)
- [Using an LDAP-Based OPSS Security Store](#)
- [Using a DB-Based OPSS Security Store](#)
- [Configuring the OPSS Security Store](#)
- [Reassociating the OPSS Security Store](#)
- [Migrating the OPSS Security Store](#)
- [Configuring the Identity Provider, Property Sets, and SSO](#)

For details about Java EE and WebLogic Security, see section Java EE and WebLogic Security in *Oracle Fusion Middleware Understanding Security for Oracle WebLogic Server*.

Note: When a WebLogic domain is setup to use policies based on the OPSS security store, JACC policies and the Java Security Manager become unavailable on all managed servers in that domain.

Important: All permission classes used in policies in the OPSS security store must be included in the class path, so the policy provider can load them when a service instance is initialized.

8.1 Introduction to the OPSS Security Store

The OPSS security store is the repository of system and application-specific policies, credentials, and keys. This centralization facilitates the administration and maintenance of policy, credential, and key data.

The OPSS security store can be file-, LDAP-, or DB-based depending on the choice of repository type, and it can be reassociated (that is, the repository type can be changed) from file-based to LDAP- or DB-based; from DB-based to LDAP- or DB-based; and from LDAP-based to LDAP- or DB-based. No other reassociation is supported. For details about the tools and procedures available to reassociate the OPSS security store, see sections [Reassociating with Fusion Middleware Control](#) and [Reassociating with the Script `reassociateSecurityStore`](#). Out-of-the-box, the OPSS security store is file-based.

The security data relevant to a Java EE application is typically packaged with the application and it can be migrated at deploy time to the OPSS security store. For details about the tools and procedures available to migrate to the OPSS security store, see sections [Migrating with Fusion Middleware Control](#) and [Migrating with the Script `migrateSecurityStore`](#).

8.2 Using an LDAP-Based OPSS Security Store

An LDAP-based policy store is typically used in production environments. The only LDAP server supported in this release is the Oracle Internet Directory (release 10.1.4.3 or later).

Note: Depending on the version, the following patches to Oracle Internet Directory are required:

- Patch to fix bug 9093298 in Oracle Internet Directory 10.1.4.
- Patch to fix bug 8736355 in Oracle Internet Directory 11.1.x
- Patch to fix bug 8426457 in Oracle Internet Directory 11.1.x and 10.1.4.3.0
- Patch to fix bug 8351672 in Oracle Internet Directory 10.1.4.3.0

To apply a patch, proceed as follows:

1. Visit Oracle Automated Release Updates.
 2. Click the **Patches** tab.
 3. Enter the bug number in the **Request Number** box, and click **Search**.
 4. Apply the patch.
-
-

To use a domain LDAP-based OPSS security store the domain administrator must configure it, as appropriate, using Oracle Enterprise Manager Fusion Middleware Control or OPSS scripts.

Important: OPSS does *not* support enabling referential integrity on Oracle Internet Directory servers. The server will not work as expected if referential integrity is enabled.

To disable a server's referential integrity, use Oracle Enterprise Manager Fusion Middleware Control as follows:

1. Select **Administration**, then **Shared Properties** from the Oracle Internet Directory menu, and then select **General**.
 2. Select **Disabled** from the Enable referential Integrity list.
-
-

For a list of properties that can be specified in a service instance, see [Appendix F.2.4, "Properties Common to All LDAP-Based Instances."](#)

The information in this section is divided into the following topics:

- [Multiple-Node Server Environments](#)
- [Prerequisites to Using an LDAP-Based Security Store](#)

8.2.1 Multiple-Node Server Environments

In domains where several server instances are distributed across multiple machines, it is highly recommended that the OPSS security store be LDAP- or DB-based.

Typically, applications do not change policy, credential, or key data. When they do, however, it is crucial that these changes be correctly propagated to all managed servers and clusters in a domain and, therefore, it is recommended that any such changes be performed in the domain administration server (and not in managed servers).

In a single-node server domain, the propagation of local changes to security data is irrelevant: in this scenario, local changes are equivalent to global changes.

In a multiple-node server domain, however, the JMX framework propagates local changes to a file-based policy to each runtime environment, so that the data is refreshed based on caching policies and configuration. For details about properties you can set on policies and credentials, see sections [Appendix F.2.1, "Policy Store Properties,"](#) and [Appendix F.2.2, "Credential Store Properties."](#)

In a multiple-node server environment, it is highly recommended that both the policy and credential stores be centralized in a LDAP- or DB-based store and configured in the administration server.

8.2.2 Prerequisites to Using an LDAP-Based Security Store

The only supported LDAP-based OPSS security store is Oracle Internet Directory. In order to ensure the proper access to the Oracle Internet Directory, you must set a node in the server directory as explained below.

Fusion Middleware Control automatically provides bootstrap credentials in the file `cwallet.sso` when that tool is used to reassociate to an LDAP-based repository. To specify these required credentials manually, see section [Section 21.4.7, "Specifying Bootstrap Credentials Manually."](#)

Setting a Node in an Oracle Internet Directory Server

The following procedure is carried out by an Oracle Internet Directory administrator.

To set a node in the LDAP Oracle Internet Directory directory, proceed as follows:

1. Create an LDIF file (assumed `jpstestnode.ldif`, for illustration purpose) specifying the following DN and CN entries:

```
dn: cn=jpsroot
cn: jpsroot
objectclass: top
objectclass: OrclContainer
```

The distinguished name of the root node (illustrated by the string `jpsroot` above) must be distinct from any other distinguished name. Some LDAP servers enforce case sensitivity by default. One root node can be shared by multiple WebLogic domains. It is not required that this node be created at the top level, as long as read and write access to the subtree is granted to the Oracle Internet Directory administrator.

2. Import this data into the LDAP server using the command `ldapadd`, as illustrated in the following example (there should be no line break in the command invocation):

```
>ldapadd -h ldap_host -p ldap_port -D cn=orcladmin -w password -v -f
jptestnode.ldif
```

3. Verify that the node has been successfully inserted using the command `ldapsearch`, as illustrated in the following example (there should be no line break in the command invocation):

```
>ldapsearch -h ldap_host -p ldap_port -D cn=orcladmin -w password -s base
-b "cn=jpsroot" objectclass="orclContainer"
```

4. Run the utility `oidstats.sql` to generate database statistics for optimal database performance, as illustrated in the following example:

```
>$ORACLE_HOME/ldap/admin/oidstats.sql
```

The above utility must be run just once after the initial provisioning. For details about this utility, consult the *Oracle Fusion Middleware User Reference for Oracle Identity Management*.

To reassociate a policy store, see [Reassociating the OPSS Security Store](#).

8.2.3 Setting Up a One-Way SSL Connection to the LDAP

This section describes how to set up a one-way SSL channel between Oracle WebLogic server or a Java SE application and the LDAP Oracle Internet Directory. Such connection may be required, for example, when reassociating to an LDAP-based target store.

Prerequisite: Configuring the Oracle Internet Directory Server

To configure the Oracle Internet Directory server to listen in one-way SSL mode, see section Enabling SSL on Oracle Internet Directory Listeners in *Oracle Fusion Middleware Administrator's Guide*.

Exporting Oracle Internet Directory's Certificate Authority (CA)

The use of `orapki` to create a certificate is needed *only if* the CA is unknown to the Oracle WebLogic server.

The following sample illustrates the use of this command to create the certificate `serverTrust.cert`:

```
>orapki wallet export -wallet CA -dn "CN=myCA" -cert serverTrust.cert
```

The above invocation prompts the user to enter the keystore password.

Before You Begin

Before configuring SSL, note that:

- The following procedures are required if the type of SSL being established is `server-auth`, and they are not required in any other case (`no-auth` or `client-auth`).
- If the flags specified in the procedures below are used in a multi-application environment, then the trust store must be shared by all those applications.

Setting Up the WebLogic Server in Case of a Java EE Application

The difference in the following procedures is because the identity store service and the policy store service use different socket factories.

To establish a one-way SSL connection between the server and the identity store, proceed as follows (if applicable, the trust CA is assumed exported):

1. If the CA is known to the Oracle WebLogic server, skip this step; otherwise, use the utility `keytool` to import the Oracle Internet Directory's CA into the WebLogic truststore.

The following invocation, which outputs the file `myKeys.jks`, illustrates the use of this command to import the file `serverTrust.cert`:

```
>keytool -import -v -trustcacerts -alias trust -file serverTrust.cert -keystore myKeys.jks -storepass keyStorePassword
```

2. Modify the script (typically `startWebLogic.sh`) that starts the server to include a line like the following, and then restart the server:

```
-Djavax.net.ssl.trustStore=<absolute path name to file myKeys.jks>
```

To establish a one-way SSL connection between the server and the policy store, proceed as follows (if applicable, the trust CA is assumed exported):

1. Use the utility `keytool` to import trust CA to the trust key store, as illustrated in the following invocation:

```
>keytool -import -v -trustcacerts -alias trust -file serverTrust.cert -keystore myKeys.jks -storepass keyStorePassword
```

2. Modify the script (typically `startWebLogic.sh`) that starts the server to include a line like the following, and then restart the server:

```
-Dweblogic.security.SSL.trustedCAKeyStore=<absolute path name to file myKeys.jks>
```

3. If the OID server uses a wild card in the SSL certificate, then add the following line to the script that starts the WebLogic server:

```
-Dweblogic.security.SSL.ignoreHostnameVerification=true
```

Setting Up the WebLogic Server in Case of a Java SE Application

The setting up in the case of Java SE applications is the same for both the identity and the policy store services.

1. If the CA is known to the Oracle WebLogic server, skip this step; otherwise, use the utility `keytool` to import the Oracle Internet Directory's CA into the WebLogic truststore.

The following invocation, which outputs the file `myKeys.jks`, illustrates the use of this command to import the file `serverTrust.cert`:

```
>keytool -import -v -trustcacerts -alias trust -file serverTrust.cert -keystore myKeys.jks -storepass keyStorePassword
```

2. Modify the script that starts the JMV to include a line like the following:

```
-Djavax.net.ssl.trustStore=<absolute path name to file myKeys.jks>
```

8.3 Using a DB-Based OPSS Security Store

A DB-based security store is typically used in production environments. The only supported DB-based security store is Oracle RDBMS (releases 10.2.0.4 or later; releases 11.1.0.7 or later; and releases 11.2.0.1 or later).

To use a DB-based OPSS security store the domain administrator must configure it, as appropriate, using Oracle Enterprise Manager Fusion Middleware Control or OPSS scripts. In case any checks are needed before the server completes its initialization, see [Section L.14, "Permission Failure Before Server Starts."](#)

For a list of properties that can be configured, see [Appendix F.2, "OPSS Configuration Properties."](#)

This section contains the following topics:

- [Prerequisites to Using a DB-Based Security Store](#)
- [Maintaining a DB-Based Security Store](#)
- [Setting Up an SSL Connection to the DB](#)

8.3.1 Prerequisites to Using a DB-Based Security Store

To use a database repository for the OPSS security store, one must first use Oracle Fusion Middleware Repository Creation Utility (RCU) to create the required schema and to seed some initial data. This setup is also required before reassociating the OPSS security store to a DB-based security store.

For details about RCU, see chapters [Repository Creation Utility Overview](#) and [Running Repository Creation Utility](#) in *Oracle Fusion Middleware Repository Creation Utility User's Guide*.

The creation the schema and seeding of initial data are explained in the following sections:

- [Creating the OPSS Schema in an Oracle Database](#)
- [Dropping the OPSS Schema in an Oracle Database](#)
- [Creating a Data Source Instance](#)

8.3.1.1 Creating the OPSS Schema in an Oracle Database

To create the OPSS schema in an Oracle database with RCU, proceed as follows:

1. Start RCU to display the RCU **Welcome** page; in this page, click **Next** to display the **Create Repository** page.
2. In that page, select the radio button **Create**; then click **Next** to display the **Database Connections Details** page.
3. In that page, enter the appropriate connectivity information: Database Type, Host Name, Port, Service Name, Username, Password, and Role.

Then click **Next** to have RCU check the entered data and perform pre-creation operations; once this check is successfully completed, RCU displays the **Select Components** dialog.

4. In that dialog, choose to use an existing schema prefix or create a new prefix, and pick the OPSS component to install the schema.

When finished selecting components, click **Next** to display the **Schema Passwords** dialog where you supply passwords, and then click **Next** to display the **Map**

Tablespaces dialog which shows the tablespace summary. Use one default tablespace and one temporary tablespace; the default tablespace names are PREFIX_IAS_OPSS and PREFIX_IAS_TEMP, respectively.

To create a non-default tablespace or datafile, click the button **Manage Tablespaces** to display the **Manage Tablespaces** dialog, where you can specify the information to create them. When finished, click **OK**. If the specified tablespaces are not yet in the database, RCU creates them and informs about this in the **Creating Tablespaces**; click **OK** to display the **Summary dialog**, which displays the summary of data you have entered, and then click **Create** to effect the creation of the additional tablespace(s) or datafile(s).

5. When the creation is completed, RCU displays the **Completion Summary**, which shows the database details.
6. Invoke the SQLPlus command illustrated below to verify that the database schema has been properly created:

```
SQL> desc jps_dn;
```

8.3.1.2 Dropping the OPSS Schema in an Oracle Database

Dropping the OPSS schema is required only if one no longer wishes to use that DB for storing OPSS security policies.

After the OPSS schema has been successfully created, use RCU to drop the OPSS schema as follows:

1. Start RCU to display the RCU **Welcome** page; in this page, click **Next** to display the **Drop Repository** page.
2. In that page, select the radio button **Drop**; then click **Next** to display the **Database Connections Details** page.
3. In that page, enter the appropriate connectivity information: Database Type, Host Name, Port, Service Name, Username, Password, and Role. Then click **Next** to display the **Select Components** dialog.
4. In that dialog, select the prefix and, in the **Component** hierarchy, check AS Common Schemas and Oracle Platform Security Services; then click **Next** to display the **Summary** page.
5. In that page, verify that the details gathered are correct, and click **Drop** to trigger the dropping; when the operation is successfully completed, RCU displays the **Completion Summary** page detailing the schema dropped.

8.3.1.3 Creating a Data Source Instance

To create a JDBC data source in a WebLogic domain using the Oracle WebLogic Administration Console, proceed as follows:

1. Login to the Console and navigate to **Services > DataSources** and select **New > Generic Data Source**.
2. Enter the **JNDI Name** and then click **Next**. Note that this name is used when configuring a DB-based store in the file `jps-config.xml`.
3. In the **Database Driver** pull-down, select **Oracle's Driver (Thin) for Instance connections; Versions:9.0.1 and later** (this is a non-XA JDBC driver) and then click **Next**.
4. Make sure that **Supports Global Transactions** is deselected and then click **Next**.

5. In the area **Connection Properties**, enter data for **Database Name**, **Host Name**, **Port**, **Database User Name**, and **Password**. Then click **Next**.
6. Inspect and test your settings and, when satisfied, click **Finish**.
7. Deploy the just created data source on the appropriate server.

For more details on the above procedure, see section *Creating a JDBC Data Source in Oracle Fusion Middleware Configuring and Managing JDBC for Oracle WebLogic Server*.

To set up a data source on WebSphere Application Server, see *Oracle Fusion Middleware Third-Party Application Server Guide*.

Note: 11.2 Oracle JDBC driver deprecated the following time zones: Etc/UCT, UCT, Etc/UTC, Etc/Universal, Etc/Zulu, and Universal. When setting a time zone for your Oracle JDBC driver, make sure that it is a non-deprecated time zone.

8.3.2 Maintaining a DB-Based Security Store

This section describes a few tasks that an administrator can follow to maintain a DB-based security store.

A DB-based security store maintains a change log that should be periodically purged. To purge it, an administrator can use the provided SQL script `opss_purge_changelog.sql`, which will purge change logs older than 24 hours, or connect to the database and run SQL `delete` (with the appropriate arguments) as illustrated in the following lines:

```
SQL>delete from jps_changelog where createdate < (select(max(createdate) - 1) from
jps_changelog);
SQL>Commit;
```

If the OPSS management API performs slowly while accessing the DB-based security store, run the `DBMS_STATS` package to gather statistics about the physical storage of a DB table, index, or cluster. This information is stored in the data dictionary and can be used to optimize the execution plan for SQL statements accessing analyzed objects.

When loading large amount of data into a DB-based security store, such as when creating thousands of new application roles, it is recommended that `DBMS_STATS` be run within short periods and concurrently with the loading activity. Otherwise, when the loading activity is small, `DBMS_STATS` needs to be run just once and according to your needs.

The following sample illustrates the use of `DBMS_STATS`:

```
EXEC DBMS_STATS.GATHER_SCHEMA_STATS('DEV_OPSS', DBMS_STATS.AUTO_SAMPLE_SIZE,
no_invalidate=>FALSE);
```

where `DEV_OPSS` denotes the name of the DB schema created during the RCU setup (see section [Creating the OPSS Schema in an Oracle Database](#)). For details about the `DBMS_STATS` package, see the *Oracle Database Administrator's Guide*.

To run `DBMS_STATS` periodically, use a shell script or an SQL script, as described next.

The following sample script runs the command `DBMS_STATS` every 10 minutes:

```
#!/bin/sh
i=1
while [ $i -le 1000 ]
do
```

```

echo $i
sqlplus dev_opss/welcome1@inst1 @opssstats.sql
sleep 600
i=`expr $i + 1`
done

```

where `opssstats.sql` contains the following text:

```

EXEC DBMS_STATS.gather_schema_stats('DEV_OPSS',DBMS_STATS.AUTO_SAMPLE_SIZE,
no_invalidate=>FALSE);
QUIT;

```

The following sample SQL script also runs the command `DBMS_STATS` every 10 minutes:

```

variable jobno number;
BEGIN
DBMS_JOB.submit
(job => :jobno,
what =>
'DBMS_STATS.gather_schema_stats(''DEV_OPSS'',DBMS_STATS.AUTO_SAMPLE_SIZE,no_invali
date=>FALSE);',
interval => 'SYSDATE+(10/24/60)');
COMMIT;
END;
/

```

To stop the periodic invocation of `DBMS_STATS` by the above SQL script, first find out its job number by issuing the following commands:

```

sqlplus '/as sysdba'
SELECT job FROM dba_jobs WHERE schema_user = 'DEV_OPSS' AND what =
'DBMS_STATS.gather_schema_stats(''DEV_OPSS'',DBMS_STATS.AUTO_SAMPLE_SIZE,
no_invalidate=>FALSE)';

```

Then issue a command like the following, in which it is assumed that the query above returned the job number 31:

```

EXEC DBMS_JOB.remove(31);

```

8.3.3 Setting Up an SSL Connection to the DB

Establishing a one- or two-way SSL connection to a DB-Based OPSS security store is optional and explained in section *Configuring SSL for the Database in Oracle Fusion Middleware Administrator's Guide*.

For additional information about SSL-related topics see the following documents:

- *SSL with Oracle JDBC Thin Driver* at the following link:
http://www.oracle.com/technology/tech/java/sqlj_jdbc/pdf/wp-oracle-jdbc_thin_ssl_2007.pdf.
- *Oracle Database JDBC Developer's Guide*.

8.4 Configuring the OPSS Security Store

For examples of store configurations for Java SE applications, see [Section 23.1, "Configuring Policy and Credential Stores in Java SE Applications."](#)

For examples of store configurations for Java EE applications, see [Example 1](#) and [Example 4](#).

For details about configuring other artifacts, see [Configuring the Identity Provider, Property Sets, and SSO](#).

8.5 Reassociating the OPSS Security Store

Reassociating the OPSS security store consists in relocating the policy, credential, and key stores from one repository to another one. The source can be file-, LDAP-, or DB-based; the target can be LDAP- or DB-based. The only type of LDAP target supported is Oracle Internet Directory; the only type of DB target supported is DB_ORACLE.

Reassociation changes the repository preserving the integrity of the data stored. For each security artifact, reassociation searches the target store and, if it finds a match for it, it updates the matching artifact; otherwise, creates a new artifact.

Reassociation is typically performed, for example, when setting a domain to use an LDAP- or DB-based OPSS store instead of the out-of-the-box file-based store. This operation can take place at any time after the OPSS store has been configured and instantiated, and it is carried out using either Fusion Middleware Control or `reassociateSecurityStore` as explained in the following sections:

- [Reassociating with Fusion Middleware Control](#)
- [Reassociating with the Script `reassociateSecurityStore`](#)

8.5.1 Reassociating with Fusion Middleware Control

Reassociation migrates the OPSS policy store (policies, credentials, and keys) from one repository to another and reconfigures the appropriate security store providers. This section explains how to perform reassociation with Fusion Middleware Control pages.

For information about other uses of the **Security Provider Configuration** page, see [Configuring the Identity Provider, Property Sets, and SSO](#).

Important Points

- Before reassociating to a target LDAP store, ensure that your setup satisfies the [Prerequisites to Using an LDAP-Based Security Store](#).
- Before reassociating to a target DB store, ensure that your setup satisfies the [Prerequisites to Using a DB-Based Security Store](#).
- If reassociation requires a one-way SSL to a target LDAP, follow the instructions in [Setting Up a One- Way SSL Connection to the LDAP](#) *before* reassociating.
- After reassociating to an LDAP store, to secure access to the root node of the Oracle Internet Directory store, follow the instructions in [Securing Access to Oracle Internet Directory Nodes](#).
- The `jps-config.xml` file produced by reassociation is good for only Java EE applications. In case of Java SE applications, edit the file `jps-config-jse.xml` to match the one described in [Section 23.1.3, "Configuring DB-Based OPSS Security Stores."](#)

To reassociate the OPSS security store with Fusion Middleware Control, proceed as follows:

1. Log in to Fusion Middleware Control and navigate to *Domain* > **Security** > **Security Provider Configuration** (if connected to Oracle WebLogic Server) or to *Cell* > **Security** > **Security Provider Configuration** (if connected to WebSphere)


Application Server), to display the **Security Provider Configuration** page, partially illustrated in the following graphic:

Security Provider Configuration

Use this page to configure WebLogic Domain policy and credential store providers, keystore and login modules used by Web Services Manager.

Security Stores

Current policy and credential store providers are shown below. To change the current policy and credential providers use the Change Association button.

Change Association 		
Name	Provider Type	Location
Policy Store		
Credential Store	File	./system-jazn-data.xml
Keystore		

The table in the area **Security Stores** shows the characteristics of the current provider configured in the domain.

- Click the button **Change Association** to display the **Set Security Provider** page, and choose the **Store Type** from the pull-down list. The text displayed on this page depends on the store type selected. The following graphic partially illustrates this page when Oracle Internet Directory is selected.

Security Provider Configuration > Set Security Provider

Information
All changes made in this page require a server restart to take effect.

Set Security Provider

OK Cancel

Specify server specific attributes to reassociate the policy, credential and farm key stores.

Store Type **LDAP Server Details**

Provide valid credential to connect to LDAP server. Farm uses this credential to connect to LDAP server for authentication and authorization.

* Host * Port Use SSL to connect * Connect DN

Test LDAP Authentication

* Password **Root Node Details**

Use this section to define provider specific configuration for this security store. To specify the root DN, enter the desired root name and domain name. Under Custom Properties, click Add, enter the name and desired value of the property in the resulting dialog, and click OK.

* Root DN Create New Domain * Domain Name **Policy Store Properties**

Specify policy store instance property configuration for getting maximum performance.

Enable Lazy Load Role Member Cache Size Permission Cache Size Update Cache Incrementally for Management Enable Store Refresh

3. If you have selected Database, enter the name of the data source in the **Datasource Name** box. This should be the name of the JDBC data source entered when the data source was created; see [Creating a Data Source Instance](#) for details. If needed, click **Select** to obtain a list of configured data source names.
4. If you have selected Oracle Internet Directory, in the **LDAP Server Details** area, specify details and connection information about the target LDAP server:
 1. Enter the host name and port number of your target Oracle Internet Directory LDAP server.
 2. Optionally, check the box **Use SSL to Connect** to establish an anonymous SSL transmission to the LDAP server.

When checking this box, keep in mind the following points:

The port of the target LDAP server must be configured to handle an anonymous SSL transmission; this port is distinct from the default (non-secure) LDAP server port.

If the reassociation is to use a one-way SSL to a target LDAP store, be sure to follow the instructions in [Setting Up a One-Way SSL Connection to the LDAP](#) before completing this step. Among other things, that setup identifies the port

to support a one-way SSL channel, and it is that port that should be specified in this step. Reassociation through a two-way SSL channel is not supported in this release.

Fusion Middleware Control modifies the file `weblogic.policy` by adding the necessary grant to support the anonymous SSL connection.

3. In the text box **Connect DN**, enter the full distinguished name, a string containing between 1 and 256 characters. For example, `cn=orcladmin,dc=us,dc=oracle,dc=com`.
4. In the box **Password**, enter the user password, also a string containing between 1 and 256 characters.
5. To verify that the connection to the LDAP server using the entered data works, click the button **Test LDAP Authentication**. If you run into any connection problem, see [Section L.9, "Failure to Establish an Anonymous SSL Connection."](#)
5. In the **Root Node Details** area, enter the root DN in the box **Root DN**, which identifies the top of the tree that contains the data in the LDAP repository. The **Domain Name** defaults to the name of the selected domain.

To solve most common errors arising from the specifications in these two fields, see [Section L.2, "Reassociation Failure."](#)

6. Optionally, in the **Policy Store Properties** and **Credential Store Properties** areas, enter service instance properties, such as Enable Lazy Load and Role Member Cache Size.

To add a new property: click **Add** to display the **Add New Property** dialog; in this dialog, enter strings for **Property Name** and **Value**; click **OK**. The added property-value pair is displayed in the table **Custom Properties**.

These properties are typically used to initialize the instance when it is created.

A property-value pair you enter modifies the domain configuration file `jps-config.xml` by adding a `<property>` element in the configuration of the LDAP service instance.

To illustrate how a service instance is modified, suppose you enter the property name `foo` and value `bar`; then the configuration for the LDAP service instance changes to contain a `<property>` element as illustrated in the following excerpt:

```
<serviceInstance name="myNewLDAPprovider" provider="someProvider"
  ...
  <property name="foo" value="bar"/>
  ...
</serviceInstance>
```

7. When finished entering your data, click **OK** to return to the **Security Provider Configuration** page. The system displays a dialog notifying the status of the reassociation. The table in the **Security Stores** area is modified to reflect the provider you have specified.
8. Restart the application server. Changes do not take effect until it has been restarted.

Reassociation modifies the domain configuration file `DOMAIN_HOME/config/fmwconfig/jps-config.xml`: it deletes any configuration for the old store provider, inserts a configuration for the new store provider, and moves the policy and credential information from the source to the destination store.

If the destination store is LDAP-based, the information is stored under the domain DN according to the following format:

```
cn=<domain_name>,cn=JpsContext,<JPS_ROOT_DN>
```

As long as the configuration of the installation relies upon the above domain DN, that node should not be deleted from the LDAP Server.

8.5.1.1 Securing Access to Oracle Internet Directory Nodes

The procedure explained in this section is optional and performed only to enhance the security to access an Oracle Internet Directory.

An access control list (ACL) is a list that specifies who can access information and what operations are allowed on the Oracle Internet Directory directory objects. The control list is specified at a node, and its restrictions apply to all entries in the subtree under that node.

ACL can be used to control the access to policy and credential data stored in an LDAP Oracle Internet Directory repository, and it is, typically, specified at the top, root node of the store.

To specify an ACL at a node in an Oracle Internet Directory repository, proceed as follows:

1. Create an LDIF file with a content that specifies the ACL:

```
dn: <storeRootDN>
changetype: modify
add: orclACI
access to entry by dn="<userDN>" (browse,add,delete) by * (none)
access to attr=(*) by dn="<userDN>" (search,read,write,compare) by * (none)
```

where storeRootDN stands for a node (typically the root node of the store), and userDN stands for the DN of the administrator data (the same userDN that was entered to perform reassociation).

2. Use the Oracle Internet Directory utility `ldapmodify` to apply these specifications to the Oracle Internet Directory.

Here is an example of an LDIF file specifying an ACL:

```
dn: cn=jpsRootNode
changetype: modify
add: orclACI
access to entry by dn="cn=myAdmin,cn=users,dc=us,dc=oracle,dc=com"
(browse,add,delete) by * ( none )
access to attr=(*) by dn="cn=myAdmin,cn=users,dc=us,dc=oracle,dc=com"
(search,read,write,compare) by * (none)
```

For more information about access control lists and the command `ldapmodify`, see chapter 18 in *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*.

8.5.2 Reassociating with the Script `reassociateSecurityStore`

The OPSS store can be reassociated with the OPSS script `reassociateSecurityStore`. For details, see [Section 9.3.29, "reassociateSecurityStore."](#)

8.6 Migrating the OPSS Security Store

A domain includes one and only one policy store. Applications can specify their own policies, but these are stored as policies in the policy store when the application is deployed to a server. All applications deployed in a domain use a common policy store, the policy store. The policy store is logically partitioned in stripes, one for each application name specified in the file `$DOMAIN_HOME/config/fmwconfig/system-jazn-data.xml` under the element `<applications>`.

Migrating the OPSS security store consists in relocating the policy, credential, and key stores from one repository to another one. The source can be file-, LDAP-, or DB-based; the target can be LDAP- or DB-based. The OPSS binaries and the target policy store must have compatible versions; for details, see [Section L.21, "Incompatible Versions of Binaries and Policy Store."](#)

During application development, an application specifies its own policies, and these can be migrated to the OPSS security store when the application is deployed with Fusion Middleware Control. Policies can also be migrated manually; in addition, each application component can specify the use of anonymous user and role, authenticated role, and JAAS mode.

The configuration of the policy store is performed by an administrator.

These topics are explained in the following sections:

- [Migrating with Fusion Middleware Control](#)
- [Migrating with the Script `migrateSecurityStore`](#)

Note: Use the system property `jps.deployment.handler.disabled` to disable the migration of application policies and credentials for applications deployed in a WebLogic Server.

When this system property is set to `TRUE`, the migration of policies and credentials at deployment is disabled for *all* applications regardless of the particular application settings in the application file `weblogic-application.xml`.

8.6.1 Migrating with Fusion Middleware Control

Application policies are specified in the application file `jazn-data.xml` and can be migrated to the policy store when the application is deployed to a server in the WebLogic environment with Fusion Middleware Control; they can also be removed from the policy store when the application is undeployed or be updated when the application is redeployed.

All three operations, the migration, the removal, and the updating of application policies, can take place regardless of the type of policy repository, but they do require particular configurations.

For details, see procedure in [Section 6.5.2, "Migrating Policies and Credentials at Deployment."](#)

8.6.2 Migrating with the Script `migrateSecurityStore`

Application-specific policies or system policies can be migrated manually from a source repository to a target repository using the OPSS script `migrateSecurityStore`.

This script is offline, that is, it does not require a connection to a running server to operate; therefore, the configuration file passed to the argument `configFile` need not be an actual domain configuration file, but it can be assembled *just* to specify the source and destination repositories of the migration.

Note: Since the script `migrateSecurityStore` recreates GUIDs and takes a long time to migrate large volume of data, you may want to consider migrating stores with an alternate procedure that uses Oracle Internet Directory bulk operations. For details, see [Section 6.5.2.3, "Migrating Large Volume Policy and Credential Stores."](#)

For further details about OPSS scripts and their syntax, see section Overview of WLST Command Categories, in *Oracle Fusion Middleware WebLogic Scripting Tool Command Reference*.

For platform-specific requirements to run an OPSS script, see [Important Note](#).

To migrate all policies (system *and* application-specific, for all applications) on WebLogic use the script (first) or interactive (second) syntaxes (arguments are written in separate lines for clarity):

```
migrateSecurityStore.py -type policyStore
                       -configFile jpsConfigFileLocation
                       -src srcJpsContext
                       -dst dstJpsContext
```

```
migrateSecurityStore(type="policyStore", configFile="jpsConfigFileLocation",
src="srcJpsContext", dst="dstJpsContext")
```

The meanings of the arguments (all required) are as follows:

- `configFile` specifies the location of a configuration file `jps-config.xml` relative to the directory where the script is run. Typically, this configuration file is created just to be used with the script and serves no other purpose. This file contains two jps-contexts that specify the source and destination stores.

In addition, if the migration involves one or two LDAP-based stores, then this file must contain a bootstrap jps-context that refers to the location of a `cwallet.sso` file where the credentials to access the LDAP based involved in the migration are kept.

- `src` specifies the name of a jps-context in the configuration file passed to the argument `configFile`.
- `dst` specifies the name of another jps-context in the configuration file passed to the argument `configFile`.

The contexts passed to `src` and `dst` must be defined in the passed configuration file and must have distinct names. From these two contexts, the script determines the locations of the source and the target repositories involved in the migration.

To migrate *just* system policies on WebLogic, use the script (first) or interactive (second) syntaxes (arguments are written in separate lines for clarity):

```
migrateSecurityStore.py -type globalPolicies
                        -configFile jpsConfigFileLocation
                        -src srcJpsContext
                        -dst dstJpsContext
```

```
migrateSecurityStore(type="globalPolicies", configFile="jpsConfigFileLocation",
src="srcJpsContext", dst="dstJpsContext")
```

The meanings of the arguments (all required) are identical to the previous case.

To migrate *just* application-specific policies on WebLogic, for one application, use the script (first) or interactive (second) syntaxes (arguments are written in separate lines for clarity):

```
migrateSecurityStore.py -type appPolicies
                        -configFile jpsConfigFileLocation
                        -src srcJpsContext
                        -dst dstJpsContext
                        -srcApp srcAppName
                        [-dstApp dstAppName]
                        [-overWrite trueOrfalse]
                        [migrateIdStoreMapping trueOrfalse]
                        [mode laxOrstrict]
```

```
migrateSecurityStore(type="appPolicies", configFile="jpsConfigFileLocation",
src="srcJpsContext", dst="dstJpsContext", srcApp="srcAppName",
[dstApp="dstAppName"], [overWrite="trueOrfalse"],
[migrateIdStoreMapping="trueOrfalse"], [mode="strict"])
```

The meanings of the arguments `configFile`, `src`, and `dst` are identical to the previous cases. The meaning of other five arguments is as follows:

- `srcApp` specifies the name of the source application, that is, the application whose policies are being migrated.
- `dstApp` specifies the name of the target application, that is, the application whose policies are being written. If unspecified, it defaults to the name of the source application.
- `migrateIdStoreMapping` specifies whether enterprise policies should be migrated. The default value is True. To filter out enterprise policies from the migration, that is, to migrate *just* application policies, set it to False.
- `overWrite` specifies whether a target policy matching a source policy should be overwritten by or merged with the source policy. Set to true to overwrite the target policy; set to false to merge matching policies. Optional. If unspecified, defaults to false.
- `mode` specifies whether the migration should stop and signal an error upon encountering a duplicate principal or a duplicate permission in an application policy. Either do not specify or set to lax to allow the migration to continue upon encountering duplicate items, to migrate just one of the duplicated items, and to log a warning to this effect.

If the input does not follow the syntax requirements above, the script execution fails and returns an error. In particular, the input must satisfy the following requisites: (a) the file `jps-config.xml` is found in the passed location; (b) the file `jps-config.xml` includes the passed jps-contexts; and (c) the source and the destination context names are distinct.

8.6.2.1 Examples of Use

For complete examples illustrating the use of this script, see [Section 6.5.2.1, "Migrating Policies Manually."](#)

8.7 Configuring the Identity Provider, Property Sets, and SSO

This section explains how to use Fusion Middleware Control to configure parameters used by the User and Role APIs, property and property sets, and to specify the Single Sign-On Provider, in the following sections:

- [Configuring the Identity Store Provider](#)
- [Configuring Properties and Property Sets](#)
- [Specifying a Single Sign-On Solution](#)

Note: The area of the page **Security Provider Configuration** labeled **Web Services Manager Authentication Providers** pertains to the configuration of Login Modules and the Keystore for Web Services Manager *only* and is not relevant to ADF or Java EE applications.

For details about the login modules available, their parameters, and the keystore for those components, see chapter 9 in *Oracle Fusion Middleware Security and Administrator's Guide for Oracle Web Services*.

8.7.1 Configuring the Identity Store Provider

To configure the parameters used by the User and Role API that interact with the identity store, proceed as follows:

1. Log in to Fusion Middleware Control and navigate to *Domain* > **Security** > **Security Provider Configuration**, or to *Cell* > **Security** > **Security Provider Configuration** to display the **Security Provider Configuration** page.
2. Expand, if necessary, the area **Identity Store Provider**, and click **Configure** to display the page **Identity Store Configuration**.
3. Manage custom properties, as appropriate, using the buttons **Add** and **Delete**.
4. When finished, click **OK** to save your settings and to return to the **Security Provider Configuration** page.

8.7.2 Configuring Properties and Property Sets

A property set is collection of properties typically used to define the properties of a service instance or generic properties of the domain.

For a list of OPSS configuration properties, see [Appendix F.2, "OPSS Configuration Properties."](#)

The elements `<property>` and `<propertySet>` in the file `$DOMAIN_HOME/config/fmwconfig/jps-config.xml` are used to define property and property sets. Property sets can be referenced by the element `<propertySetRef>`.

To define a property or a property set, proceed as follows:

1. Log in to Fusion Middleware Control and navigate to *Domain* > **Security** > **Security Provider Configuration**, or to *Cell* > **Security** > **Security Provider Configuration** to display the **Security Provider Configuration** page.

2. Expand, if necessary, the area **Advanced Properties**, and click **Configure** to display the **Advanced Properties** page, in which you can enter properties and property sets.
3. To enter a property, click **Add** in the **Properties** area to display the dialog **Add New Property**, and enter a property name and value. When finished, click **OK**. The entered property appears on the **Properties** table.
4. To enter a property set, click **Add Property Set** in the **Property Sets** area to display the dialog **Add Property Set**, and enter the property set name.
5. To enter a property in a property set, select a property set from the existing ones, then click **Add Property** to display the dialog **Add New Property**, and then enter a property name and value. The entered property is added to the list of properties in the selected property set.
6. Use the button **Delete** to remove a selected item from any table. When finished entering or editing properties and property sets, click **OK**.
7. Restart the Oracle WebLogic Server. Changes do not take effect until the server has been restarted.

The addition or deletion of property sets modifies the domain configuration file `$DOMAIN_HOME/config/fmwconfig/jps-config.xml`; the changes do not take effect until the server is restarted.

The elements `<property>` and `<propertySet>` added by the previous procedure are inserted directly under the element `<jpsConfig>`.

8.7.3 Specifying a Single Sign-On Solution

This section explains the OPSS Single Sign-On (SSO) Framework and how to configure an SSO solution using Fusion Middleware Control, in the following sections:

- [The OPSS SSO Framework](#)
- [Configuring an SSO Solution with Fusion Middleware Control](#)
- [OAM Configuration Example](#)

8.7.3.1 The OPSS SSO Framework

The OPSS SSO Framework provides a way to integrate applications in a domain with an SSO solution. Specifically, it provides applications a common set of APIs across SSO products, to handle login, logout and auto login.

One of these solutions, the OAM solution, is available out-of-the-box, and it includes the following features:

- Dynamic authentication - Upon accessing a part of a secured artifact that requires authentication, the application triggers authentication and redirects the user to be authenticated by the appropriate solution.
- Auto login - A user who has initially accessed an application anonymously registers an account with the application; upon a successful registration, the user is redirected to the authentication URL; the user can also be automatically logged in without being prompted.
- Global logout - When a user logs out of one application, the logout propagates across to any other application that is enabled by the solution.

For a configuration example of an OAM solution, see [OAM Configuration Example](#).

An SSO solution must provide a standard way for applications to login and logout users. After successful authentication, the SSO service is responsible to redirect the user to the appropriate URL.

It is assumed that the domain where the solution is applied has been configured to allow the Subject to contain the anonymous user and role before login and after logout, and authenticated roles after login. It is also assumed that the SSO provider has implemented a Credential Mapping Service. In the case of the out-of-the-box OAM solution, the provider implements `CredentialMapperService` that produces the appropriate OAM token.

The OPSS SSO framework does not support multi-level authentication.

Integration with the desired SSO solution requires a separate installation and appropriate configuration of the solution. For details about recommended solutions, see [Part IV, "Single Sign-On Configuration"](#).

8.7.3.2 Configuring an SSO Solution with Fusion Middleware Control

To specify the SSO solution used by a domain, proceed as follows:

1. Log in to Fusion Middleware Control and navigate to *Domain* > **Security** > **Security Provider Configuration** or *Cell* > **Security** > **Security Provider Configuration** to display the **Security Provider Configuration** page.
2. In that page, click the **Configure** in the Single Sign-On Provider area to display the **Single Sign-On Provider** page.
3. In that page, check the box **Configure Single Sign-On**, to allow entering data for the provider. All boxes are grayed out until this box is checked.
4. Select the **Provider Type** from the pull-down list, and enter the corresponding data for the selected provider (the data required changes with the type selected).
5. Select the **Authentication Level** from the pull-down list.
6. Optionally, manage the provider **Custom Properties** using the buttons **Add**, **Edit**, and **Delete**, at the bottom of the page.
7. When finished, click **OK** to save the entered data.

8.7.3.3 OAM Configuration Example

The SSO service configuration entered with the procedure described in [Configuring an SSO Solution with Fusion Middleware Control](#) is written to the file `jps-config.xml`. The data specified includes:

- A particular SSO service
- The auto-login and auto-logout URIs
- The authentication level
- The query parameters contained in the URLs returned by the selected SSO service
- The appropriate settings for token generation

The following fragment of a `jps-config.xml` file illustrates the configuration of an OAM SSO provider:

```
<propertySets>
  <propertySet name = "props.auth.url">
    <property name = "login.url.BASIC" value =
"http://host:port/oam_login.cgi?level=BASIC"/>
    <property name = "login.url.FORM" value =
```

```

"http://host:port/oam_login.cgi?level=FORM"/>
  <property name = "login.url.DIGEST" value =
"http://host:port/oam_login.cgi?level= DIGEST"/>
  <property name = "autologin.url" value = " http://host:port/obrar.cgi"/>
  <property name = "logout.url" value = "http://host:port/logout.cgi"/>
  <property name = "param.login.successurl" value = "successurl"/>
  <property name = "param.login.cancelurl" value = "cancelurl"/>
  <property name = "param.autologin.targeturl" value = "redirectto"/>
  <property name = "param.autologin.token" value = "cookie"/>
  <property name = "param.logout.targeturl" value = "targeturl"/>
</propertySet>

  <propertySet name="props.auth.uri">
    <property name="login.url.BASIC"
value="/${app.context}/adfauthentication?level=BASIC" />
    <property name="login.url.FORM"
value="/${app.context}/adfauthentication?level=FORM" />
    <property name="login.url.DIGEST"
value="/${app.context}/adfauthentication?level=DIGEST" />
    <property name="autologin.url" value="/obrar.cgi" />
    <property name="logout.url"
value="/${app.context}/adfauthentication?logout=true" />
  </propertySet>

  <propertySet name = "props.auth.level">
    <property name = "level.anonymous" value = "0"/>
    <property name = "level.BASIC" value = "1"/>
    <property name = "level.FORM" value = "2"/>
    <property name = "level.DIGEST" value = "3"/>
  </propertySet>
</propertySets>

<serviceProviders>
  <serviceProvider name = "sso.provider"
class = "oracle.security.jps.internal.sso.SsoServiceProvider"
type = "SSO">
  <description>SSO service provider</description>
</serviceProvider>
</serviceProviders>

<serviceInstances>
  <serviceInstance name = "sso" provider = "sso.provider">
    <propertySetRef ref = "props.auth.url"/>
    <propertySetRef ref = "props.auth.level"/>
    <property name = "default.auth.level" value = "2"/>
    <property name = "token.type" value = "OAMSSOToken"/>
    <property name = "token.provider.class" value =
"oracle.security.jps.wls.internal.sso.WlsTokenProvider"/>
    <property name="sso.provider.class"
value="oracle.security.wls.oam.providers.sso.OAMSSOServiceProviderImpl"/>
  </serviceInstance>
</serviceInstances>

<jpsContexts default = "default">
  <jpsContext name = "default">
    <serviceInstanceRef ref = "sso"/>
  </jpsContext>
</jpsContexts>

```

Table 8–1 describes the meaning of the properties involved in the configuration of an SSO provider.

Table 8–1 SSO Provider Properties

Property Name	Description
<code>logout.url</code>	The SSO provider logout URL.
<code>login.url.BASIC</code>	The SSO provider BASIC logout URL.
<code>login.url.FORM</code>	The SSO provider FORM logout URL.
<code>login.url.DIGEST</code>	The SSO provider DIGEST logout URL.
<code>autologin.url</code>	The self-registration URL for auto-login.
<code>logout.url</code>	The SSO provider logout URL.
<code>param.login.successurl</code>	The URL redirect after a successful login.
<code>param.login.cancelurl</code>	The URL redirect after a query cancelation.
<code>param.autologin.targeturl</code>	The URL redirect after auto-login.
<code>param.autologin.token</code>	The token for auto-login.
<code>param.logout.targeturl</code>	The URL redirect after logging out.

Regarding the configuration of an SSO provider, note the following important remarks:

- Any SSO provider must define the URI for at least the FORM login with the property `login.url.FORM`. The value need not be a URL.
- If the application supports a self-registration page URI or URL, it must be specified with the property `autologin.url`.
- If the SSO solution supports a global logout URI or URL, it must be specified with the property `logout.url`. The OAM solution supports global logout.
- The following properties, illustrated in the preceding example, are optional:
 - `param.login.successurl`
 - `param.login.cancelurl`
 - `param.autologin.targeturl`
 - `param.login.token`
 - `param.logout.targeturl`
- The use of the variable `app.context` in URI specifications, illustrated in values within the property set `props.auth.uri` in the preceding example, is allowed for only ADF applications when integrating with the OAM solution.
- The property set `props.auth.level` is required.
- The reference to `props.auth.url` is required.
- The property `sso.provider.class` within a service instance of the SSO provider is the fully qualified name of the class implementing a specific SSO solution.

In the case of the OAM solution, the provided class name is `oracle.security.wls.oam.providers.sso.OAMSSOServiceProviderImpl`.

- The property name `default.auth.level` within a service instance of the SSO provider must be set to 2, as illustrated in the preceding example.
- The property `token.type` within a service instance of the SSO provider is required.

This token type identifies the token set on the HTTP request by the SSO provider upon a successful authentication; the SSO provider uses this token, after the first time, to ensure that the user does not need to be reauthenticated and that his sign-on is still valid. In the case of the OAM solution, the token type must be `OAMSSOToken`, as illustrated in the preceding example.

- The property `token.provider.class` within a service instance of the SSO provider is the fully qualified name of the token class, and it is provider-specific.
- If an application implements a self-registration logic and wants to auto login a user after successful self-registration, it must call the OPSS `autoLogin` API; in turn, to allow this call, it must grant that application a code source permission named `CredentialMapping` with class `JpsPermission`.

The following fragment of the file `system-jazn-data.xml` illustrates the specification of this permission to the application `MyApp`:

```
<grant>
  <grantee>
    <codesource>
      <url>file:${oracle.deployed.app.dir}/<MyApp>${oracle.deployed.app.ext}
    </url>
    </codesource>
  </grantee>
  <permissions>
    <permission>
      <class>oracle.security.jps.JpsPermission</class>
      <name>CredentialMapping</name>
    </permission>
  </permissions>
</grant>
```

Note the use of system variables in the URL specification. For details, see Example in [<url>](#).

Managing the Policy Store

The following sections explain how an administrator can manage policies using either Fusion Middleware Control, OPSS scripts, or Oracle Entitlements Server:

- [Managing the Policy Store](#)
- [Managing Policies with Fusion Middleware Control](#)
- [Managing Application Policies with OPSS Scripts](#)
- [Managing Application Policies with Oracle Entitlements Server](#)

Typical operations include:

- Changing the grants of an existing application role.
- Revoking a permission from a principal.
- Creating and deleting application roles.
- Listing all application roles and members of an application role.

This chapter also includes the following sections:

- [Caching and Refreshing the Cache](#)
- [Granting Policies to Anonymous and Authenticated Roles with WLST Scripts](#)
- [Application Stripe for Versioned Applications in WLST Scripts](#)
- [Guidelines to Configure the Policy Store](#)

9.1 Managing the Policy Store

Only a user with the appropriate permissions, such as the domain administrator, can access data in the policy store.

The following sections explain how an administrator can manage policies using either Fusion Middleware Control, OPSS scripts, or Oracle Entitlements Server. Typical operations include:

- [Managing Policies with Fusion Middleware Control](#)
- [Managing Application Policies with OPSS Scripts](#)
- [Managing Application Policies with Oracle Entitlements Server](#)

To avoid unexpected authorization failures and to manage policies effectively, note the following important points:

Important Point 1: Before deleting a user, revoke all permissions, application roles, and enterprise groups that have been granted to the user. If you fail to remove all security artifacts referencing a user to be deleted, they are left dangling and, potentially, be inadvertently inherited if another user with the same name or uid is created at a later time.

Similar considerations apply to when a user name or uid is changed: all policies (grants, permissions, groups) referring to old data must be updated so that it works as expected with the changed data.

See [Section L.11, "User Gets Unexpected Permissions."](#)

Important Point 2: Policies use case sensitivity in names when they are applied. The best way to avoid possible authorization errors due to case in user or group names is to use the spelling of those names exactly as specified in the identity store.

It is therefore recommended that:

- When provisioning a policy, the administrator spell the names of users and groups used in the policy *exactly* as they are in the identity repository. This guarantees that queries into the policy store (involving a user or group name) work as expected.
- When entering a user name at run-time, the end-user enter a name that matches *exactly* the case of a name supplied in the identity repository. This guarantees that the user is authorized as expected.

See [Section L.4, "Failure to Grant or Revoke Permissions - Case Mismatch."](#)

Important Point 3: The name of a resource type, a resource, or an entitlement can contain printable characters only and it cannot start or end with a white space.

For other considerations regarding the use of characters in policies, in particular in role names, see [Section L.16, "Characters in Policies."](#)

Important Point 4: Authorization failures are not visible, by default, in the console. To have authorization failures sent to the console you must set the system variable `jps.auth.debug` as follows:

```
-Djps.auth.debug=true
```

In particular, to have `JpsAuth.checkPermission` failures sent to the console, you must set the variable as above.

9.2 Managing Policies with Fusion Middleware Control

Fusion Middleware Control allows managing system and application policies in a WebLogic domain, regardless of the type of policy store provider used in the domain, as explained in the following sections:

- [Managing Application Policies](#)
- [Managing Application Roles](#)
- [Managing System Policies](#)

9.2.1 Managing Application Policies

This section explains how to use Fusion Middleware Control to manage application policies.

Note: If multiple applications are to share a permission and to prevent permission check failures, the corresponding permission class must be specified in the system class path.

1. Log in to Fusion Middleware Control and navigate to *Domain* > **Security** > **Application Policies** (if the application is deployed on Oracle WebLogic Server), or to *Cell* > **Security** > **Application Policies** (if it is deployed on WebSphere Application Server), to display the **Application Policies** page, partially illustrated in the following graphic:


Application Policies
 Application policies are the authorization policies that an application relies upon for controlling access to its resources.
 To manage users and groups in the WebLogic Domain, use the [Oracle WebLogic Server Security Provider](#).





Policy Store Provider

Scope WebLogic Domain
 Provider XML
 Location ./system-jazn-data.xml

Search

Select an application stripe in policy store , select principal type and enter search keyword to query application security grants assigned to the principals. Click on searched principal to query policies assigned.

Application Stripe: ReAssApp
 Principal Type: User
 Name: Starts With 

 Create... |  Create Like... |  Edit... |  Delete...

Principal	Display Name
No security policies found.	

The area **Policy Store Provider** is read-only; when expanded, it displays the policy store provider currently in use in the domain or cell.

2. To display policies in an application matching a given principal or permission, expand the **Search** area, select the application stripe to search, enter a string to match (a principal name, principal group, or application role), and click the blue button. The results of the search are displayed in the table at the bottom of the page.

3. To create an application policy for the selected application stripe, click **Create** to display the **Create Application Grant** page where you add principals and permissions for the grant being created.
 1. To add permissions, click **Add** in the **Permissions** area to display the **Add Permission** dialog.

In the **Search** area of that dialog, first select **Permissions** or **Resource Types**; if **Permissions** was selected, then identify permissions matching a class or resource name, and determine the **Permission Class** and **Resource Name**; if **Resource Types** was selected, then identify the resource types matching a type name, and determine a type; then click **OK** to return to the **Create Application Grant** page. The permission you selected is displayed in the table in the **Permissions** area.
 2. To add principals, click the button **Add** in the **Grantee** area to display the dialog **Add Principal**.

In the **Search** area of that dialog, select a **Type**, enter strings to match principal names and display names, and click the blue button; the result of the query is displayed in the **Searched Principals** table; then select one or more rows from that table, and click **OK** to return to the **Create Application Grant** page. The principals you selected are displayed in the table in the **Grantee** area
 3. At any point you can remove an item from the table in the **Grantee** area by selecting it and clicking the **Delete** button; similarly, you can modify an item from that table by selecting it and clicking the **Edit** button.
 4. When finished, click **OK** to return to the **Application Policies** page. The principal and permissions of the policy created are displayed in the table at the bottom of the page.
4. To create an application policy based on an existing one:
 1. Select an existing policy from the table.
 2. Click **Create Like**, to display the **Create Application Grant Like** page. Notice that in this page the table of permissions is automatically filled in with the data extracted from the policy you selected.
 3. Modify those values, as appropriate, as explained in the substeps of step 3 above, and then click **OK**.


9.2.2 Managing Application Roles

This section explains how to use Fusion Middleware Control to manage application roles.

1. Log in to Fusion Middleware Control and navigate to *Domain* > **Security** > **Application Roles** (if the application is deployed on Oracle WebLogic Server), or to *Cell* > **Security** > **Application Roles** (if it is deployed on WebSphere Application Server), to display the **Application Roles** page partially illustrated in the following graphic:

Application Roles

Application roles are the roles used by security aware applications that are specific to the application. These roles are seeded by applications in single global policy store when the applications are registered. These are also application roles that are created in the context of end users accessing the application.

 To manage users and groups in the WebLogic Domain, use the [Oracle WebLogic Server Security Provider](#).

Policy Store Provider





Scope WebLogic Domain
 Provider XML
 Location ./system-jazn-data.xml

Search

Select an application and enter search keyword for role name to search for roles defined by this application. Use application stripe to search if application uses a stripe that is different from application name.

Application Stripe

Role Name

 Create... |  Create Like... |  Edit... |  Delete...

Role Name	Display Name
No application roles found.	

The area **Policy Store Provider** is read-only; when expanded, it displays the policy store provider currently in use in the domain or cell.

- To display roles in an application, expand the **Search** area, choose an application stripe to search, enter the data to match role names, and click the blue button. The results of the search are displayed in the table at the bottom of the page.
- To create an application role, click **Create** to display the **Create Application Role** page. Note that you need not enter data in all areas at once; for example, you could create a role by entering the role name and display name, save your data, and later on specify the members in it; similarly, you could enter data for role mapping at a later time.

In the area **General**, specify the following attributes of the role being created:

- The name of the role, in the text box **Role Name**.
- Optionally, the name to display for the role, in the text box **Display Name**.
- Optionally, a description of the role, the text box **Description**.

In the area **Members**, specify the users, groups, or other application roles, if any, into which the role being created is mapped.

To add application roles to the application role being created:

- Click **Add** , to display the **Add Principal** dialog.
- In this dialog, select a **Type** (application role, group, or user), enter a string to match principal names, and click the blue button; the result of the search is displayed in the **Searched Principals** table; select one or more principals from that table.
- When finished, click **OK** to return to the **Create Application Role** page. The selected application roles are displayed in the table **Members**.

4. At any point you can remove an item from the Members table by selecting it and clicking the **Delete** button; similarly, you can modify an item from the table by selecting it and clicking the **Edit** button.
5. Click **OK** to effect the role creation (or updating) and to return to the **Application Roles** page. The role just created is displayed in the table at the bottom of that page.
6. To create an application role based on an existing one:
 1. Select an existing role from the table.
 2. Click **Create Like**, to display the **Create Application Role Like** page. Notice that in this page some data is automatically filled in with the data extracted from the role you selected.
 3. Modify the list of roles and users, as appropriate, and then click **OK**.

To understand how permissions are inherited in a role hierarchy, see [Section 2.2.1, "Permission Inheritance and the Role Hierarchy."](#)

9.2.3 Managing System Policies


This section explains how to use Fusion Middleware Control to manage system policies for an Oracle WebLogic Server domain or for a WebSphere Application Server cell.

The procedure below enables creating two types of system policies: principal policies and codebase policies. A principal policy grants permissions to a list of users or groups. A codebase policy grants permissions to a piece of code, typically a URL or a JAR file; for example, an application using the Credential Store Framework requires an appropriate codebase policy. Wildcards and patterns are not supported in codebase URLs.

1. Log in to Fusion Middleware Control and navigate to *Domain* > **Security** > **System Policies** or to *Cell* > **Security** > **System Policies**, as appropriate, to display the **System Policies** page partially illustrated in the following graphic:

System Policies

System policies are the system-wide policies applied to all applications deployed to current management domain. You can grant special permissions and privileges to principal or codebase.

 To manage users and groups in the WebLogic Domain, use the [Oracle WebLogic Server Security Provider](#).





Policy Store Provider

Search

Enter search keyword for codebase to query system security grants. Click on any searched codebase entry to show permissions assigned to the codebase.

Type

Name 

 Create...	 Create Like...	 Edit...	 Delete...
Name			
Administrators			

The area **Policy Store Provider** is read-only; when expanded, it displays the policy store provider currently in use in the domain or cell.

2. To display system policies matching a given type, name, and permission, expand the **Search** area, enter the data to match, and click the blue button. The results of the search are displayed in the table at the bottom of the page; to display all current system policies, select the type **All** and leave the name and permission boxes blank.
3. At any point, you can edit the characteristics of a selected policy by clicking the **Edit** button, or remove it from the list by clicking the **Delete** button.

To create a system policy:

1. Click **Create** to display the **Create System Grant** page.
2. Select type of policy to create: **Principal** or **Codebase**. The UI differs slightly depending on the type chosen; the steps below assume the selection **Principal**.
3. To add permissions, click the button **Add** in the **Permissions** area to display the **Add Permission** dialog and choose a permission to add to the policy being created.
 1. Use the **Search** area to query permissions matching a type, principal name, or permission name. The result of the search is display in the table in the **Search** area.
 2. To choose the permission to add, select a permission from the table; note that, when a permission is selected, its details are rendered in the read-only **Customize** area.
 3. Click **OK** to return to the **Create System Grant** page. The selected permission is added to the table **Permissions**.
4. At any point, you can select a permission from the **Permissions** table and use the button **Edit** to change the characteristics of the permission, or the button **Delete** to remove it.
5. Click **OK** to return to the System Policies page.
6. The table in the area **Permissions for Codebase** is read-only and it shows the resource name, actions, and permission class associated with a codebase system policy.

9.3 Managing Application Policies with OPSS Scripts

An OPSS script is either a WLST script, in the context of the Oracle WebLogic Server, or a WASAdmin script, in the context of the WebSphere Application Server. The scripts listed in this section apply to both platforms: WebLogic Application Server and WebSphere Application Server.

An **online** script is a script that requires a connection to a running server. Unless otherwise stated, scripts listed in this section are online scripts and operate on a policy store, regardless of whether it is file-, LDAP-, or DB-based. There are a few scripts that are **offline**, that is, they do not require a server to be running to operate.

Read-only scripts can be performed only by users in the following WebLogic groups: Monitor, Operator, Configurator, or Admin. Read-write scripts can be performed only by users in the following WebLogic groups: Admin or Configurator. All WLST scripts are available out-of-the-box with the installation of the Oracle WebLogic Server.

WLST scripts can be run in interactive mode or in script mode. In interactive mode, you enter the script at a command-line prompt and view the response immediately after. In script mode, you write scripts in a text file (with a py file name extension) and run it without requiring input, much like the directives in a shell script.

WASAdmin scripts can be run in interactive mode only.

Important Note

Before invoking an OPSS script you must run (according to the platform you use) one of the scripts below to ensure that the required JARs are added to the class path.

On WebLogic:

```
>sh $ORACLE_HOME/common/bin/wlst.sh
```

To run an online script, you must connect to a WebLogic server as follows:

```
>java weblogic.WLST
>connect('servername', 'password', 'localhost:portnum')
```

For details about running OPSS scripts on WebSphere, see *Oracle Fusion Middleware Third-Party Application Server Guide*.

OPSS provides the following scripts on all supported platforms to administer application policies (all scripts are **online**, unless otherwise stated):

- [listAppStripes](#)
- [createAppRole](#)
- [deleteAppRole](#)
- [grantAppRole](#)
- [revokeAppRole](#)
- [listAppRoles](#)
- [listAppRolesMembers](#)
- [grantPermission](#)
- [revokePermission](#)
- [listPermissions](#)
- [deleteAppPolicies](#)
- [createResourceType](#)
- [getResourceType](#)
- [deleteResourceType](#)
- [createResource](#)
- [deleteResource](#)
- [listResources](#)
- [listResourceActions](#)
- [createEntitlement](#)
- [getEntitlement](#)
- [deleteEntitlement](#)
- [addResourceToEntitlement](#)

- [revokeResourceFromEntitlement](#)
- [listEntitlements](#)
- [grantEntitlement](#)
- [revokeEntitlement](#)
- [listEntitlement](#)
- [listResourceTypes](#)
- [reassociateSecurityStore](#)

All class names specified in the above scripts must be fully qualified path names. The argument `appStripe` refers to the application stripe (typically, identical to the application name) and identifies the subset of policies pertaining to a particular application.

For important information about the authenticated and the anonymous roles and WLST scripts, see [Section 9.5, "Granting Policies to Anonymous and Authenticated Roles with WLST Scripts."](#)

For the correct usage of the application stripe in versioned applications, see [Section 9.6, "Application Stripe for Versioned Applications in WLST Scripts."](#)

9.3.1 listAppStripes

The script `listAppStripes` lists application stripes. This script can be run in offline or online mode. When run in offline mode, a configuration file must be passed, and it lists the application stripes in the policy store referred to by the configuration in the default context of the passed configuration file. When run in online mode, a configuration file must *not* be passed, and it lists stripes in the policy store of the domain to which you connect. In any mode, if a regular expression is passed, it lists the application stripes with names that match the regular expression; otherwise, it lists all application stripes.

If this command is used in offline mode after reassociating to a DB-based, the configuration file produced by the reassociation *must* be manually edited as described in [Running an Offline Script after Reassociating to a DB-Based Store](#).

Script Mode Syntax

```
listAppStripes.py [-configFile configFileName]
                 [-regularExpression aRegExp]
```

Interactive Mode Syntax

```
listAppStripes([configFile="configFileName"] [, regularExpression="aRegExp"])
```

The meanings of the arguments are as follows:

- `configFile` specifies the path to the OPSS configuration file. Optional. If specified, the script runs offline; the default context in the specified configuration file *must not* have a service instance reference to an identity store. If unspecified, the script runs online and it lists application stripes in the policy store.
- `regularExpression` specifies the regular expression that stripe names returned should match. Optional. If unspecified, it matches all names. To match substrings, use the character `*`.

Examples of Use

The following (online) invocation returns the list of application stripes in the policy store:

```
listAppStripes.py
```

The following (offline) invocation returns the list of application stripes in the policy store referenced in the default context of the specified configuration file:

```
listAppStripes.py -configFile /home/myFiles/jps-config.xml
```

The following (online) invocation returns the list of application stripes that contain the prefix App:

```
listAppStripes.py -regularExpression App*
```

9.3.2 createAppRole

The script `createAppRole` creates an application role in the policy store with given application stripe and role name.

Script Mode Syntax

```
createAppRole.py -appStripe appName
                 -appRoleName roleName
```

Interactive Mode Syntax

```
createAppRole(appStripe="appName", appRoleName="roleName")
```

The meanings of the arguments (all required) are as follows:

- `appStripe` specifies an application stripe.
- `appRoleName` specifies a role name.

Example of Use

The following invocation creates an application role with application stripe `myApp` and role name `myRole`:

```
createAppRole.py -appStripe myApp -appRoleName myRole
```

9.3.3 deleteAppRole

The script `deleteAppRole` removes an application role from the passed stripe. Specifically, this script applies a cascading deletion by removing:

- All grants where the role is present
- The role from any other role of which it is a member
- All roles that are member of the role

Script Mode Syntax

```
deleteAppRole.py -appStripe appName -appRoleName roleName
```

Interactive Mode Syntax

```
deleteAppRole(appStripe="appName", appRoleName="roleName")
```


The meanings of the arguments (all required) are as follows:

- `appStripe` specifies an application stripe.
- `appRoleName` specifies a role name.

Example of Use

The following invocation removes the role with application stripe `myApp` and name `myRole`:

```
deleteAppRole.py -appStripe myApp -appRoleName myRole
```

9.3.4 grantAppRole

The script `grantAppRole` adds a principal (class and name) to a role with a given application stripe and name, and it can be used to build or modify an application role hierarchy.

Script Mode Syntax

```
grantAppRole.py -appStripe appName
                 -appRoleName roleName
                 -principalClass className
                 -principalName prName
```

Interactive Mode Syntax

```
grantAppRole(appStripe="appName", appRoleName="roleName",
principalClass="className", principalName="prName")
```

The meanings of the arguments (all required) are as follows:

- `appStripe` specifies an application stripe.
- `appRoleName` specifies a role name.
- `principalClass` specifies the fully qualified name of a class; this class must be included in the class path so that it is available at runtime. Typically, if the principal is a user, the class is `weblogic.security.principal.WLSUserImpl`, and if the principal is a group, the class is `weblogic.security.principal.WLSGroupImpl`.
- `principalName` specifies the principal name.

Example of Use

The following invocation adds the principal `myPrincipal`, defined by the default principal implementation class `WLSGroupImpl`, to the role `myRole` in the application stripe `myApp`:

```
grantAppRole.py -appStripe myApp
                 -appRoleName myRole
                 -principalClass weblogic.security.principal.WLSGroupImpl
                 -principalName myPrincipal
```

9.3.5 revokeAppRole

The script `revokeAppRole` removes a principal (class and name) from a role with a given application stripe and name, and it can be used to modify an application role hierarchy.

Script Mode Syntax

```
revokeAppRole.py -appStripe appName
                 -appRoleName roleName
                 -principalClass className
                 -principalName prName
```

Interactive Mode Syntax

```
revokeAppRole(appStripe="appName", appRoleName="roleName",
principalClass="className", principalName="prName")
```

The meanings of the arguments (all required) are as follows:

- `appStripe` specifies an application stripe.
- `appRoleName` specifies a role name.
- `principalClass` specifies the fully qualified name of the principal class.
- `principalName` specifies the principal name.

Example of Use

The following invocation removes the principal `myPrincipal`, defined by the default principal implementation class `WLSGroupImpl`, from the role `myRole` in the application stripe `myApp`:

```
revokeAppRole.py -appStripe myApp
                 -appRoleName myRole
                 -principalClass weblogic.security.principal.WLSGroupImpl
                 -principalName myPrincipal
```

9.3.6 listAppRoles

The script `listAppRoles` lists all roles with a given application stripe.

Script Mode Syntax

```
listAppRoles.py -appStripe appName
```

Interactive Mode Syntax

```
listAppRoles(appStripe="appName")
```

The meaning of the argument (required) is as follows:

- `appStripe` specifies an application stripe.

Example of Use

The following invocation returns all the roles with application stripe `myApp`:

```
listAppRoles.py -appStripe myApp
```

9.3.7 listAppRolesMembers

The script `listAppRoleMembers` lists all members in a role with a given application stripe and role name.

Script Mode Syntax

```
listAppRoleMembers.py -appStripe appName
                     -appRoleName roleName
```

Interactive Mode Syntax

```
listAppRoleMembers(appStripe="appName", appRoleName="roleName")
```

The meanings of the arguments (all required) are as follows:

- `appStripe` specifies an application stripe.
- `appRoleName` specifies a role name.

Example of Use

The following invocation returns all the members in a role with application stripe `myApp` and name `myRole`:

```
listAppRoleMembers.py -appStripe myApp
                      -appRoleName myRole
```

9.3.8 grantPermission

The script `grantPermission` creates a permission granted to a codebase or URL or principal, in either an application policy or the global policy section.

Script Mode Syntax

```
grantPermission [-appStripe appName]
                [-codeBaseURL url]
                [-principalClass prClassName]
                [-principalName prName]
                -permClass permissionClassName
                [-permTarget permName]
                [-permActions comma_separated_list_of_actions]
```

Interactive Mode Syntax

```
grantPermission([appStripe="appName",] [codeBaseURL="url",]
                [principalClass="prClassName",] [principalName="prName",]
                permClass="permissionClassName", [permTarget="permName",]
                [permActions="comma_separated_list_of_actions"])
```

The meanings of the arguments (optional arguments are enclosed in between square brackets) are as follows:

- `appStripe` specifies an application stripe. If not specified, then the script works on system policies.
- `codeBaseURL` specifies the URL of the code granted the permission.
- `principalClass` specifies the fully qualified name of a class (grantee).
- `principalName` specifies the name of the grantee principal.
- `permClass` specifies the fully qualified name of the permission class.
- `permTarget` specifies, when available, the name of the permission target. Some permissions may not include this attribute.
- `permActions` specifies the list of actions granted. Some permissions may not include this attribute and the actions available depend on the permission class.

Examples of Use

The following invocation creates an application permission (for the application with application stripe `myApp`) with the specified data:

```
grantPermission.py -appStripe myApp
```

```
-principalClass my.custom.Principal
-principalName manager
-permClass java.security.AllPermission
```

The following invocation creates a system permission with the specified data:

```
grantPermission.py -principalClass my.custom.Principal
-principalName manager
-permClass java.io.FilePermission
-permTarget /tmp/fileName.ext
-permActions read,write
```

9.3.9 revokePermission

The script `revokePermission` removes a permission from a principal or codebase defined in an application or the global policy section.

Script Mode Syntax

```
revokePermission [-appStripe appName]
                 [-codeBaseURL url]
                 [-principalClass prClassName]
                 [-principalName prName]
                 -permClass permissionClassName
                 [-permTarget permName]
                 [-permActions comma_separated_list_of_actions]
```

Interactive Mode Syntax

```
revokePermission([appStripe="appName",][codeBaseURL="url",]
[principalClass="prClassName",] [principalName="prName",]
permClass="permissionClassName", [permTarget="permName",] [permActions="comma_
separated_list_of_actions"])
```

The meanings of the arguments (optional arguments are enclosed in between square brackets) are as follows:

- `appStripe` specifies an application stripe. If not specified, then the script works on system policies.
- `codeBaseURL` specifies the URL of the code granted the permission.
- `principalClass` specifies the fully qualified name of a class (grantee).
- `principalName` specifies the name of the grantee principal.
- `permClass` specifies the fully qualified name of the permission class.
- `permTarget` specifies, when available, the name of the permission target. (Note that some permissions may not include this attribute.)
- `permActions` specifies the list of actions removed. Note that some permissions may not include this attribute and the actions available depend on the permission class.

Examples of Use

The following invocation removes the application permission (for the application with application stripe `myApp`) with the specified data:

```
revokePermission.py -appStripe myApp
-principalClass my.custom.Principal
-principalName manager
```

```
-permClass java.security.AllPermission
```

The following invocation removes the system permission with the specified data:

```
revokePermission.py -principalClass my.custom.Principal
                    -principalName manager
                    -permClass java.io.FilePermission
                    -permTarget /tmp/fileName.ext
                    -permActions read,write
```

9.3.10 listPermissions

The script `listPermissions` lists all permissions granted to a given principal.

Script Mode Syntax

```
listPermissions [-appStripe appName]
                -principalClass className
                -principalName prName
```

Interactive Mode Syntax

```
listPermissions([appStripe="appName",] principalClass="className",
principalName="prName")
```

The meanings of the arguments (optional arguments are enclosed in between square brackets) are as follows:

- `appStripe` specifies an application stripe. If not specified, then the script works on system policies.
- `principalClass` specifies the fully qualified name of a class (grantee).
- `principalName` specifies the name of the grantee principal.

Examples of Use

The following invocation lists all permissions granted to a principal by the policies of application `myApp`:

```
listPermissions.py -appStripe myApp
                  -principalClass my.custom.Principal
                  -principalName manager
```

The following invocation lists all permissions granted to a principal by system policies:

```
listPermissions.py -principalClass my.custom.Principal
                  -principalName manager
```

9.3.11 deleteAppPolicies

The script `deleteAppPolicies` removes all policies with a given application stripe.

Script Mode Syntax

```
deleteAppPolicies -appStripe appName
```

Interactive Mode Syntax

```
deleteAppPolicies(appStripe="appName")
```

The meaning of the argument (required) is as follows:

- `appStripe` specifies an application stripe. If not specified, then the script works on just system policies.

Example of Use

```
deleteAppPolicies -appStripe myApp
```

9.3.12 createResourceType

The script `createResourceType` inserts a new `<resource-type>` entry in the policy store within a given application stripe and with specified name, display name, description, and actions. Optional arguments are enclosed in between square brackets; all other arguments are required.

Script Mode Syntax

```
createResourceType -appStripe appStripeName
                  -resourceTypeName resTypeName
                  -displayName displName
                  -description descripString
                  [-provider resTypeProvider]
                  [-matcher resTypeClass]
                  -actions resTypeActions
                  [-delimiter delimChar]
```

Interactive Mode Syntax

```
createResourceType(appStripe="appStripeName", resourceTypeName="resTypeName",
displayName="displName", description="descripString"
[, provider="resTypeProvider", matcher="resTypeClass"], actions="resTypeActions"[,
delimiter="delimChar"])
```

The meaning of the arguments is as follows:

- `appStripe` specifies the application stripe where to insert the resource type.
- `resourceTypeName` specifies the name of the resource type to insert.
- `displayName` specifies the name for the resource type used in UI gadgets.
- `description` specifies a brief description of the resource type.
- `provider` specifies the provider for the resource type.
- `matcher` specifies the class of the resource type. If unspecified, it defaults to `oracle.security.jps.ResourcePermission`.
- `actions` specifies the actions allowed on instances of the resource type.
- `delimiter` specifies the character used to delimit the list of actions. If unspecified, it defaults to comma `,`.

Example of Use

The following invocation creates a resource type in the stripe `myApplication` with actions `BWPrint` and `ColorPrint` delimited by a semicolon:

```
createResourceType -appStripe myApplication
                  -resourceTypeName Printer
                  -displayName PRINTER
                  -description A resource type representing a Printer
                  -provider Printer
                  -matcher com.printer.Printer
                  -allowedActions BWPrint;ColorPrint
```

```
-delimiter ;
```

9.3.13 getResourceType

The script `getResourceType` returns the relevant parameters of a <resource-type> entry in the policy store within a given application stripe and with specified name.

Script Mode Syntax

```
getResourceType -appStripe appStripeName
                -resourceTypeName resTypeName
```

Interactive Mode Syntax

```
getResourceType (appStripe="stripeName", resourceTypeName="resTypeName")
```

The meaning of the arguments is as follows:

- `appStripe` specifies the application stripe from where to fetch the resource type.
- `resourceTypeName` specifies the name of the resource type to fetch.

Example of Use

The following invocation fetches the resource type `myResType` from the stripe `myApplication`:

```
getResourceType -appStripe myApplication
                -resourceTypeName myResType
```

9.3.14 deleteResourceType

The script `deleteResourceType` removes a resource type with a given name from the passed application stripe. This script applies a cascading deletion by removing all resource instances of the resource type from entitlements and all grants that use resource instances of the resource type.

Important: A resource type *cannot* be modified after it has been created. If you need to modify a resource type in any way (such as adding, renaming, or deleting an action in it), you must delete the resource type and create a new one with the appropriate values. Specifically, you must:

- Create a new resource type.
 - Create the required new resource instances.
 - Create the required grants.
-
-

Script Mode Syntax

```
deleteResourceType -appStripe appStripeName
                  -resourceTypeName resTypeName
```

Interactive Mode Syntax

```
deleteResourceType (appStripe="stripeName", resourceTypeName="resTypeName")
```

The meaning of the arguments is as follows:

- `appStripe` specifies the application stripe from where to remove the resource type.

- `resourceTypeName` specifies the name of the resource type to remove.

Example of Use

The following invocation removes the resource type `myResType` from the stripe `myApplication`:

```
deleteResourceType -appStripe myApplication
                  -resourceTypeName myResType
```

9.3.15 createResource

The script `createResource` creates a new resource of a specified type in a specified application stripe. The passed resource type must exist in the passed application stripe.

Script Mode Syntax

```
createResource -appStripe appStripeName
              -name resName
              -type resTypeName
              [-displayName dispName]
              [-description descript]
```

Interactive Mode Syntax

```
createResource(appStripe="appStripeName", name="resName", type="resTypeName"
[, -displayName="dispName"] [, -description="descript"])
```

The meaning of the arguments is as follows:

- `appStripe` specifies the application stripe where the resource is created.
- `name` specifies the name of the resource created.
- `type` specifies the type of resource created. The passed resource type *must* be present in the application stripe at the time this script is invoked.
- `displayName` specifies the display name of the resource created. Optional.
- `description` specifies the description of the resource created. Optional.

Example of Use

The following invocation creates the resource `myResource` in the stripe `myApplication`:

```
createResource -appStripe myApplication
              -name myResource
              -type myResType
              -displayName myNewResource
```

9.3.16 deleteResource

The script `deleteResource` deletes a resource and all its references from entitlements in an application stripe. The script performs a cascading deletion: if the entitlement refers to one resource only, it removes the entitlement; otherwise, it removes from the entitlement the resource actions for the passed type.

Script Mode Syntax

```
deleteResource -appStripe appStripeName
              -name resName
              -type resTypeName
```


Interactive Mode Syntax

```
deleteResource(appStripe="appStripeName", name="resName", type="resTypeName")
```

The meaning of the arguments is as follows:

- `appStripe` specifies the application stripe where the resource is deleted.
- `name` specifies the name of the resource deleted.
- `type` specifies the type of resource deleted. The passed resource type *must* be present in the application stripe at the time this script is invoked.

Example of Use

The following invocation deletes the resource `myResource` in the stripe `myApplication`:

```
deleteResource -appStripe myApplication
               -name myResource
               -type myResType
```

9.3.17 listResources

The script `listResources` lists resources in a specified application stripe. If a resource type is specified, it lists all the resources of the specified resource type; otherwise, it lists all the resources of all types.

Script Mode Syntax

```
listResources -appStripe appStripeName
              [-type resTypeName]
```

Interactive Mode Syntax

```
listResources(appStripe="appStripeName" [,type="resTypeName"])
```

The meaning of the arguments is as follows:

- `appStripe` specifies the application stripe where the resources are listed.
- `type` specifies the type of resources listed. The passed resource type *must* be present in the application stripe at the time this script is invoked.

Examples of Use

The following invocation lists all resources of type `myResType` in the stripe `myApplication`:

```
listResources -appStripe myApplication
               -type myResType
```

The following invocation lists all resources in the stripe `myApplication`:

```
listResources -appStripe myApplication
```

9.3.18 listResourceActions

The script `listResourceActions` lists the resources and actions in an entitlement within an application stripe.

Script Mode Syntax

```
listResourceActions -appStripe appStripeName
                   -permSetName entitlementName
```

Interactive Mode Syntax

```
listResourceActions (appStripe="appStripeName", permSetName="entitlementName")
```

The meaning of the arguments is as follows:

- `appStripe` specifies the application stripe where the entitlement resides.
- `permSetName` specifies the name of the entitlement whose resources and actions to list.

Example of Use

The following invocation lists the resources and actions of the entitlement `myEntitlement` in the stripe `myApplication`:

```
listResourceActions -appStripe myApplication
                   -permSetName myEntitlement
```

9.3.19 createEntitlement

The script `createEntitlement` creates a new entitlement with just one resource and a list of actions in a specified application stripe. Use `addResourceToEntitlement` to add additional resources to an existing entitlement; use `revokeResourceFromEntitlement` to delete resources from an existing entitlement.

Script Mode Syntax

```
createEntitlement -appStripe appStripeName
                 -name entitlementName
                 -resourceName resName
                 -actions actionList
                 [-displayName dispName]
                 [-description descript]
```

Interactive Mode Syntax

```
createEntitlement (appStripe="appStripeName", name="entitlementName",
resourceName="resName", actions="actionList" [, -displayName="dispName"]
[, -description="descript"])
```

The meaning of the arguments is as follows:

- `appStripe` specifies the application stripe where the entitlement is created.
- `name` specifies the name of the entitlement created.
- `resourceName` specifies the name of the one resource member of the entitlement created.
- `actions` specifies a comma-separated the list of actions for the resource `resourceName`.
- `diplayName` specifies the display name of the resource created. Optional.
- `description` specifies the description of the entitlement created. Optional.

Example of Use

The following invocation creates the entitlement `myEntitlement` with just the resource `myResource` in the stripe `myApplication`:

```
createEntitlement -appStripe myApplication
                 -name myEntitlement
```

```
-resourceName myResource
-actions read,write
```

9.3.20 getEntitlement

The script `getEntitlement` returns the name, display name, and all the resources (with their actions) of an entitlement in an application stripe.

Script Mode Syntax

```
getEntitlement -appStripe appStripeName
              -name entName
```

Interactive Mode Syntax

```
getEntitlement(appStripe="appStripeName", name="entName")
```

The meaning of the arguments is as follows:

- `appStripe` specifies the application stripe where the entitlement is located.
- `name` specifies the name of the entitlement to access.

Example of Use

The following invocation returns the information of the entitlement `myEntitlement` in the stripe `myApplication`:

```
getEntitlement -appStripe myApplication
              -name myEntitlement
```

9.3.21 deleteEntitlement

The script `deleteEntitlement` deletes an entitlement in a specified application stripe. The script performs a cascading deletion by removing all references to the specified entitlement in the application stripe.

Script Mode Syntax

```
deleteEntitlement -appStripe appStripeName
                 -name entName
```

Interactive Mode Syntax

```
deleteEntitlement(appStripe="appStripeName", name="entName")
```

The meaning of the arguments is as follows:

- `appStripe` specifies the application stripe where the entitlement is deleted.
- `name` specifies the name of the entitlement to delete.

Example of Use

The following invocation deletes the entitlement `myEntitlement` in the stripe `myApplication`:

```
deleteEntitlement -appStripe myApplication
                 -name myEntitlement
```

9.3.22 addResourceToEntitlement

The script `addResourceToEntitlement` adds a resource with specified actions to an entitlement in a specified application stripe. The passed resource type must exist in the passed application stripe.

Script Mode Syntax

```
addResourceToEntitlement -appStripe appStripeName
                        -name entName
                        -resourceName resName
                        -resourceType resType
                        -actions actionList
```

Interactive Mode Syntax

```
addResourceToEntitlement (appStripe="appStripeName", name="entName",
resourceName="resName",actions="actionList")
```

The meaning of the arguments is as follows:

- `appStripe` specifies the application stripe where the entitlement is located.
- `name` specifies the name of the entitlement to modify.
- `resourceName` specifies the resource to add.
- `resourceType` specifies the type of the resource to add. The passed resource type *must* be present in the application stripe at the time this script is invoked.
- `actions` specifies the comma-separated list of actions for the added resource.

Example of Use

The following invocation adds the resource `myResource` to the entitlement `myEntitlement` in the application stripe `myApplication`:

```
addResourceToEntitlement -appStripe myApplication
                        -name myEntitlement
                        -resourceName myResource
                        -resourceType myResType
                        -actions view,edit
```

9.3.23 revokeResourceFromEntitlement

The script `revokeResourceFromEntitlement` removes a resource from an entitlement in a specified application stripe.

Script Mode Syntax

```
revokeResourceFromEntitlement -appStripe appStripeName
                              -name entName
                              -resourceName resName
                              -resourceType resTypeName
                              -actions actionList
```

Interactive Mode Syntax

```
revokeResourceFromEntitlement (appStripe="appStripeName", name="entName",
resourceName="resName" , -resourceType="resTypeName", actions="actionList")
```

The meaning of the arguments is as follows:

- `appStripe` specifies the application stripe where the entitlement is located.

- `name` specifies the name of the entitlement to modify.
- `resourceName` specifies the type of resource to remove.
- `resourceType` specifies the type of the resource to remove.
- `actions` specifies the comma-separated list of actions to remove.

Example of Use

The following invocation removes the resource `myResource` from the entitlement `myEntitlement` in the stripe `myApplication`:

```
revokeResourceFromEntitlement -appStripe myApplication
                             -name myEntitlement
                             -resourceName myResource
                             -resourceType myResType
                             -actions view,edit
```

9.3.24 listEntitlements

The script `listEntitlements` lists all the entitlements in an application stripe. If a resource name and a resource type are specified, it lists the entitlements that have a resource of the specified type matching the specified resource name; otherwise, it lists all the entitlements in the application stripe.

Script Mode Syntax

```
listEntitlements -appStripe appStripeName
                [-resourceTypeName resTypeName]
                [-resourceName resName]
```

Interactive Mode Syntax

```
listEntitlements(appStripe="appStripeName" [,resourceTypeName="resTypeName",
resourceName="resName"])
```

The meaning of the arguments is as follows:

- `appStripe` specifies the application stripe from where to list entitlements.
- `resourceTypeName` specifies the name of the type of the resources to list. Optional.
- `resourceName` specifies the name of resource to match. Optional.

Examples of Use

The following invocation lists all the entitlements in the stripe `myApplication`:

```
listEntitlements -appStripe myApplication
```

The following invocation lists all the entitlements in the stripe `myApplication` that contain a resource type `myResType` and a resource whose name match the resource name `myResName`:

```
listEntitlements -appStripe myApplication
                -resourceTypeName myResType
                -resourceName myResName
```

9.3.25 grantEntitlement

The script `grantEntitlement` creates a new entitlement with a specified principal in a specified application stripe.

Script Mode Syntax

```
grantEntitlement -appStripe appStripeName
                 -principalClass principalClass
                 -principalName principalName
                 -permSetName entName
```

Interactive Mode Syntax

```
grantEntitlement (appStripe="appStripeName", principalClass="principalClass",
principalName="principalName" , -permSetName="entName")
```

The meaning of the arguments is as follows:

- *appStripe* specifies the application stripe where the entitlement is created.
- *principalClass* specifies the class associated with the principal.
- *principalName* specifies the name of the principal to which the entitlement is granted.
- *permSetName* specifies the name of the entitlement created.

Example of Use

The following invocation creates the entitlement *myEntitlement* in the stripe *myApplication*:

```
grantEntitlement -appStripe myApplication
                 -principalClass
oracle.security.jps.service.policystore.ApplicationRole
                 -principalName myPrincipalName
                 -permSetName myEntitlement
```

9.3.26 revokeEntitlement

The script *revokeEntitlement* deletes an entitlement and revokes the entitlement from the principal in a specified application stripe.

Script Mode Syntax

```
revokeEntitlement -appStripe appStripeName
                  -principalClass principalClass
                  -principalName principalName
                  -permSetName entName
```

Interactive Mode Syntax

```
revokeEntitlement (appStripe="appStripeName", principalClass="principalClass",
principalName="principalName" , -permSetName="entName")
```

The meaning of the arguments is as follows:

- *appStripe* specifies the application stripe where the entitlement is deleted.
- *principalClass* specifies the class associated with the principal.
- *principalName* specifies the name of the principal to which the entitlement is revoked.
- *permSetName* specifies the name of the entitlement deleted.

Example of Use

The following invocation deletes the entitlement `myEntitlement` in the stripe `myApplication`:

```
revokeEntitlement -appStripe myApplication
                 -principalClass
oracle.security.jps.service.policystore.ApplicationRole
                 -principalName myPrincipalName
                 -permSetName myEntitlement
```

9.3.27 listEntitlement

The script `listEntitlement` lists an entitlement in a specified application stripe. If a principal name and a class are specified, it lists the entitlements that match the specified principal; otherwise, it lists all the entitlements.

Script Mode Syntax

```
listEntitlement -appStripe appStripeName
                [-principalName principalName
                -principalClass principalClass]
```

Interactive Mode Syntax

```
listEntitlement (appStripe="appStripeName" [, principalName="principalName",
principalClass="principalClass"])
```

The meaning of the arguments is as follows:

- `appStripe` specifies the application stripe where the entitlement is located.
- `principalName` specifies the name of the principal to match. Optional.
- `principalClass` specifies the class of the principal to match. Optional.

Example of Use

The following invocation lists all entitlements in the stripe `myApplication`:

```
listEntitlement -appStripe myApplication
```

9.3.28 listResourceTypes

The script `listResourceTypes` lists all the resource types in a specified application stripe.

Script Mode Syntax

```
listResourceTypes -appStripe appStripeName
```

Interactive Mode Syntax

```
listResourceTypes (appStripe="appStripeName")
```

The meaning of the arguments is as follows:

- `appStripe` specifies the application stripe where the resource types are located.

Example of Use

The following invocation lists all resource types in the stripe `myApplication`:

```
listResourceTypes -appStripe myApplication
```

9.3.29 reassociateSecurityStore

The script `reassociateSecurityStore` migrates the OPSS security store from a source to a target LDAP- or DB-based store, and it resets the default policy and credential services to the target repository. It also allows specifying that the OPSS security store be shared with that in a different domain (see optional argument `join` below). The OPSS binaries and the target policy store must have compatible versions; for details, see [Section L.21, "Incompatible Versions of Binaries and Policy Store."](#)

The source can be a file-, LDAP-, or DB-based store; the only type of LDAP target supported is Oracle Internet Directory; the only type of DB target supported is `DB_ORACLE`. This script uses and modifies the domain configuration file `jps-config.xml`, and it is supported in only the interactive mode.

For recommendations involving reassociation, see [Important Points](#). After reassociating to a DB-based store and before using any OPSS script in offline mode, some manual editing is necessary; for details, see [Running an Offline Script after Reassociating to a DB-Based Store](#).

Interactive Mode Syntax

The script syntax varies slightly according to the type of the target store.

When the target is an LDAP-based store, use the following syntax:

```
reassociateSecurityStore(domain="domainName", servertype="OID",
ldapurl="hostAndPort", jpsroot="cnSpecification", admin="cnSpecification",
password="passWord" [,join="trueOrfalse"][,keyFilePath="dirLoc",
keyFilePassword="password"])
```

When the target is a DB-based store, use the following syntax:

```
reassociateSecurityStore(domain="domainName", servertype="DB_ORACLE",
datasourcename="datasourceName", jpsroot="jpsRoot" [,admin="adminAcct"]
[,password="passWord"][,join="trueOrfalse"])
```

The meaning of the arguments (all required) is as follows:

- `domain`: on WebLogic, specifies the domain name where the reassociating takes place; on WebSphere, specifies the WebSphere cell name.
- `admin` specifies, in case of an LDAP target, the administrator's user name on the target server, and the format is `cn=userName`.
In case of a DB target, it is required only when the DB has a protected data source (protected with user/password); in this case, it specifies the user name set to protect the data source when the data source was created; that user and password must be present in the bootstrap credential store.
- `password` specifies the password associated with the user specified for the argument `admin`. It is required in case of an LDAP target.
In case of a DB target, it is required only when the DB has a protected data source; in this case, it specifies the password associated with the user specified for the argument `admin`.
- `ldapurl` specifies the URI of the LDAP server. The format is `ldap//:host:port`, if you are using the default port, or `ldaps://host:port`, if you are using an anonymous SSL or one-way SSL transmission. The secure port must be configured to handle the desired SSL connection mode, and must be distinct from the default (non-secure) port.
- `servertype` specifies the kind of the target LDAP server or DB server. The only valid types are `OID` and `DB_ORACLE`.

- `jpsroot` specifies the root node in the target LDAP repository under which all data is migrated. The format is `cn=nodeName`.
- `join` specifies whether the domain is to share an OPSS security store in another domain. Optional. Set to `true` to share an existing store in another domain; set to `false` otherwise. If unspecified, it defaults to `false`. The use of this argument allows multiple WebLogic domains to point to the same logical OPSS security store.

Important: When an OPSS security store is reassociated with `join=true`, the bootstrap wallet from the first domain must be manually copied to the second domain. The reason for this requirement is that the first domain generates a local key that is used to encrypt the keystore data and the second domain needs to have the same key in its bootstrap wallet in order to decrypt that data.

- `datasourcename` specifies the JNDI name of the JDBC data source; this should be identical to the value of the JNDI name data source entered when the data source was created; see [Section 8.3.1.3, "Creating a Data Source Instance."](#)
- `keyFilePath` specifies the directory where the file `ewallet.p12` is created; the content of this file is encrypted and secured by the value passed to `keyFilePassword`. Optional. Use in conjunction with argument `keyFilePassword`.
- `keyFilePassword` specifies the password to secure the file `ewallet.p12`. Optional. Use in conjunction with argument `keyFilePath`.

Examples of Use

```
reassociateSecurityStore(domain="myDomain", admin="cn=adminName",
password="myPass", ldapurl="ldaps://myhost.example.com:3060", servertype="OID",
jpsroot="cn=testNode")
```

Suppose that you want some *other* domain (distinct from `myDomain`, say `otherDomain`) to share the policy store in `myDomain`. Then you would invoke the script as follows:

```
reassociateSecurityStore(domain="otherDomain", admin="cn=adminName",
password="myPass", ldapurl="ldaps://myhost.example.com:3060", servertype="OID",
jpsroot="cn=testNode", join="true")
```

9.3.30 Running an Offline Script after Reassociating to a DB-Based Store

The `jps` configuration file produced by the reassociation to a DB-based store cannot be passed, as is, to any *offline* OPSS script. Before running an OPSS script in offline mode after having reassociated to a DB-based store, the configuration file must be edited manually as described below.

The following examples illustrate fragments of `jps` configuration files before and after reassociating to a DB-based OPSS security store, and the changes required on the configuration file produced by the reassociation.

Before Reassociation

The following fragment illustrates the configuration of a file-based policy store before being reassociated to a DB-based store:

```
<serviceInstance name="policystore.xml" provider="policystore.xml.provider"
location="./system-jazn-data.xml">
```

```
<description>File Based Policy Store Service Instance</description>
</serviceInstance>
```

After Reassociation

The following fragment illustrates the property set `props.db.1` in the file generated by the reassociation of the above store to a DB-based store:

```
<propertySet name="props.db.1">
  <property value="cn=soa_domain" name="oracle.security.jps.farm.name"/>
  <property value="cn=jpsroot" name="oracle.security.jps.ldap.root.name"/>
  <property value="jdbc/opss" name="datasource.jndi.name"/>
</propertySet>

<serviceInstance provider="policystore.provider" name="policystore.db">
  <property value="DB_ORACLE" name="policystore.type"/>
  <propertySetRef ref="props.db.1"/>
</serviceInstance>
```

Required Editing

The property set above must be replaced with the following:

```
<propertySet name="props.db.1">
  <property value="cn=myDomain" name="oracle.security.jps.farm.name"/>
  <property value="DB_ORACLE" name="server.type"/>
  <property value="cn=myRoot" name="oracle.security.jps.ldap.root.name"/>
  <property name="jdbc.url" value="jdbc:oracle:thin:@myhost.com:1521/srv_name"/>
  <property name="jdbc.driver" value="oracle.jdbc.driver.OracleDriver"/>
  <property name="bootstrap.security.principal.key" value="myKeyName" />
  <property name="bootstrap.security.principal.map" value="myMapName" />
</propertySet>
```

The value of the property `jdbc.url` must match the name of the JDBC data source entered when the data source was created; the values of the bootstrap credentials (map and key) must match those passed to the OPSS script `addBootstrapCredential` when the bootstrap was created.

The edited file can then be passed to the offline script.

9.4 Caching and Refreshing the Cache

OPSS optimizes the authorization process by caching security artifacts.

When an application policy (or some other security artifact) is modified, the change becomes effective depending on where the application and the tool used to modified the artifact are running:

- If both the application and the tool are running on the same host and in the same domain, the change becomes effective immediately.
- Otherwise, if the application and the tool are running on different hosts or in different domains, the change becomes effective *after* the policy store cache is refreshed. The frequency of the cache refresh is determined by the value of the property `oracle.security.jps.ldap.policystore.refresh.interval`. The default value is 10 minutes.

9.4.1 An Example

The following use case illustrates the authorization behavior in four scenarios when (from a different domain or host) Oracle Entitlements Server is used to modify security artifacts, and the property `oracle.security.jps.policystore.refresh.interval` is set to 10 minutes.

The use case assumes that:

- A user is member of an enterprise role.
- That enterprise role is included as a member of an application role.
- The application role is granted a permission that governs some application functionality.

Under the above assumptions, we now examine the authorization result in the following four scenarios.

Scenario A

1. The user logs in to the application.
2. The user accesses the functionality secured by the application role.
3. From another host (or domain), Oracle Entitlements Server is used to remove the enterprise role from the application role.
4. The user logs out from the application, and *immediately* logs back in.
5. The user is still able to access the functionality secured by the application role.

The reason for this outcome is that the policy cache has not yet been refreshed with the change introduced in step 3 above.

Scenario B

1. The user logs in to the application.
2. The user accesses the functionality secured by the application role.
3. From another host (or domain), Oracle Entitlements Server is used to remove the enterprise role from the application role.
4. The user logs out from the application, and logs back in *after 10 minutes*.
5. The user is not able to access the functionality secured by the application role.

The reason for this outcome is that the policy cache has been refreshed with the change introduced in step 3 above.

Scenario C

1. The user logs in to the application.
2. The user accesses the functionality secured by the application role.
3. From another host (or domain), Oracle Entitlements Server is used to remove the enterprise role from the application role.
4. The user does not log out and remains able to access the functionality secured by the application role *within 10 minutes*.

The reason for this outcome is that the policy cache has not yet been refreshed with the change introduced in step 3 above.

Scenario D

1. The user logs in to the application.
2. The user accesses the functionality secured by the application role.
3. From another host (or domain), Oracle Entitlements Server is used to remove the enterprise role from the application role.
4. The user does not log out, waits *more than 10 minutes*, and then attempts to access the functionality secured by the application role: the access is denied.

The reason for this outcome is that the policy cache has been refreshed with the change introduced in step 3 above.

9.5 Granting Policies to Anonymous and Authenticated Roles with WLST Scripts

Several WLST scripts require the specification of the principal name and the principal class for a role involved in the operation.

For example, the following invocation adds a principal to the role with application stripe `myApp` and name `myAppRole`:

```
grantAppRole.py -appStripe myApp -appRoleName myAppRole
                 -principalClass myPrincipalClass -principalName myPrincipal
```

When in such scripts the principal refers to the authenticated role or the anonymous role, the principal names and principal classes are fixed and *must* be one of the following pairs:

- Authenticated role
 - Name: `authenticated-role`
 - Class:
`oracle.security.jps.internal.core.principals.JpsAuthenticatedRoleImpl`
- Anonymous role
 - Name: `anonymous-role`
 - Class:
`oracle.security.jps.internal.core.principals.JpsAnonymousRoleImpl`

The list of WLST scripts that required the above principal name and class specification are the following:

- `grantAppRole`
- `revokeAppRole`
- `grantPermission`
- `revokePermission`
- `listPermissions`

9.6 Application Stripe for Versioned Applications in WLST Scripts

Several WLST scripts require the specification of an application stripe. If the application is not versioned, the application stripe defaults to the application name.

Otherwise, if the application is versioned, the application name and the application stripe are not identical.

For example, the name of a versioned application with name `myApp` and version 1 is displayed `myApp (v1 . 0)` in Fusion Middleware Control pages, but the application stripe of this application is `myApp#v1 . 0`.

In general, an application with display name `appName (vers)` has application stripe `appName#vers`. It is this last string that should be passed as the application stripe in WLST scripts, as illustrated in the following invocation:

```
>listAppRoles myApp#v1.0
```

The list of WLST scripts that can use stripe specification are the following:

- `createAppRole`
- `deleteAppRole`
- `grantAppRole`
- `revokeAppRole`
- `listAppRoles`
- `listAppRoleMembers`
- `grantPermission`
- `revokePermission`
- `listPermissions`
- `deleteAppPolicies`

9.7 Managing Application Policies with Oracle Entitlements Server

Oracle Entitlements Server allows managing and searching application policies and other security artifacts in a WebLogic domain that uses an Oracle Internet Directory LDAP policy store.

For details, see the following topics in *Oracle Fusion Middleware Administrator's Guide for Oracle Entitlements Server*:

- Querying Security Artifacts
- Managing Policies and Roles

9.8 Guidelines to Configure the Policy Store

For details about OPSS properties tune up, see section Oracle Platform Security Services Tuning in *Oracle Fusion Middleware Performance and Tuning Guide*.

Managing the Credential Store

A credential can hold user names, passwords, and tickets; credentials can be encrypted. Credentials are used during authentication, when principals are populated in subjects, and, further, during authorization, when determining what actions the subject can perform.

Oracle Platform Security Services includes the Credential Store Framework (CSF), a set of APIs that applications can use to create, read, update, and manage credentials securely. A typical use of the credential store is to store user names and passwords to access some external system, such as a database or an LDAP-based repository.

This chapter is divided into the following sections:

- [Credential Types](#)
- [Encrypting Credentials](#)
- [Managing Credentials with Fusion Middleware Control](#)
- [Managing Credentials with OPSS Scripts](#)

10.1 Credential Types

OPSS supports the following types of credentials according to the data they contain:

- A *password* credential encapsulates a user name and a password.
- A *generic* credential encapsulates any customized data or arbitrary token, such as a symmetric key.

In CSF, a credential is uniquely identified by a map name and a key name. Typically, the map name corresponds with the name of an application and all credentials with the same map name define a logical group of credentials, such as the credentials used by the application. The pair of map and key names must be unique for all entries in a credential store.

Oracle Wallet is the default file-based credential store, and it can store X.509 certificates; production environments typically use either an Oracle Internet Directory LDAP-based or an RDBMS DB-based credential store.

10.2 Encrypting Credentials

OPSS supports storing encrypted data in file- and LDAP-based credential stores. (In case of DB-based credential stores, data is always encrypted.) OPSS uses an encryption key to encrypt and decrypt data when it is read from or written to the credential store. To enable the encryption of credentials in a file- or LDAP-based store, set the following property in the credential store service instance of the file `jps-config.xml`:

```
<property name="encrypt" value="true" />
```

By default, credentials are kept in clear-text.

The Encryption Key

Assuming the above property set, OPSS automatically generates a random 256-bit AES key when the domain is restarted. Since the keys generated are practically distinct, a domain uses a unique encryption key. In addition to the first generated encryption key, there may be other keys (roll-over keys) automatically generated over time and used to encrypt and decrypt data. The only way to get a roll-over key is by restarting the domain.

When a new roll-over key is produced, data in the credential store is not immediately re-encrypted with the new key. Instead, data is re-encrypted (with the new key) only when it is written. This implies that to get all data to use the same encryption key, all credentials must be read and written.

Domains Sharing a Credential Store

If two or more domains share a credential store and encryption is enabled in that store, then each of those domains must use the same encryption key; this applies regardless of the type, LDAP or DB, of the credential store. To ensure this, OPSS provides offline scripts to export, import, and restore keys in the domain bootstrap wallet, so that an encryption key generated in one domain can be carried over to all other domains sharing the credential store. For details about these commands, see [Managing Credentials with OPSS Scripts](#).

The following scenarios illustrate how to set credential encryption in a cluster of two domains, Domain1 and Domain2. (In case of more than two domains, treat each additional domain as Domain2 in the illustration below.)

Note: The following scenarios assume that the credential store is LDAP-based, but the use of `exportEncryptionKey` and `importEncryptionKey` to import and export keys across domains applies also to DB-based credential stores (in which data is always encrypted).

First Scenario

Assume that Domain1 has reassociated to an LDAP-based credential store, and Domain2 has *not yet joined* to that store. Then, to enable credential encryption on that store, proceed as follows:

1. Set the property `encrypt` to true in Domain1's `jps-config.xml` file and restart the domain.
2. Use the OPSS script `exportEncryptionKey` to extract the key from Domain1's bootstrap wallet into the file `ewallet.p12`; note that the value of the argument `keyFilePassword` passed to the script must be used later when importing that key into another domain.
3. Set the property `encrypt` to true in Domain2's `jps-config.xml` file.

At this point you can complete the procedure in one of two ways; both of them use the OPSS script `reassociateSecurityStore`, but with different syntaxes. For details about this script, see [Section 9.3.29, "reassociateSecurityStore."](#)

The first approach is as follows:

1. Use the OPSS script `reassociateSecurityStore` to reassociate Domain2's credential store to that used by Domain1; use the argument `join` and *do not use* the arguments `keyFilePassword` and `keyFilePath`.
2. Use the OPSS script `importEncryptionKey` to write the extracted `ewallet.p12` into Domain2's bootstrap wallet; note that the value of the argument `keyFilePassword` must be identical to the one used when the file `ewallet.p12` was generated.
3. Restart Domain2's server.

The second approach is as follows:

1. Use the OPSS script `reassociateSecurityStore` to reassociate Domain2's credential store to that used by Domain1; use the arguments `join`, `keyFilePassword`, and `keyFilePath`.
2. Restart Domain2's server.

Second Scenario

Assume that Domain1 has reassociated to an LDAP-based credential store and Domain2 has *already joined* to that store. Then, to enable credential encryption on that store, proceed as follows:

1. Set the property `encrypt` to true in Domain1's `jps-config.xml` file and restart the domain.
2. Use the OPSS script `exportEncryptionKey` to extract the key from Domain1's bootstrap wallet into the file `ewallet.p12`; note that the value of the argument `keyFilePassword` passed to the script must be used later when importing that key into another domain.
3. Set the property `encrypt` to true in Domain2's `jps-config.xml` file.
4. Use the OPSS script `importEncryptionKey` to write the extracted `ewallet.p12` into Domain2's bootstrap wallet; note that the value of the argument `keyFilePassword` must be identical to the one used when the file `ewallet.p12` was generated.
5. Restart Domain2's server.

Important Note: In case of multiple domains sharing a credential store in which encryption has been enabled, every time a roll-over key is generated in one of those domains, the administrator *must* import that key to each of the other domains in the cluster using the OPSS scripts `exportEncryptionKey` and `importEncryptionKey`.

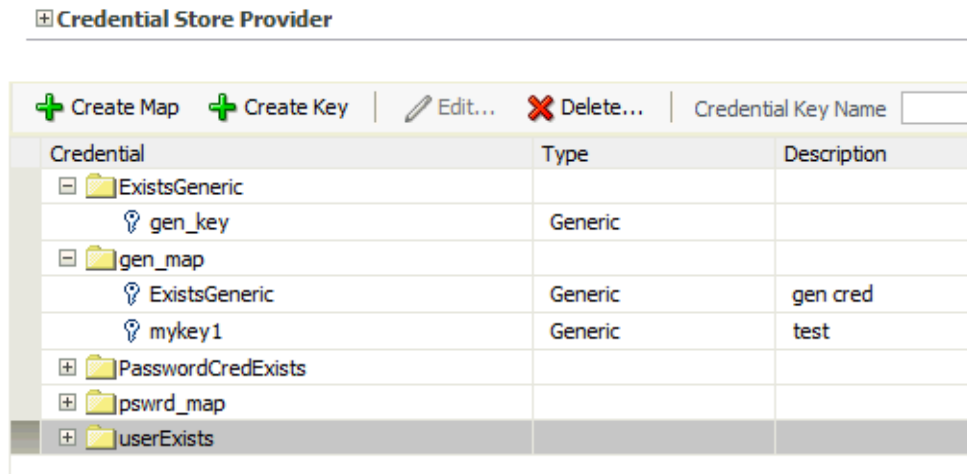
10.3 Managing the Credential Store

Credentials can be provisioned, retrieved, modified, or deleted, but only by a user in the appropriate administration role. The following sections explain how an administrator can manage credentials using Fusion Middleware Control pages or OPSS scripts, and how code can access data in the CSF.

10.4 Managing Credentials with Fusion Middleware Control

The following procedure explains how to use Fusion Middleware Control to manage credentials.

1. Log in to Fusion Middleware Control and navigate to *Domain* > **Security** > **Credentials** (if the application is deployed on Oracle WebLogic Server), or to *Cell* > **Security** > **Application Policies** (if it is deployed on WebSphere Application Server), to display the **Credentials** page partially illustrated in the following graphic:



The area **Credential Store Provider** is read-only; when expanded, it displays the credential store provider currently in use in the domain or cell.

2. To display credentials matching a given key name, enter the string to match in the **Credential Key Name** box, and then click the blue button. The result of the search is displayed in the table at the bottom of the page.
3. At any point, you can remove an item by selecting it and clicking the **Delete** button; similarly, you can modify an item from the table by selecting it and clicking **Edit** button. Note that deleting a credential map, deletes all keys in it.

To create a credential map:

1. Click **Create Map** to display the **Create Map** dialog.
2. In this dialog, enter the name of the map for the credential being created.
3. Click **OK** to return to the **Credentials** page. The new credential map name is displayed with a map icon in the table.

To add a key to a credential map:

1. Click **Create Key** to display the **Create Key** dialog.
2. In this dialog, select a map from the menu **Select Map** for the key being created, enter a key in the text **Key** box, and select a type (Password or Generic) from the pull-down menu **Type**. The dialog display changes according the type selected.

If Password was selected, enter the required fields (Key, User Name, Password, Confirm Passwords).

If Generic was selected, enter the required field Key and the credential information either as text (select **Enter as Text** radio button), or as a list of key-value pairs (select **Enter Map of Property Name and Value Pairs** radio button); to add a key-value pair, click **Add Row**, and then enter the Property Name, Value, and Confirm Value in the added arrow.

Figure 10-1 illustrates th dialog used to create a password key.

3. Click **OK** to return to the **Credentials** page. The new key is displayed under the map icon corresponding to the map you selected.

Figure 10–1 The Create Key Dialog

The screenshot shows a 'Create Key' dialog box with the following fields and controls:

- Select Map:** A dropdown menu with 'userExists' selected.
- * Key:** A text input field.
- Type:** A dropdown menu with 'Password' selected.
- * User Name:** A text input field.
- * Password:** A text input field.
- * Confirm Password:** A text input field.
- Description:** A larger text input area.
- Buttons:** 'OK' and 'Cancel' buttons at the bottom right.

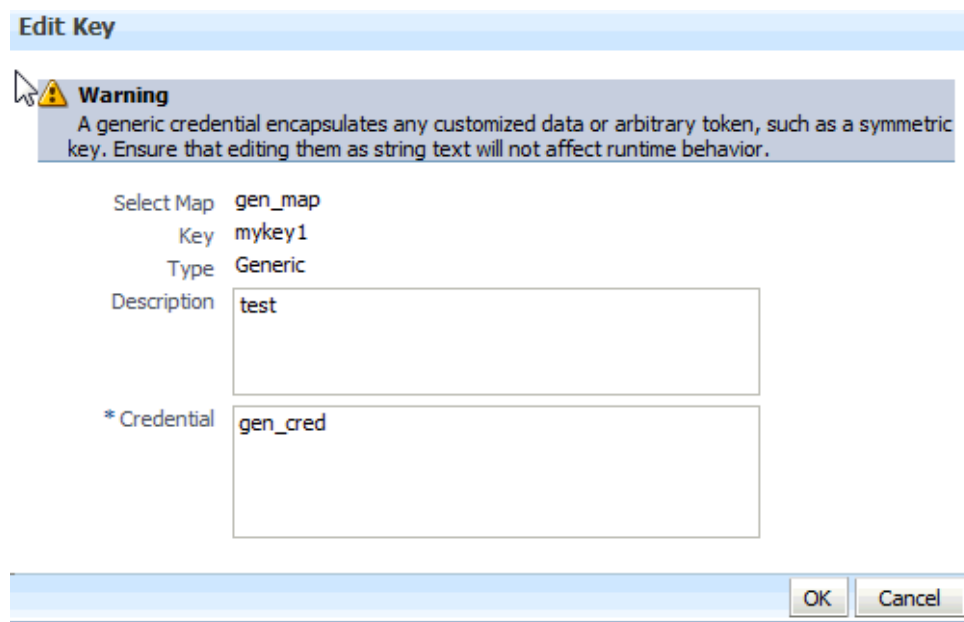
To edit a key:

1. Select a key from the table.
2. Click **Edit** to bring up the **Edit Key** dialog.
3. In that dialog, modify the key data as appropriate. In case of editing a generic key, use the red X next to a row to delete the corresponding property-value pair.

[Figure 10–2](#) illustrates the dialog used to edit a generic key.

4. Click **OK** to save your changes and return to the **Credentials** page.

For specific considerations that apply to ADF applications only, see section *How to Edit Credentials Deployed with the Application* in *Oracle Fusion Middleware Administrator's Guide for Oracle Application Development Framework*.

Figure 10–2 The Create Key Dialog

10.5 Managing Credentials with OPSS Scripts

An OPSS script is either a WLST script, in the context of the Oracle WebLogic Server, or a WASAdmin script, in the context of the WebSphere Application Server. The scripts listed in this section apply to both platforms: WebLogic Application Server and WebSphere Application Server.

An **online** script is a script that requires a connection to a running server. Unless otherwise stated, scripts listed in this section are online scripts and operate on a policy store, regardless of whether it is file-, LDAP-, or DB-based. There are a few scripts that are **offline**, that is, they do not require a server to be running to operate.

Read-only scripts can be performed only by users in the following WebLogic groups: Monitor, Operator, Configurator, or Admin. Read-write scripts can be performed only by users in the following WebLogic groups: Admin or Configurator. All WLST scripts are available out-of-the-box with the installation of the Oracle WebLogic Server.

WLST scripts can be run in interactive mode or in script mode. In interactive mode, you enter the script at a command-line prompt and view the response immediately after. In script mode, you write scripts in a text file (with a py file name extension) and run it without requiring input, much like the directives in a shell script.

WASAdmin scripts can be run in interactive mode only. For details, see *Oracle Fusion Middleware Third-Party Application Server Guide*.

For platform-specific requirements to run an OPSS script, see [Important Note](#).

OPSS provides the following scripts on all supported platforms to administer credentials (all scripts are **online**, unless otherwise stated):

- [listCred](#)
- [updateCred](#)
- [createCred](#)
- [deleteCred](#)

- [modifyBootStrapCredential](#)
- [addBootStrapCredential](#)
- [exportEncryptionKey](#)
- [importEncryptionKey](#)
- [restoreEncryptionKey](#)

10.5.1 listCred

The script `listCred` returns the list of attribute values of a credential in the credential store with given map name and key name. This script lists the data encapsulated in credentials of type password only.

Script Mode Syntax

```
listCred.py -map mapName -key keyName
```

Interactive Mode Syntax

```
listCred(map="mapName", key="keyName")
```

The meanings of the arguments (all required) are as follows:

- `map` specifies a map name (folder).
- `key` specifies a key name.

Example of Use

The following invocation returns all the information (such as user name, password, and description) in the credential with map name `myMap` and key name `myKey`:

```
listCred.py -map myMap -key myKey
```

10.5.2 updateCred

The script `updateCred` modifies the type, user name, and password of a credential in the credential store with given map name and key name. This script updates the data encapsulated in credentials of type password only. Only the interactive mode is supported.

Interactive Mode Syntax

```
updateCred(map="mapName", key="keyName", user="userName", password="passW",  
[desc="description"])
```

The meanings of the arguments (optional arguments are enclosed by square brackets) are as follows:

- `map` specifies a map name (folder) in the credential store.
- `key` specifies a key name.
- `user` specifies the credential user name.
- `password` specifies the credential password.
- `desc` specifies a string describing the credential.

Example of Use

The following invocation updates the user name, password, and description of the password credential with map name `myMap` and key name `myKey`:

```
updateCred(map="myMap", key="myKey", user="myUsr", password="myPassw")
```

10.5.3 createCred

The script `createCred` creates a credential in the credential store with a given map name, key name, user name and password. This script can create a credential of type password only. Only the interactive mode is supported.

Interactive Mode Syntax

```
createCred(map="mapName", key="keyName", user="userName", password="passw",  
[desc="description"])
```

The meanings of the arguments (optional arguments are enclosed by square brackets) are as follows:

- `map` specifies the map name (folder) of the credential.
- `key` specifies the key name of the credential.
- `user` specifies the credential user name.
- `password` specifies the credential password.
- `desc` specifies a string describing the credential.

Example of Use

The following invocation creates a password credential with the specified data:

```
createCred(map="myMap", key="myKey", user="myUsr", password="myPassw")
```

10.5.4 deleteCred

The script `deleteCred` removes a credential with given map name and key name from the credential store.

Script Mode Syntax

```
deleteCred.py -map mapName -key keyName
```

Interactive Mode Syntax

```
deleteCred(map="mapName", key="keyName")
```

The meanings of the arguments (all required) are as follows:

- `map` specifies a map name (folder).
- `key` specifies a key name.

Example of Use

The following invocation removes the credential with map name `myMap` and key name `myKey`:

```
deleteCred.py -map myMap -key myKey
```

10.5.5 modifyBootStrapCredential

The offline script `modifyBootStrapCredential` modifies the bootstrap credentials configured in the default jps context, and it is typically used in the following scenario: suppose that the policy and credential stores are LDAP-based, and the credentials to access the LDAP store (stored in the LDAP server) are changed. Then this script can be used to seed those changes into the bootstrap credential store.

This script is available in interactive mode only.

Interactive Mode Syntax

```
modifyBootStrapCredential(jpsConfigFile="pathName", username="usrName",
password="usrPass")
```

The meanings of the arguments (all required) are as follows:

- `jpsConfigFile` specifies the location of the file `jps-config.xml` relative to the location where the script is run.
- `username` specifies the distinguished name of the user in the LDAP store.
- `password` specifies the password of the user.

Example of Use

Suppose that in the LDAP store, the password of the user with distinguished name `cn=orcladmin` has been changed to `welcome1`, and that the configuration file `jps-config.xml` is located in the current directory.

Then the following invocation changes the password in the bootstrap credential store to `welcome1`:

```
modifyBootStrapCredential(jpsConfigFile='./jps-config.xml',
username='cn=orcladmin', password='welcome1')
```

Any output regarding the audit service can be disregarded.

10.5.6 addBootStrapCredential

The offline script `addBootStrapCredential` adds a password credential with given map, key, user name, and user password to the bootstrap credentials configured in the default jps context of a jps configuration file.

This script is available in interactive mode only.

Interactive Mode Syntax

```
addBootStrapCredential(jpsConfigFile="pathName", map="mapName", key="keyName",
username="usrName", password="usrPass")
```

The meanings of the arguments (all required) are as follows:

- `jpsConfigFile` specifies the location of the file `jps-config.xml` relative to the location where the script is run.
- `map` specifies the map of the credential to add.
- `key` specifies the key of the credential to add.
- `username` specifies the name of the user in the credential to add.
- `password` specifies the password of the user in the credential to add.

Example of Use

The following invocation adds a credential to the bootstrap credential store:

```
addBootstrapCredential(jpsConfigFile='./jps-config.xml', map='myMapName',
key='myKeyName', username='myUser', password='myPassword')
```

10.5.7 exportEncryptionKey

The offline script `exportEncryptionKey` extracts the encryption key from a domain's bootstrap wallet to the file `ewallet.p12`.

Interactive Mode Syntax

```
exportEncryptionKey(jpsConfigFile="pathName", keyFilePath="dirloc"
,keyFilePassword="password")
```

The meanings of the arguments (all required) are as follows:

- `jpsConfigFile` specifies the location of the file `jps-config.xml` relative to the location where the script is run.
- `keyFilePath` specifies the directory where the file `ewallet.p12` is created; note that the content of this file is encrypted and secured by the value passed to `keyFilePassword`.
- `keyFilePassword` specifies the password to secure the file `ewallet.p12`; note that this same password must be used when importing that file.

10.5.8 importEncryptionKey

The offline script `importEncryptionKey` writes an encryption key from the file `ewallet.p12` to a domain's bootstrap wallet.

Interactive Mode Syntax

```
importEncryptionKey(jpsConfigFile="pathName", keyFilePath="dirloc"
,keyFilePassword="password")
```

The meanings of the arguments (all required) are as follows:

- `jpsConfigFile` specifies the location of the file `jps-config.xml` relative to the location where the script is run.
- `keyFilePath` specifies the directory where the `ewallet.p12` is located.
- `keyFilePassword` specifies the password used when the file `ewallet.p12` was generated.

10.5.9 restoreEncryptionKey

The offline script `restoreEncryptionKey` restores the last key to a bootstrap wallet.

Interactive Mode Syntax

```
restoreEncryptionKey(jpsConfigFile="pathName")
```

The meaning of the argument (required) is as follows:

- `jpsConfigFile` specifies the location of the file `jps-config.xml` relative to the location where the script is run.

Managing Keys and Certificates with the Keystore Service

This chapter explains how to use the Keystore Service to administer keys and certificates.

- [About the Keystore Service](#)
- [About Keystore Service Commands](#)
- [Getting Help for Keystore Service Commands](#)
- [Keystore Service Command Reference](#)

11.1 About the Keystore Service

The OPSS Keystore Service enables you to manage keys and certificates for SSL, message security, encryption, and related tasks. You use the Keystore Service to create and maintain keystores that contain keys, certificates, and other artifacts.

- [Structure of the Keystore Service](#)
- [Types of Keystores](#)
- [Domain Trust Store](#)

11.1.1 Structure of the Keystore Service

Each keystore created with the Keystore Service is uniquely referenced by an application stripe and keystore:

- **Application Stripe**
Keys and certificates created in the keystore reside in an application stripe or product, and each stripe in a domain is uniquely named.
- **Keystore**
The keystore name is unique within an application stripe. Each product or application is allowed to create more than one key store within its application stripe.

Thus (appstripe1, keystoreA) is unique and distinct from (appstripe1, keystoreB), which is distinct from (appstripe2, keystoreA).

In turn, each keystore may contain the following entries, referenced by an alias that is unique within the keystore :

- Asymmetric Keys - These include the public key and the corresponding private key, and are typically used for SSL communication. The public key is wrapped in a certificate.
- Symmetric Keys - These keys are generally used for encryption.
- Trusted Certificates - These certificates are typically used to establish trust with an SSL peer.

11.1.2 Types of Keystores

The Keystore Service lets you create two types of keystores:

- Keystores protected solely by Permission
These types of key stores are protected by authorization policies and any access to them by runtime code is protected by code source permissions. The key data in the backend is encrypted using an encryption key that is generated uniquely per domain.
- Keystores protected by both Permission and Password
These types of key stores are protected both by authorization policies and key store and/or key passwords. Any access to them by runtime code requires both code source permissions as well as access to the key store and key password (if different from the key store password). The key data in the backend is encrypted using the key store/key password through password based encryption (PBE).

It is recommended that you use permission-protected keystores for applications. If you require high security and are willing to manage passwords, however, consider using keystores that are both password- and permission-protected.

Note: The Keystore Service does not manage passwords for keystore or keys. The product or application is responsible for managing them in an appropriate repository. For example, you may choose to store the passwords for your applications in a credential store.

11.1.3 Domain Trust Store

Although each application may configure multiple keystores for its SSL usage, a domain-level trust store comes pre-configured for all products and applications to use for trust management.

This domain trust store contains the trusted certificates of most well-known third-party Certificate Authorities (CAs) as well as the trusted certificate of the demo CA that is configured with the Keystore Service. Each application can simply point to this domain trust store for its SSL needs, eliminating the need to create a dedicated trust store for this task.

One-Way SSL

For one-way SSL, applications can simply use the domain trust store and do not need to create any keystore or trust store.

Two-Way SSL

For two-way SSL, applications should create only the keystore containing their identity certificate, and use the domain trust store for trust.

Note: The domain trust store is a shared store for all products and applications in a domain. The decision to add or remove trust should not be taken lightly since it may affect all other products in the domain.

Consider creating a custom trust store only if a product's trust management requirements are not met by the domain trust store.

11.2 About Keystore Service Commands

The Keystore Service uses a dedicated set of commands for keystore operations such as creating and managing keystores, exporting certificates, and generating keypairs. While their usage is similar, these commands are distinct from other OPSS commands.

The starting point for using the Keystore Service command set is `getOpssService`, which gets an OPSS service command object that enables you to:

- execute commands for the service
- obtain command help

The general syntax is:

```
variable = getOpssService(name='service_name')
```

where

- the `variable` stores the command object
- the service name refers to the service whose command object is to be obtained. The only valid value is 'KeyStoreService'.

For example:

```
svc = getOpssService(name='KeyStoreService')
```

11.3 Getting Help for Keystore Service Commands

To obtain help for any Keystore Service command, start by obtaining a service command object as explained in [Section 11.2](#). Use this object in conjunction with the help command and the command in question.

To obtain a list of all Keystore Service commands, enter:

```
svc.help()
```

To obtain help for a specific command, enter:

```
svc.help('command-name')
```

For example, the following returns help for the `exportKeyStore` command:

```
svc.help('exportKeyStore')
```

11.4 Keystore Service Command Reference

This section provides a reference to the keystore service commands, which are listed in [Table 11-1](#).

Table 11–1 Keystore Service Commands

Command	Description
<code>changeKeyPassword</code>	Changes the password for a key.
<code>changeKeyStorePassword</code>	Changes the password of a keystore.
<code>createKeyStore</code>	Creates a new keystore.
<code>deleteKeyStore</code>	Deletes the named keystore.
<code>deleteKeyStoreEntry</code>	Deletes a keystore entry.
<code>exportKeyStore</code>	Exports a keystore to file.
<code>exportKeyStoreCertificate</code>	Exports a certificate, trusted certificate, or certificate chain.
<code>exportKeyStoreCertificateRequest</code>	Generates and exports a certificate request.
<code>generateKeyPair</code>	Generates a key pair in a keystore.
<code>generateSecretKey</code>	Generates a symmetric key in a keystore.
<code>getKeyStoreCertificates</code>	Retrieves information about a certificate or trusted certificate.
<code>getKeyStoreSecretKeyProperties</code>	Retrieves secret key properties.
<code>importKeyStore</code>	Imports a keystore from a file.
<code>importKeyStoreCertificate</code>	Imports a certificate, trusted certificate or certificate chain.
<code>listExpiringCertificates</code>	Lists expiring certificates and optionally renews them.
<code>listKeyStoreAliases</code>	Lists the aliases in a keystore.
<code>listKeyStores</code>	Lists the keystores in a stripe.

11.4.1 changeKeyPassword

Description

Changes the password for a key.

Syntax

```
svc.changeKeyPassword(appStripe='stripe', name='keystore', password='password',
alias='alias', currentkeypassword='currentkeypassword',
newkeypassword='newkeypassword')
```

where:

- `svc`=the service command object obtained through a call to `getOpssService()`.
- `appStripe`= the name of the stripe containing the keystore.
- `name`= name of the keystore.
- `password`= the keystore password.
- `alias`= alias of the key entry whose password is changed.
- `currentkeypassword`= the current key password.
- `newkeypassword`= the new key password.

Example

```
svc.changeKeyPassword(appStripe='system', name='keystore', password='password',
alias='orakey', currentkeypassword='currentkeypassword',
newkeypassword='newkeypassword')
```

11.4.2 changeKeyStorePassword

Description

Changes the password of a keystore.

Syntax

```
svc.changeKeyStorePassword(appStripe='stripe', name='keystore',
currentpassword='currentpassword', newpassword='newpassword')
```

where:

- svc=the service command object obtained through a call to getOpssService().
- appStripe= the name of the stripe containing the keystore.
- name= the name of the keystore whose password is changed.
- currentpassword= current keystore password.
- newpassword= new keystore password

Example

```
svc.changeKeyStorePassword(appStripe='system', name='keystore2',
currentpassword='currentpassword', newpassword='newpassword')
```

11.4.3 createKeyStore

Description

Creates a new keystore.

Syntax

```
svc.createKeyStore(appStripe='stripe', name='keystore',
password='password',permission=true|false)
```

where:

- svc=the service command object obtained through a call to getOpssService().
- appStripe= the name of the stripe in which the keystore is created.
- name= the name of the keystore.
- password= Password of the keystore.
- permission= true if keystore is protected by permission only, false if protected by both permission and password.

Example

```
svc.createKeyStore(appStripe='system', name='keystore1',
password='password',permission=true)
```

11.4.4 deleteKeyStore

Description

Deletes the named keystore.

Syntax

```
svc.deleteKeyStore(appStripe='stripe', name='keystore', password='password')
```

where:

- `svc`=the service command object obtained through a call to `getOpssService()`.
- `appStripe`= the name of the stripe in which keystore is deleted.
- `name`= the name of the keystore to be deleted.
- `password`= password of the keystore to be deleted.

Example

```
svc.deleteKeyStore(appStripe='system', name='keystore1', password='password')
```

11.4.5 deleteKeyStoreEntry

Description

Deletes a keystore entry.

Syntax

```
svc.deleteKeyStoreEntry(appStripe='stripe', name='keystore', password='password',  
alias='alias', keypassword='keypassword')
```

where:

- `svc`=the service command object obtained through a call to `getOpssService()`.
- `appStripe`= the name of the stripe containing the keystore.
- `name`= the name of the keystore.
- `password`= the keystore password.
- `alias`= alias of the entry to be deleted.
- `keypassword`= the key password of the entry to be deleted.

Example

```
svc.deleteKeyStoreEntry(appStripe='system', name='keystore2', password='password',  
alias='orakey', keypassword='keypassword')
```

11.4.6 exportKeyStore

Description

Exports a keystore to a file.

Syntax

```
svc.exportKeyStore(appStripe='stripe', name='keystore', password='password',
aliases='comma-separated-aliases', keypasswords='comma-separated-keypasswords',
type='keystore-type', filepath='absolute_file_path')
```

where:

- svc=the service command object obtained through a call to getOpssService().
- appStripe= the name of the stripe containing the keystore.
- name= the name of the keystore.
- password= the keystore password.
- aliases= comma separated list of aliases to be exported.
- keypasswords= comma separated list of the key passwords corresponding to aliases.
- type= exported keystore type. Valid values are 'JKS' or 'JCEKS'.
- filepath= absolute path of the file where keystore is exported.

Example

```
svc.exportKeyStore(appStripe='system', name='keystore2',
password='password', aliases='orakey,seckey',
keypasswords='keypassword1,keypassword2', type='JKS', filepath='/tmp/file.jks')
```

11.4.7 exportKeyStoreCertificate

Description

Exports a certificate, trusted certificate or certificate chain.

Syntax

```
svc.exportKeyStoreCertificate(appStripe='stripe', name='keystore',
password='password', alias='alias', keypassword='keypassword',
type='entrytype', filepath='absolute_file_path')
```

where:

- svc=the service command object obtained through a call to getOpssService().
- appStripe= the name of the stripe containing the keystore.
- name= the name of the keystore.
- password= the keystore password.
- alias= alias of the entry to be exported.
- keypassword= the key password.

- `type`= type of keystore entry to be exported. Valid values are 'Certificate', 'TrustedCertificate' or 'CertificateChain'.
- `filepath`= absolute path of the file where certificate, trusted certificate or certificate chain is exported.

Example

```
svc.exportKeyStoreCertificate(appStripe='system', name='keystore2',  
password='password', alias='orakey', keypassword='keypassword',  
type='Certificate', filepath='/tmp/cert.txt')
```

11.4.8 exportKeyStoreCertificateRequest

Description

Generates and exports a certificate request.

Syntax

```
svc.exportKeyStoreCertificateRequest(appStripe='stripe', name='keystore',  
password='password', alias='alias', keypassword='keypassword', filepath='absolute_  
file_path')
```

where:

- `svc`=the service command object obtained through a call to `getOpssService()`.
- `appStripe`= the name of the stripe containing the keystore.
- `name`= the name of the keystore.
- `password`= the keystore password.
- `alias`= alias of the key pair from which certificate request is generated.
- `keypassword`= the key pair password.
- `filepath`= absolute path of the file where certificate request should be exported.

Example

```
svc.exportKeyStoreCertificateRequest(appStripe='system', name='keystore2',  
password='password', alias='orakey', keypassword='keypassword',  
filepath='/tmp/certreq.txt')
```

11.4.9 generateKeyPair

Description

Generates a key pair in a keystore and wraps it in a demo CA-signed certificate.

Syntax

```
svc.generateKeyPair(appStripe='stripe', name='keystore', password='password',  
dn='distinguishedname', keysize='keysize', alias='alias',  
keypassword='keypassword')
```

where:

- svc=the service command object obtained through a call to `getOpssService()`.
- appStripe= the name of the stripe containing the keystore.
- name= the name of the keystore where key pair is generated.
- password= the keystore password.
- dn= the distinguished name of the certificate wrapping the key pair.
- keysize= the key size.
- alias= the alias of the key pair entry.
- keypassword= the key password.

Example

```
svc.generateKeyPair(appStripe='system', name='keystore2', password='password',
dn='cn=www.oracle.com', keysize='1024', alias='orakey', keypassword='keypassword')
```

11.4.10 generateSecretKey**Description**

Generates a symmetric key in a keystore.

Syntax

```
svc.generateSecretKey(appStripe='stripe', name='keystore', password='password',
algorithm='algorithm', keysize='keysize', alias='alias',
keypassword='keypassword')
```

where:

- svc=the service command object obtained through a call to `getOpssService()`.
- appStripe= the name of the stripe containing the keystore.
- name= the name of the keystore where symmetric key is generated.
- password= the keystore password.
- algorithm= the symmetric key algorithm.
- keysize= the key size.
- alias= the alias of the key entry.
- keypassword= the key password.

Example

```
svc.generateSecretKey(appStripe='system', name='keystore2', password='password',
algorithm='AES', keysize='128', alias='seckey', keypassword='keypassword')
```

11.4.11 getKeyStoreCertificates**Description**

Retrieves information about a certificate or trusted certificate.

Syntax

```
svc.getKeyStoreCertificates(appStripe='stripe', name='keystore',  
password='password', alias='alias', keypassword='keypassword')
```

where:

- svc=the service command object obtained through a call to `getOpssService()`.
- appStripe= the name of the stripe containing the keystore.
keypassword= the key password.
- name= the name of the keystore.
- password= the keystore password.
- alias= the alias of the certificate, trusted certificate, or certificate chain to be displayed.

Example

```
svc.getKeyStoreCertificates(appStripe='system', name='keystore3',  
password='password', alias='orakey', keypassword='keypassword')
```

11.4.12 getKeyStoreSecretKeyProperties

Description

Retrieves secret key properties like the algorithm.

Syntax

```
svc.getKeyStoreSecretKeyProperties(appStripe='stripe', name='keystore',  
password='password', alias='alias', keypassword='keypassword')
```

where:

- svc=the service command object obtained through a call to `getOpssService()`.
- appStripe= the name of the stripe containing the keystore.
- name= the name of the keystore.
- password= the keystore password.
- alias= the alias of the secret key whose properties are displayed.
- keypassword= the secret key password.

Example

```
svc.getKeyStoreSecretKeyProperties(appStripe='system', name='keystore3',  
password='password', alias='seckey', keypassword='keypassword')
```

11.4.13 importKeyStore

Description

Imports a keystore from file.

Syntax

```
svc.importKeyStore(appStripe='stripe', name='keystore', password='password',
aliases='comma-separated-aliases', keypasswords='comma-separated-keypasswords',
type='keystore-type', permission=true|false, filepath='absolute_file_path')
```

where:

- svc=the service command object obtained through a call to getOpssService().
- appStripe= the name of the stripe containing the keystore.
- name= the name of the keystore.
- password= the keystore password.
- aliases= comma separated aliases of the entries to be imported from file.
- keypasswords= comma separated passwords of the keys in file.
- type= Imported keystore type. Valid values are 'JKS' or 'JCEKS'.
- filepath= absolute path of the keystore file to be imported.

Example

```
svc.importKeyStore(appStripe='system', name='keystore2',
password='password', aliases='orakey,seckey', keypasswords='keypassword1,
keypassword2', type='JKS', permission=true, filepath='/tmp/file.jks')
```

11.4.14 importKeyStoreCertificate

Description

Imports a certificate, trusted certificate or certificate chain.

Syntax

```
svc.importKeyStoreCertificate(appStripe='stripe', name='keystore',
password='password', alias='alias', keypassword='keypassword',
type='entrytype', filepath='absolute_file_path')
```

where:

- svc=the service command object obtained through a call to getOpssService().
- appStripe= the name of the stripe containing the keystore.
- name= the name of the keystore.
- password= the keystore password.
- alias= alias of the entry to be imported.
- keypassword= the key password of the newly imported entry.
- type= type of keystore entry to be imported. Valid values are 'Certificate', 'TrustedCertificate' or 'CertificateChain'.
- filepath= absolute path of the file from where certificate, trusted certificate or certificate chain is imported.

Example

```
svc.importKeyStoreCertificate(appStripe='system', name='keystore2',
```

```
password='password', alias='orakey', keypassword='keypassword',  
type='Certificate', filepath='/tmp/cert.txt')
```

11.4.15 listExpiringCertificates

Description

Lists expiring certificates and optionally renews them.

Syntax

```
svc.listExpiringCertificates(days='days', autorenew=true|false)
```

where:

- `svc`=the service command object obtained through a call to `getOpssService()`.
- `days`=only list certificates within these many days from expiration.
- `autorenew= true` for automatically renewing expiring certificates, `false` for only listing them.

Example

```
svc.listExpiringCertificates(days='365', autorenew=true)
```

11.4.16 listKeyStoreAliases

Description

Lists the aliases in a keystore for a given type of entry.

Syntax

```
svc.listKeyStoreAliases(appStripe='stripe', name='keystore', password='password',  
type='entrytype')
```

where:

- `svc`=the service command object obtained through a call to `getOpssService()`.
- `appStripe`= the name of the stripe containing the keystore.
- `name`= the name of the keystore.
- `password`= the keystore password.
- `type`= the type of entry for which aliases are listed. Valid values are 'Certificate', 'TrustedCertificate', 'SecretKey' or '*'.

Examples

```
svc.listKeyStoreAliases(appStripe='system', name='keystore2', password='password',  
type='Certificate')
```

```
svc.listKeyStoreAliases(appStripe='system', name='keystore2', password='password',  
type='TrustedCertificate')
```

```
svc.listKeyStoreAliases(appStripe='system', name='keystore2', password='password',
```

```
type='SecretKey')  
  
svc.listKeyStoreAliases(appStripe='system', name='keystore2', password='password',  
type='*')
```

11.4.17 listKeyStores

Description

Lists all the keystores in a stripe.

Syntax

```
svc.listKeyStores(appStripe='stripe')
```

where:

- `svc`=the service command object obtained through a call to `getOpssService()`.
- `appStripe`= the name of the stripe whose keystores are listed.

Examples

```
svc.listKeyStores(appStripe='system')
```

```
svc.listKeyStores(appStripe='*')
```

Introduction to Oracle Fusion Middleware Audit Framework

In Oracle Fusion Middleware 11g Release 1 (11.1.1), auditing provides a measure of accountability and answers the "who has done what and when" types of questions. This chapter introduces auditing in Oracle Fusion Middleware. It contains the following topics:

- [Benefits and Features of the Oracle Fusion Middleware Audit Framework](#)
- [Overview of Audit Features](#)
- [Oracle Fusion Middleware Audit Framework Concepts](#)

12.1 Benefits and Features of the Oracle Fusion Middleware Audit Framework

This section contains these topics:

- [Objectives of Auditing](#)
- [Today's Audit Challenges](#)
- [Oracle Fusion Middleware Audit Framework in 11g](#)

12.1.1 Objectives of Auditing

With compliance becoming an integral part of any business requirement, audit support is also becoming a focus in enterprise deployments. Customers are looking for application vendors to provide out-of-the-box audit support. In addition, middleware customers who are deploying custom applications would like to centralize the auditing of their deployed applications wherever audit is appropriate.

IT organizations are looking for several key audit features driven by compliance, monitoring, and analytics requirements.

Compliance

Compliance is obviously a major requirement in the enterprise. With regulations such as Sarbanes-Oxley (financial) and Health Insurance Portability and Accountability Act (healthcare), many customers must now be able to audit on identity information and user access on applications and devices. These include events like:

- User profile change
- Access rights changes
- User access activity

- Operational activities like starting and stopping applications, upgrades, and backups

This allows compliance officers to perform periodic reviews of compliance policies.

Monitoring

The audit data naturally provides a rich set of data for monitoring purpose. In addition to any log data and component metrics that are exposed, audit data can be used to create dashboards and to build Key Performance Indicators (KPIs) for alerts to monitor the health of the various systems on an ongoing basis.

Analytics

Audit data can also be used in assessing the efficacy of controls through analysis on the audit data. The data can also be used for risk analysis. Based on historical data, a risk score can be calculated and assigned to any user. Any runtime evaluation of user access can include the various risk scores as additional criteria to protect access to the systems.

12.1.2 Today's Audit Challenges

To satisfy the audit requirements, IT organizations often battle with the deficiencies in audit support for their deployed applications. There is no reliable standard for:

- Audit Record Generation
- Audit Record Format and Storage
- Audit Policy Definition

As a result, today's audit solutions suffer from a number of key drawbacks:

- There is no centralized audit framework.
- The quality of audit support is inconsistent from application to application.
- Audit data is scattered across the enterprise.
- Complex data correlation is required before any meaningful cross-component analysis can be conducted.
- Audit policies and their configurations are also scattered.

These factors are costing IT organization considerable amount of time and resources to build and maintain any reasonable audit solutions. With the data scattered among individual silos, and the lack of consistency and centralization, the audit solutions also tend to be fragile with idiosyncrasies among applications from different vendors with their current audit capabilities.

12.1.3 Oracle Fusion Middleware Audit Framework in 11g

Oracle Fusion Middleware Audit Framework, introduced in 11g Release 1 (11.1.1), is designed to provide a centralized audit framework for the middleware family of products. The framework provides audit service for the following:

- **Middleware Platform** - This includes Java components such as Oracle Platform Security Services (OPSS) and Oracle Web Services. These are components that are leveraged by applications deployed in the middleware. Indirectly, all the deployed applications leveraging these Java components will benefit from the audit framework auditing events that are happening at the platform level.

- Java EE applications - The objective is to provide a framework for Java EE applications, including Oracle's own Java EE-based components. Java EE applications are able to create application-specific audit events.
- System Components - For system components in the middleware that are managed by Oracle Process Manager and Notification Server, the audit framework also provides an end-to-end structure similar to that for Java components.

See Also: Understanding Key Oracle Fusion Middleware Concepts in the *Oracle Fusion Middleware Administrator's Guide*.

12.2 Overview of Audit Features

Key features of the Oracle Fusion Middleware Audit Framework include:

- A uniform system for administering audits across a range of Java components, system components, and applications
- Extensive support for Java component auditing, which includes:
 - support for Oracle Platform Security Services auditing for non-audit-aware applications
 - the ability to search for audit data at any application level
- Capturing authentication history/failures, authorization history, user management, and other common transaction data
- Flexible audit policies
 - pre-seeded audit policies, capturing customers' most common audit events, are available for ease of configuration
 - tree-like policy structure simplifies policy setup
- Prebuilt compliance reporting features
 - Oracle Fusion Middleware Audit Framework provides out-of-the-box analytical reporting capabilities within Oracle BI Publisher; data can be analyzed on multiple dimensions (Execution Context ID (ECID), user ID, and so on) across multiple components. These reports can also be customized according to your preferences.
 - Reports are based on centralized audit data.
 - Customers can customize the reports or write their own based on the published audit schema.

See [Chapter 14, "Using Audit Analysis and Reporting"](#) for details.

- Audit record storage

Audit data store (database) and files (bus-stop) are available. Maintaining a common location for all audit records simplifies maintenance.

Using an audit data store lets you generate reports with Oracle Business Intelligence Publisher.

- Common audit record format

Highlights of the audit trail include:

- baseline attributes like outcome (status), event date-time, user, and so on
- event-specific attributes like authentication method, source IP address, target user, resource, and so on

- contextual attributes like the execution context ID (ECID), session ID, and others
- Common mechanism for audit policy configuration
Oracle Fusion Middleware Audit Framework offers a unified method for configuring audit policies in the domain.
- Leverages the Oracle Fusion Middleware 11g infrastructure
 - is usable across Oracle Fusion Middleware 11g components and services such as Oracle Web Services Manager, Oracle Internet Directory, Oracle Virtual Directory, and Oracle Directory Integration and Provisioning
 - integrates with Oracle Enterprise Manager Fusion Middleware Control for UI-based configuration and management
 - integrates with `wlst` for command-line, script-based configuration
 - leverages the SPI infrastructure of Oracle Platform Security Services
- Utilizes a new dynamic metadata model in 11g Release 1 (11.1.1) Patch Set 5 to enable applications to integrate with the audit framework:
 - applications can register with the audit service at any time
 - simplifies the ability of applications to leverage the audit framework to define and log audit events
 - provides versioning of event definitions and enables audit clients to upgrade definitions independent of release cycles.

12.3 Oracle Fusion Middleware Audit Framework Concepts

This section introduces basic concepts of the Oracle Fusion Middleware Audit Framework:

- [Audit Architecture](#)
- [Key Technical Concepts](#)
- [Audit Metadata Storage](#)
- [Audit Data Storage](#)
- [Analytics](#)

12.3.1 Audit Architecture

The Oracle Fusion Middleware Audit Framework consists of the following key components:

- Audit APIs
These are APIs provided by the audit framework for any audit-aware components integrating with the Oracle Fusion Middleware Audit Framework. During runtime, applications may call these APIs where appropriate to audit the necessary information about a particular event happening in the application code. The interface allows applications to specify event details such as username and other attributes needed to provide the context of the event being audited.
- The audit framework provides these APIs:
 - audit service API

- audit client API
- **Audit Events and Configuration**

The Oracle Fusion Middleware Audit Framework provides a set of generic events for convenient mapping to application audit events. Some of these include common events such as authentication. The framework also allows applications to define application-specific events.

These event definitions and configurations are implemented as part of the audit service in Oracle Platform Security Services. Configurations can be updated through Enterprise Manager (UI) and WLST (command-line tool)
- **The Audit Bus-stop**

Bus-stops are local files containing audit data records before they are pushed to the audit data store. In the event that no audit data store is configured, audit data remains in these bus-stop files. The bus-stop files are simple text files that can be queried easily to look up specific audit events. When an audit data store is in place, the bus-stop acts as an intermediary between the component and the audit data store. The local files are periodically uploaded to the audit data store based on a configurable time interval.

A key advantage of the audit data store is that audit data from multiple components can be correlated and combined in reports, for example, authentication failures in all middleware components, instances and so on.
- **Audit Loader**

As its name implies, the audit loader loads audit data from the audit bus-stop into the audit data store, if one is configured. For Java component auditing, the audit loader is a startup class that is started as part of the container start-up. For system components, the audit loader is a periodically spawned process that is invoked by OPMN.
- **Audit Data Store**

The audit data store is a database that contains a pre-defined Oracle Fusion Middleware Audit Framework schema, created by Repository Creation Utility (RCU). Once configured, all the audit loaders are aware of the audit data store and upload data to it periodically. The audit data in the store is expected to be cumulative and will grow overtime. Ideally, this should not be an operational database used by any other applications - rather, it should be a standalone RDBMS used for audit purposes only.

The audit database can store audit events generated by Oracle components as well as user applications integrated with the audit framework.
- **Audit Metadata Store**

The audit metadata store contains audit event definitions for components and applications.
- **Audit Configuration Mbeans**

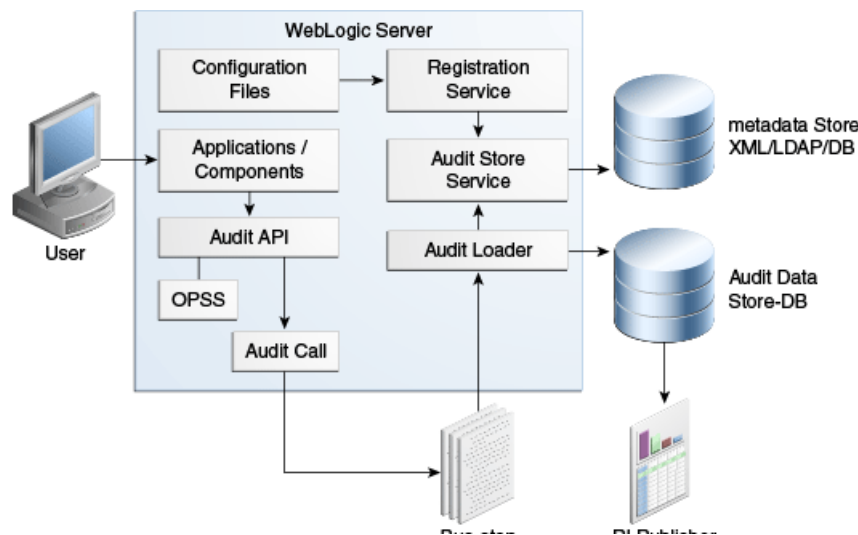
All audit configuration is managed through audit configuration MBeans. For Java components and applications, these MBeans are present in the domain administration server and the audit configuration is centrally managed. For system components, separate MBean instances are present for every component instance. Enterprise Manager UI and command-line tools manage Audit configuration using these MBeans.
- **Oracle Business Intelligence Publisher**

The data in the audit data store is exposed through pre-defined reports in Oracle Business Intelligence Publisher. The reports allow users to drill down the audit data based on various criteria. For example:

- Username
- Time Range
- Application Type
- Execution Context Identifier (ECID)

You can also use Oracle Business Intelligence Publisher to create your own audit reports.

Figure 12-1 Audit Event Flow



Audit Flow

The process can be illustrated by looking at the actions taken in the framework when an auditable event (say, login) occurs within an application server instance:

Note: The architecture shown in [Figure 12-1](#) contains an audit data store; if your site did not configure an audit data store, the audit records reside in the bus-stop files.

1. During application deployment or audit service start-up, a client such as a Java EE application or Oracle component registers with the audit service.
2. The service reads the application's pre-configured audit definition file and updates the metadata store with the audit definitions.
3. When a user accesses the component or application, an audit API function is called to audit the event.
4. The audit framework checks if events of this type, status, and with certain attributes need to be audited.
5. If so, the audit function is invoked to create the audit event structure and collect event information like the status, initiator, resource, ECID, and so on.

6. The event is stored on a local file in an intermediate location known as the bus-stop; each component has its own bus-stop.
7. If a database is configured for an audit store, the audit loader pulls the events from the bus-stops, uses the application's metadata to format the data, and moves the data to the audit store.
8. Reports can also be generated from the audit data using Oracle BI Publisher. A set of pre-defined reports are available. (See [Chapter 14, "Using Audit Analysis and Reporting"](#).)

Application Behavior in Case of Audit Failure

It is important to note that an application does not stop execution if it is unable to record an audit event for any reason.

12.3.2 Key Technical Concepts

This section introduces key concepts in the Oracle Fusion Middleware Audit Framework.

Audit-Aware Components

The term "audit-aware" refers to components that are integrated with the Oracle Fusion Middleware Audit Framework so that audit policies can be configured and events can be audited for those components. Oracle Internet Directory is an example of an audit-aware component.

Stand-alone applications can integrate with the Oracle Fusion Middleware Audit Framework through configuration with the `jps-config.xml` file. For details, see [Chapter 28](#).

Audit Metadata Store

The audit metadata store contains audit event definitions for components as well as applications integrated with the audit framework.

Audit Data Store

The audit data store is the repository for audit event data.

Note: The metadata store is separate from the audit data store.

Audit Loader

The Audit Loader is a module of the Oracle WebLogic Server instance and provides process control for that instance. The audit loader is responsible for collecting the audit records for all components running in that instance and loading them to the audit data store.

Audit Policy

An audit policy is a declaration of the type of events to be captured by the audit framework for a particular component. For Java components, the audit policy is defined at the domain level. For system components, the audit policy is managed at the component instance level.

Oracle Fusion Middleware Audit Framework provides several pre-defined policy types:

- None

- Low (audits fewer events, definition is component-dependent)
- Medium (audits many events, definition is component-dependent)
- Custom (implements filters to narrow the scope of audited events)

Audit Policy Component Type

This refers to the component type to be audited; for example, Oracle Internet Directory is a source of auditable events during authentication.

For lists of the events that can be audited for each component, see [Section C.1, "Audit Events"](#).

Event Filters

Certain audit events implement filters to control when the event is logged. For example, a successful login event for the Oracle Internet Directory component may be filtered for specific users.

For details, see [Section 13.3, "Managing Audit Policies"](#).

Oracle Platform Security Services

Oracle Platform Security Services, a key component of the Oracle Fusion Middleware 11g, is the Oracle Fusion Middleware security implementation for Java features such as Java Authentication and Authorization Service (JAAS) and Java EE security.

For more information about OPSS, see [Section 1.1, "What is Oracle Platform Security Services?"](#).

12.3.3 Audit Metadata Storage

Audit metadata refers to information about audit events, their attributes and categories.

For details, see [Chapter 28](#).

12.3.4 Audit Data Storage

As shown in [Figure 12–1](#), audit data can reside in two types of storage:

- bus-stop files for intermediate storage of audit data. Each component instance writes to its own bus-stop.

Bus-stop files are the default out-of-the-box storage mechanism for audit records:

- For Java components, there is one bus-stop for each Oracle WebLogic Server instance. Audit records generated for all Java EE components running in a given Oracle WebLogic Server instance are stored in the same bus-stop.
- For system components, there is a separate bus-stop for each component; thus, for example, each instance of Oracle Internet Directory has its own bus-stop.

Bus-stop files are text-based and easy to query. For further details, see [Section 12.3.1, "Audit Architecture"](#)

- permanent storage in a database; this is known as the audit data store.

If using a database, audit records generated by all components in all Oracle Fusion Middleware 11g instances in the domain are stored in the same store. You must use an audit data store to utilize Oracle Business Intelligence Publisher reports.

You can move from file-based storage to an audit data store. This requires a specific configuration procedure. See [Section 13.2.3, "Configure a Database Audit Data Store for Java Components"](#) for details.

Advantages of Using a Database Store

Having the audit records in the bus-stop files has some practical limitations:

- you cannot view domain-level audit data
- reports cannot be run on Oracle BI Publisher

Thus, there are certain advantages to using a database audit data store:

- You can use Oracle Business Intelligence Publisher for reporting.
- The database store centralizes records from all components in the domain, whereas the bus-stop stores audit records on a per-instance basis.
- performance may be improved compared to file-based storage

For these reasons, Oracle recommends that customers switch to a database store for enhanced auditing capabilities.

12.3.5 Analytics

With Oracle Fusion Middleware 11g, you can utilize Oracle Business Intelligence as a full-featured tool for structured reporting.

A large number of pre-defined reports are available, such as:

- Users created/deleted
- User transactions
- Authentication and authorization failures
- Policy violations

With Oracle Business Intelligence:

- You can select records based on criteria like username, date-time range, and so on.

Note that Oracle Business Intelligence works with the database audit store only, and is not usable with bus-stop files.



Oracle BI Publisher page

The pre-defined audit report types available with Oracle Business Intelligence include:

- errors and exceptions
- operational
- user activity
- authentication and authorization history
- transaction history

For further details, see [Section C.2, "Pre-built Audit Reports."](#) You can also use the audit schema details to create custom audit reports as needed.

Configuring and Managing Auditing

This chapter explains how to perform day-to-day audit administration tasks.

See Also: [Chapter 12, "Introduction to Oracle Fusion Middleware Audit Framework"](#) for background information about auditing in Oracle Fusion Middleware.

- [Audit Administration Tasks](#)
- [Managing the Audit Data Store](#)
- [Managing Audit Policies](#)
- [Audit Logs](#)
- [Advanced Management of Database Store](#)

13.1 Audit Administration Tasks

The audit administrator should plan the site's audit setup carefully by following the steps in these areas:

- **Implementation Planning**

This includes planning the type of store to use for audit records, data store configuration details, and so on.

See [Section 13.2, "Managing the Audit Data Store"](#) for details.
- **Policy administration**

The administrator must configure the appropriate audit policies to ensure that the required audit events are generated.

This is an ongoing activity since the audit policies must be able to reflect changes to the application environment, addition of components and users, and so on.

See [Section 13.3, "Managing Audit Policies"](#) for details.
- **Reports Management**

This includes planning for and configuring audit reports and queries.

See [Chapter 14, "Using Audit Analysis and Reporting"](#) for details.
- **Data Administration**

This includes planning/increasing the database size required to store the audit data generated, backing up the audit data and purging the audit data based on company policy.

See [Section 13.5, "Advanced Management of Database Store"](#) for details about audit data store administration.

13.2 Managing the Audit Data Store

Out of the box, the audit framework uses the file system to store audit records. In a production environment, however, Oracle recommends that you use a database audit data store to provide scalability and high-availability for the audit framework.

In addition, an audit data store residing in a database allows the audit data to be viewed through Oracle Business Intelligence Publisher with pre-packaged audit reports that are available with that product. Oracle Business Intelligence Publisher is available in the 11g Release 1 (11.1.1) CD pack.

This section explains these audit data store management tasks in detail:

- [Create the Audit Schema using RCU](#)
- [Set Up Audit Data Sources](#)
- [Configure a Database Audit Data Store for Java Components](#)
- [Configure a Database Audit Data Store for System Components](#)
- [Tuning the Bus-stop Files](#)
- [Configuring the Stand-alone Audit Loader](#)

13.2.1 Create the Audit Schema using RCU

To switch to a database as the permanent store for your audit records, you first use the Repository Creation Utility (RCU) to create a database store for audit data.

Note: The bus-stop files store audit records in the absence of database storage.

This section explains how to create the audit schema. Once the database schema is created, you can:

- create a datasource to point to this schema
- update the domain configuration to switch the audit data store for audit records (see [Section 13.2.3.2, "Configure the Audit Data Store"](#)).

Note: This discussion assumes that RCU and the database is already installed in your environment. See the Installation Guide for more information.

Before You Begin

Before you begin, make sure to collect the details on which database to use, along with the DBA credentials to use.

Configuring the Database Schema

Take these steps to configure a schema for the audit data store:

1. Go to `$RCU_HOME/bin` and execute the RCU utility.
2. Choose **Create** at the starting screen. Click **Next**.

3. Enter your database details and click **Next**.
4. Choose the option to create a new prefix, for example `IDM`.
5. Also, select 'Audit Services' from the list of schemas.
6. Click **Next** and accept the tablespace creation.
7. Check for any errors while the schemas are being created.

This process will take several minutes to complete.

13.2.2 Set Up Audit Data Sources

As explained in [Section 13.2.1, "Create the Audit Schema using RCU"](#), after you create a database schema to store audit records in a database, you must set up an Oracle WebLogic Server audit data source that points to that schema.

Take these steps to set up an audit data source:

Note: This task is performed with the Oracle WebLogic Server administration console.

1. Connect to the Oracle WebLogic Server administration console:
`http://host:7001/console`
2. Under JDBC, click the Data Sources link.
3. The Data Sources page appears. Click **New** to create a new data source.
4. Enter the following details for the new data source:
 - **Name:** Enter a name such as `Audit Data Source-0`.
 - **JNDI Name:** `jdbc/AuditDB`
 - **Database Type:** Oracle
 - **Database Driver:** Oracle's Driver (Thin XA) Versions: 9.0.1, 9.0.2, 10, 11

If deploying to a managed cluster server, also check **AdminServer**; this ensures that the data source is listed in the audit data store when switching from file to database store.

Click **Next**.

5. The Transaction Options page appears. Click **Next**.
6. The Connection Properties page appears. Enter the following information:
 - **Database Name:** Enter the name of the database to which you will connect. This usually maps to the `SID`.
 - **Host Name:** Enter the hostname of the database.
 - **Port:** Enter the database port.
 - **Database User Name:** This is the name of the audit schema that you created in RCU. The suffix is always `IAU` for the audit schema. For example, if you gave the prefix as `test`, then the schema name is `test_iau`.
 - **Password:** This is the password for the audit schema that you created in RCU.

Click **Next**.

7. The next page lists the JDBC driver class and database details. Accept the defaults, and click **Test Configuration** to test the connection. If you see the message "Connection established Successfully", click **Next**. If it displays any error, go back and check the connection details.
8. In the Select Targets page, select the servers where this data source needs to be configured, and click **Finish**.

13.2.2.1 Multiple Data Sources

For scalability and high availability, you can configure Oracle Real Application Clusters for your audit data.

For details, see:

- Setting Up Auditing with a RAC Database Store in the *Oracle Fusion Middleware High Availability Guide*
- Using WebLogic Server to Configure Audit Data Sources and Multi Data Sources in the *Oracle Fusion Middleware High Availability Guide*
- Configuring the JDBC String for the Audit Loader in the *Oracle Fusion Middleware High Availability Guide*
- Using WebLogic Server with Oracle RAC in *Oracle Fusion Middleware Configuring and Managing JDBC for Oracle WebLogic Server*

13.2.3 Configure a Database Audit Data Store for Java Components

After the schema is created, configuring a database-based audit data store involves:

- creating a data source that points to the audit schema you created, and
- configuring the audit data store to point to the data source

This section describes the following tasks related to audit data store configuration:

- [View Audit Data Store Configuration](#)
- [Configure the Audit Data Store](#)

Note:

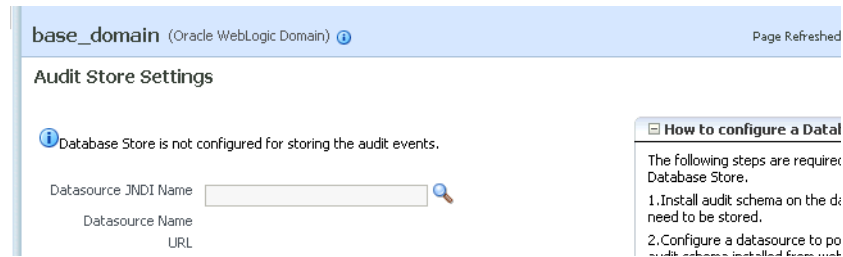
These steps configure the audit data store for Java components only. Separate steps are needed to configure the audit data store for system components. See [Section 13.2.4, "Configure a Database Audit Data Store for System Components"](#).

By configuring the same database to store audit records for Java components and system components, you can ensure that reports for both types of components can be viewed together.

13.2.3.1 View Audit Data Store Configuration

Note: This task is performed with Oracle Enterprise Manager Fusion Middleware Control.

To view the current audit data store configuration, navigate to *Domain*, then **Security**, then **Audit Store**.



Audit store configuration

This page shows:

- whether or not a database is configured as the audit data store. By default a database is not configured, and audit records are stored in bus-stop files.
- Datasource JNDI Name - If a database store is configured for audit records, this field shows the JNDI name of the datasource. This field is empty when the audit data store is not configured.
- Datasource Name - If a database store is configured for audit records, this field shows the datasource name. This field is not displayed when the audit data store is file-based.
- URL - If a database repository is configured for audit records, this field shows the data source URL, which is the connect string used to connect to the database. This field is not displayed when the audit data store is file-based.

See [Section 13.2.2, "Set Up Audit Data Sources"](#) for datasource examples.

13.2.3.2 Configure the Audit Data Store

You can change from storing audit records in a file to using a database audit data store.

Take these steps to configure the audit data store:

1. Navigate to *Domain*, then **Security**, then **Audit Store**. The Audit Store page appears.
2. Click the searchlight icon next to the Datasource JNDI Name field.
3. A dialog box appears showing the list of datasources available for audit records in the domain. Select the desired datasource and click **OK**.
4. The selected datasource is displayed in the Datasource JNDI Name field. Click **Apply** to continue, or **Revert** to abandon the update.

Note: You can also use the WLST `setAuditRepository()` command to change the audit data store settings. See Appendix D, *Fusion Middleware Audit Framework Reference* for details.

5. Restart all the Oracle WebLogic Servers in the domain. This enables Audit Loader Startup Class present in Oracle WebLogic Server to re-read the configuration.
6. You can test the changes by setting an audit policy to test event collection. For example, you can set the Medium audit policy for Oracle Platform Security Services. For details, see [Section 13.3.1, "Manage Audit Policies for Java Components with Fusion Middleware Control"](#).

7. Execute a scenario so that auditing can generate an audit event. For example, creating a credential will trigger an audit record based on the policy you configured in Step 6.
8. Check for errors and exceptions in the server logs
 - Check `$DOMAIN_HOME/jrfServer_admin.out`
 - Check `$DOMAIN_HOME/servers/$SERVER_NAME/logs/`.

13.2.3.3 Deconfigure the Audit Data Store

Since a database is the recommended store for audit records, switching from database to file mode is discouraged. However, [Section 13.3.4, "Manage Audit Policies Manually"](#) discusses a property called the `audit.repositoryType` whose value can be set to 'File' to switch to file storage.

Note: You cannot use Fusion Middleware Control or WLST to switch from database to file mode; this requires manual configuration as explained in [Section 13.3.4, "Manage Audit Policies Manually"](#).

When you switch from database to file, events that were collected in the database are not transferred back to the file system. If this switch is temporary, then the audit events collected in the file are automatically pushed to database when you switch to database store again.

13.2.4 Configure a Database Audit Data Store for System Components

Oracle Process Manager and Notification Server (OPMN) manages several system components running in Oracle WebLogic Server. For these components, the mechanism through which the audit events are pushed from local bus-stop files to the database audit data store is handled by OPMN.

Note: If your system component runs in a clustered deployment, you must configure the audit data store at each instance of the component so that all instances push out records to the store.

You must execute the following steps in every instance of the component to configure an audit data store:

Note:

These steps configure the audit data store for system components only. Separate steps are needed to configure the audit data store for Java components. See [Section 13.2.3, "Configure a Database Audit Data Store for Java Components"](#).

By configuring the same database to store audit records for Java components and system components, you can ensure that reports for both types of components can be viewed together.

1. Open the `opmn.xml` file, which resides in
`$ORACLE_INSTANCE/config/OPMN/opmn/opmn.xml`

2. Locate the `rmd-definitions` element, which looks like this:

```
<rmd-definitions>
  <rmd name="AuditLoader" interval="15">
    <conditional>
      <![CDATA[({time}>=00:00)]]>
    </conditional>
    <action value="exec $ORACLE_HOME/jdk/bin/java -classpath
$COMMON_COMPONENTS_HOME/modules/oracle.osdt_11.1.1/osdt_cert.jar:
$COMMON_COMPONENTS_HOME/modules/oracle.osdt_11.1.1/osdt_core.jar:
$ORACLE_HOME/jdbc/lib/ojdbc5.jar:
$COMMON_COMPONENTS_HOME/modules/oracle.iau_11.1.1/fmw_audit.jar:
$COMMON_COMPONENTS_HOME/modules/oracle.pki_11.1.1/oraclepki.jar
-Doracle.home=$ORACLE_HOME
-Doracle.instance=$ORACLE_INSTANCE
-Dauditloader.jdbcString=jdbc:oracle:thin:@host:port:sid
-Dauditloader.username=username
oracle.security.audit.ajl.loader.StandaloneAuditLoader"/>
    <exception value="exec /bin/echo
PERIODICAL CALL For Audit Loader FAILED"/>
  </rmd>
</rmd-definitions>
```

3. Replace the existing RMD definition for audit loader; you need to modify *only* these values:

- `jdbcString` - this is the database JDBC connection string; change this from the default string to a valid connection string.
- `username`
- `interval` - this is the interval in seconds at which audit records are pushed from the component's bus-stop file to the audit data store.

By default the interval value is set very high (31536000 seconds) so that the audit loader is effectively disabled. Change this to a reasonable interval such as 15 seconds.

Note: Insert these lines after the `<ias-instance>` tag is closed.

4. Save and exit the file.

5. Ensure that `ORACLE_HOME`, `ORACLE_INSTANCE`, and `COMMON_COMPONENTS_HOME` are defined. For example:

```
ORACLE_HOME = /u01/oracle/as11_oh
ORACLE_INSTANCE = /u01/oracle/instances/instance
COMMON_COMPONENTS_HOME = $MW_HOME/oracle_common
```

6. Populate the audit data store password in the secret store. This is the password that you have specified when creating the audit schema in RCU:

```
ORACLE_HOME/jdk/bin/java -classpath
$COMMON_COMPONENTS_HOME/modules/oracle.osdt_11.1.1/osdt_cert.jar:
$COMMON_COMPONENTS_HOME/modules/oracle.osdt_11.1.1/osdt_core.jar:
$ORACLE_HOME/jdbc/lib/ojdbc5.jar:
$COMMON_COMPONENTS_HOME/modules/oracle.iau_11.1.1/fmw_audit.jar:
$COMMON_COMPONENTS_HOME/modules/oracle.pki_11.1.1/oraclepki.jar
-Doracle.home=$ORACLE_HOME -Doracle.instance=$ORACLE_INSTANCE
-Dauditloader.jdbcString=jdbc:oracle:thin:@host:port:sid
-Dauditloader.username=username
```

```
-Dstore.password=true
-Dauditloader.password=password
oracle.security.audit.ajl.loader.StandaloneAuditLoader
```

Enter the appropriate values for `jdbcString`, `username`, `password`.

Note: The above syntax is relevant to Linux. For Windows, substitute ":" with ";" to separate the jars in the classpath.

7. Reload OPMN:

```
$ORACLE_INSTANCE/bin/opmnctl validate (Validation step to verify edits)
$ORACLE_INSTANCE/bin/opmnctl reload
```

8. Execute a scenario in an audited component to generate an audit event.

9. Check for errors/events uploaded at `$ORACLE_INSTANCE/diagnostics/logs/OPMN/opmn/rmd.out`. The output will look like this

```
8/08/26 10:54:24 global:AuditLoader
```

13.2.4.1 Deconfigure the Audit Data Store

Since a database is the recommended store for audit records, switching from database to file mode is discouraged. However, if needed, you can use the same steps that were shown in the preceding task for configuring the audit data store through the `opmn.xml` file to update the RMD definition to deconfigure the audit data store. Locate the `rmd-definitions` element and replace the existing RMD definition for audit loader:

Note: If your system component runs in a clustered deployment, you must deconfigure the audit data store at each instance of the component.

- `jdbcString` - Change the database JDBC connection string back to the default string `jdbc:oracle:thin:@host:port:sid`.
- `interval` - Set this interval back to the default value of 31536000.

Save and exit the file, and reload OPMN.

13.2.5 Tuning the Bus-stop Files

This section contains topics related to maintaining file-based storage of audit records, including:

- bus-stop file locations
- file size
- directory size

Note: Manually purging audit files to free up space is not recommended. Instead, use file and directory sizing features to control space, as described below.

Location of Bus-stop Files

Bus-stop files for Java components are located in:

```
$DOMAIN_HOME/servers/$SERVER_NAME/logs/auditlogs/Component_Type
```

Bus-stop files for system components are located in:

```
$ORACLE_INSTANCE/auditlogs/Component_Type/Component_Name
```

File Size

Java Components

The size of a file for the file storage mode can be managed using the `max.fileSize` property described in the configuration file `jps-config.xml`. This property controls the maximum size of a bus-stop file for Java components.

Specify the sizes in bytes as described in [Section 13.3.4, "Manage Audit Policies Manually"](#).

System Components

The size of a file for the file storage mode can be set in the `auditconfig.xml` file. See [Section 13.3.4.4, "Manually Configuring Audit for System Components"](#).

Note: If you switch from file to database store for audit data, all the events collected in the audit files are pushed into the database tables and the audit files are deleted.

Directory Size

Java Components

The size of a directory for the file can be managed using the `max.DirSize` property described in the configuration file `jps-config.xml`. This property controls the maximum size of a bus-stop directory.

Specify the sizes in bytes as described in [Section 13.3.4, "Manage Audit Policies Manually"](#).

System Components

The size of a directory for the file storage mode can be set in the `auditconfig.xml` file. See [Section 13.3.4.4, "Manually Configuring Audit for System Components"](#).

13.2.6 Configuring the Stand-alone Audit Loader

As shown in [Figure 12–1](#), Common Audit Framework's audit loader moves records from bus-stop files to the audit data store. The mechanism driving the audit loader depends on the application environment:

- Java EE components and applications deployed in Oracle WebLogic Server use the audit loader functionality provided through the application server.
- System components and non-Java applications use the audit loader functionality provided through Oracle Process Manager and Notification Server (OPMN).
- Java SE applications, which run outside an application server container, use a stand-alone audit loader.

This section explains how to set up and execute the stand-alone audit loader:

- [Configuring the Environment](#)
- [Running the Stand-Alone Audit Loader](#)

13.2.6.1 Configuring the Environment

Before you can run the stand-alone audit loader, you must a) configure certain properties, and b) ensure that the password for the database schema user exists in the secret store.

13.2.6.1.1 Property Configuration

You must configure the following properties:

- `ORACLE_HOME` environment variable
- `COMMON_COMPONENTS_HOME` environment variable
- `ORACLE_INSTANCE` environment variable
- `auditloader.jdbcString` system property
- `auditloader.username` system property

13.2.6.1.2 Password Storage for the Database Schema User

The password for the database schema user is kept in the secret store. Storing the password is a one-time operation for which you use the `StandAloneAuditLoader` command with the `-Dstore.password=true` option.

Issue the `StandAloneAuditLoader` command to store the password as follows:

```
$ORACLE_HOME/jdk/bin/java
-classpath $COMMON_COMPONENTS_HOME/modules/oracle.osdt_11.1.1/osdt_cert.jar:
$COMMON_COMPONENTS_HOME/modules/oracle.osdt_11.1.1/osdt_core.jar:
$COMMON_COMPONENTS_HOME/modules/oracle.jdbc_11.1.1/ojdbc6dms.jar:
$COMMON_COMPONENTS_HOME/modules/oracle.iau_11.1.1/fmw_audit.jar:
$COMMON_COMPONENTS_HOME/modules/oracle.pki_11.1.1/oraclepki.jar
-Doracle.instance=$ORACLE_INSTANCE
-Dauditloader.jdbcString=jdbc:oracle:thin:@host:port:sid
-Dauditloader.username=username
-Dstore.password=true
oracle.security.audit.ajl.loader.StandaloneAuditLoader
```

13.2.6.2 Running the Stand-Alone Audit Loader

Issue the `StandAloneAuditLoader` command to load audit records as follows:

```
$ORACLE_HOME/jdk/bin/java
-classpath $COMMON_COMPONENTS_HOME/modules/oracle.osdt_11.1.1/osdt_cert.jar:
$COMMON_COMPONENTS_HOME/modules/oracle.osdt_11.1.1/osdt_core.jar:
$COMMON_COMPONENTS_HOME/modules/oracle.jdbc_11.1.1/ojdbc6dms.jar:
$COMMON_COMPONENTS_HOME/modules/oracle.iau_11.1.1/fmw_audit.jar:
$COMMON_COMPONENTS_HOME/modules/oracle.pki_11.1.1/oraclepki.jar
-Doracle.home=$ORACLE_HOME
-Doracle.instance=$ORACLE_INSTANCE
-Dauditloader.jdbcString=jdbc:oracle:thin:host:port:sid
-Dauditloader.username=username
oracle.security.audit.ajl.loader.StandaloneAuditLoader
```

You can schedule this command through a batch or cron job so that audit records are periodically uploaded to the audit data store.

13.3 Managing Audit Policies

What is an Audit Policy?

An audit policy is a declaration of the type of events to be captured by the audit framework for a particular component. For Java components, the audit policy is defined at the domain level. For system components, the audit policy is managed at the component instance level.

For example, an audit policy could specify that all authentication failures should be audited for an Oracle Internet Directory instance.

How Policies are Configured

Oracle Fusion Middleware Audit Framework lets you configure audit policies and provides highly granular controls over the types of events and data being audited. Policies can be configured through the Enterprise Manager UI tool and through the WLST command-line interface.

Policy changes do not require server or instance restart.

The remainder of this section explains how to view, and update audit policy:

- [Manage Audit Policies for Java Components with Fusion Middleware Control](#)
- [Manage Audit Policies for System Components with Fusion Middleware Control](#)
- [Manage Audit Policies with WLST](#)
- [Manage Audit Policies Manually](#)

See Also:

- [Section 12.3.2, "Key Technical Concepts"](#) for additional background.
- Appendix D, Oracle Fusion Middleware Audit Framework Reference for a list of Java components and system components.
- *Oracle Fusion Middleware Third-Party Application Server Guide* for details about executing audit commands on third-party application servers.

13.3.1 Manage Audit Policies for Java Components with Fusion Middleware Control

The domain Audit Policy Settings page manages audit events for all Java components such as Oracle Identity Federation, and system libraries like Oracle Platform Security Services.

Note:

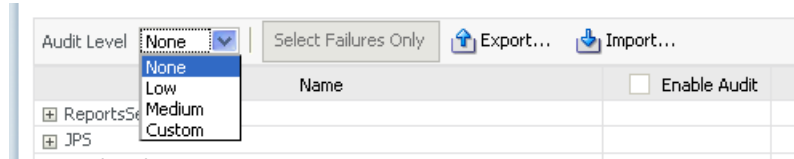
- Audit policy for system components is managed in the component home pages. See [Section 13.3.2, "Manage Audit Policies for System Components with Fusion Middleware Control"](#)
 - See the note at the beginning of [Section 13.3, "Managing Audit Policies"](#) titled "Policy Changes Require Server or Instance Restart".
-
-

See Also : [Section C.1.1, "What Components Can be Audited?"](#) for the list of auditable components.

Each component and its events are organized in a tree structure under the Name column. The tree can be expanded to reveal the details of the events available.

Use these steps to view and update the currently configured audit policies:

1. Log in to Fusion Middleware Control.
2. Using the topology panel to the left, navigate to the domain of interest under "WebLogic Domain".
3. From the domain menu, navigate to *Domain > Security > Audit Policy Settings*. The Audit Policy Settings page appears
4. A drop-down list of pre-configured audit levels can be selected. Two pre-defined levels (Low, Medium) will automatically pick up a subset of the audit events for all the components. In most cases, the pre-defined levels are sufficient.



Note: The table of events under the drop-down box cannot be edited for the pre-defined levels. It can only be edited in custom level.

- None - No events are selected for audit.
- Low - A small set of events is selected, typically those having the smallest impact on component performance.
- Medium - This is a superset of the "Low" set of events. These events may have a higher impact on component performance.
- Custom - This level enables you to fine-tune the policy, and is described in Step 5 below.

The table shows the applications running in the domain.

Name	Enable Audit	Filter
Directory Integration Platform Server	<input type="checkbox"/>	
Oracle Platform Security Services	<input type="checkbox"/>	
Oracle Identity Federation	<input type="checkbox"/>	
Oracle Web Services Manager - Agent	<input type="checkbox"/>	
Oracle Web Services Manager - Policy Manager	<input type="checkbox"/>	
ReportsServer	<input type="checkbox"/>	
Oracle Web Services Manager - Policy Attachment	<input type="checkbox"/>	
Oracle Web Services	<input type="checkbox"/>	

The table consists of these columns:

- Name - shows components and applications in the domain.
- Enable Audit - shows whether the corresponding event type is being audited. This column is greyed out unless the Custom audit policy is in force.
- Filter - shows any filters in effect for the event type.

5. To customize the audit policy, use the "Custom" option from the drop-down. This allows you to select all the events or hand-pick the appropriate subset as desired by checking the relevant boxes under the "Enable Audit" column. When you choose the Custom level, an optional filter is available for success and failure outcomes of each individual event to further control how they are audited, as explained in Step 6 below.
6. Filters are rule-based expressions that you can define to qualify or filter events for audit. The expressions are based on attributes of the event. For example, a Login type event could specify an initiator as a user filter in which case the event would generate an audit record whenever the specified user logged in.

A pencil icon indicates that a filter is available for the corresponding event.

Name	<input checked="" type="checkbox"/> Enable Audit	Filter	Edit Filter
[-] ReportsServer	<input checked="" type="checkbox"/>		
[-] JPS	<input checked="" type="checkbox"/>		
[-] Authorization	<input checked="" type="checkbox"/>		
[-] CheckPermission	<input checked="" type="checkbox"/>		
Success	<input checked="" type="checkbox"/>		
Failure	<input checked="" type="checkbox"/>		
[-] CheckSubject	<input checked="" type="checkbox"/>		
[-] Credential Management	<input checked="" type="checkbox"/>		

Click on the icon to bring up the Edit Filter dialog.

Edit Filter: CheckPermission (Success)

Define an audit filter by specifying one or more conditions. Only events that meet the conditions will be audited.

() AND OR Condition

Host Id
 Host Network Address
 Initiator
 Client IP Address
 Target User
 Resource
 Roles

Note: Each filter attribute has a formal name and a display name. You may see either name in the filter edit dialog. Display names are shown in the drop-down, while names are shown in the edit dialog. For example, if you select 'Client Address IP' in the drop-down box, it is renamed to 'RemoteIP' after you add it to the filter expression.

7. Click the "Select Failures Only" button to select only failed events in the policy - for example, a failed authentication. The Enable Audit box is now checked for failed events.
8. Import/Export - These buttons enable you to save and re-use a policy configuration. At any time while editing the policy, click Export to save the current settings to a file, and Import to load the settings from a saved file.
9. Optionally, under "Users to Always Audit", you can specify a comma-separated list of users to force the audit framework to audit events initiated by these users; auditing occurs regardless of the audit level or filters that have been specified.

Users to Always Audit

Enter a comma-delimited list of user accounts that will always be audited.

Users

Notes:

- Be aware that if you use this feature to audit key users such as system administrators, this will generate audit traffic anytime that user touches any of the auditable events for any component. For example, a component's audit policy may be set to None, but if these users perform some activity in the component instance, it is still audited.
 - No validation is performed for user names you enter in this field.
-

10. If you made any policy changes, click **Apply** to save the changes. For Java components, you must restart the managed Oracle WebLogic Server (on which the affected Java component is running) for the changes to be effective.

Click **Revert** to discard any policy changes and revert to the existing policy.

About Component Events

Each component and application in the domain defines its own set of auditable events. Thus, when you expand the Names column of the table, each component displays a list of events that applies to instances of that component.

13.3.2 Manage Audit Policies for System Components with Fusion Middleware Control

This section describes how to view and update audit policies for system components that are managed through OPMN.

Notes:

- Audit policy for Java components is managed in the domain context. See [Section 13.3.1, "Manage Audit Policies for Java Components with Fusion Middleware Control"](#)
 - See the note at the beginning of [Section 13.3, "Managing Audit Policies"](#) titled "Policy Changes Require Server or Instance Restart". Oracle Internet Directory instances do not require a restart.
-

Audit policy for system components is managed in their home pages. The domain Audit Policy Settings page manages audit events for Java components running in the domain.

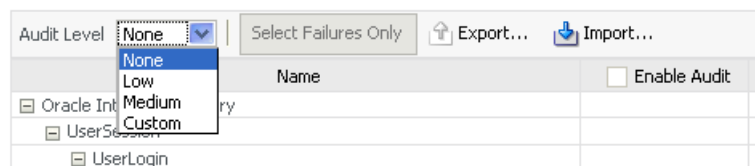
See Also : [Section C.1.1, "What Components Can be Audited?"](#) for the list of auditable components.

The events are organized in a tree structure under the Name column. The tree can be expanded to reveal the details of the events available.

Use these steps to view and update audit policies for OPMN-managed components:

1. Log in to Fusion Middleware Control.
2. Using the topology panel to the left, navigate to the system component of interest such as Oracle Internet Directory.
3. From the component menu, navigate to *Security*, then *Audit Policy*. The Audit Policy Settings page appears

4. A drop-down list of pre-configured audit levels can be selected. Two pre-defined levels (Low, Medium) will automatically pick up a subset of the audit events for all the components.



Note: The table of events under the drop-down box cannot be edited for the pre-defined levels. It can only be edited in custom level.

- None - No events are selected for audit.
- Low - A small set of events is selected, typically those having the smallest impact on component performance.
- Medium - This is a superset of the "Low" set of events. These events may have a higher impact on component performance.
- Custom - This level enables you to fine-tune the policy, and is described in Step 5 below.





The table shows the events you can audit for the component instance. This example is for Oracle Internet Directory:

Name	<input type="checkbox"/> Enable Audit	Filter
[-] Oracle Internet Directory		
[-] UserSession		
[-] UserLogin		
Success		
Failure		
[-] UserLogout		
Success		
Failure		
[+] Authorization		
[+] DataAccess		

The table consists of these columns:

- Name - shows the component events grouped by type, such as Authorization events.
 - Enable Audit - shows whether the corresponding event type is being audited. This column is greyed out unless the Custom audit policy is in force.
 - Filter - shows any filters in effect for the event type.
5. To customize the audit policy, use the "Custom" option from the drop-down. This allows you to select all the events or hand-pick the appropriate subset as desired by checking the relevant boxes under the "Enable Audit" column. When you choose the Custom level, an optional filter available for success and failure outcomes of each individual event to further control how they are audited, as explained in Step 6 below.
 6. Filters are rule-based expressions that you can define to qualify or filter events for audit. The expressions are based on attributes of the event. For example, a Login type event could specify an initiator as a user filter in which case the event would generate an audit record whenever the specified user logged in.

A pencil icon indicates that a filter is available for the corresponding event.

Name	<input type="checkbox"/> Enable Audit	Filter	Edit Filter
Oracle Internet Directory	<input type="checkbox"/>		
UserSession	<input type="checkbox"/>		
UserLogin	<input type="checkbox"/>		
Success	<input type="checkbox"/>		
Failure	<input type="checkbox"/>		
UserLogout	<input type="checkbox"/>		
Success	<input type="checkbox"/>		
Failure	<input type="checkbox"/>		

Click on the icon to bring up the Edit Filter dialog.

Edit Filter: UserLogin (Success)

Define an audit filter by specifying one or more conditions. Only events that meet the conditions will be audited.

() AND OR Condition

HostId
HostNwaddr
Initiator
RemoteIP
Roles

Help

Note: Each filter attribute has a formal name and a display name. You may see either name in the filter edit dialog. Display names are shown in the drop-down, while names are shown in the edit dialog. For example, if you select 'Client Address IP' in the drop-down box, it is renamed to 'RemoteIP' after you add it to the filter expression.

- Click "Select Failures Only" to select only failed events in the policy - for example, a failed authentication. The Enable Audit box is now checked for failed events.
- Import/Export - These buttons enable you to save and re-use a policy configuration. At any time while editing the policy, click Export to save the current settings to a file, and Import to load the settings from a saved file.
- Optionally, under "Users to Always Audit", a comma-separated list of users can be specified to force the audit framework to audit events initiated by these users; auditing occurs regardless of the audit level or filters that have been specified.

Users to Always Audit

Enter a comma-delimited list of user accounts that will always be audited.

Users

Notes:

- Be aware that if you use this feature to audit key users such as system administrators, this will generate audit traffic anytime that user touches any of the auditable events for any component. For example, a component's audit policy may be set to None, but if these users perform some activity in the component instance, it is still audited.
- No validation is performed for user names you enter in this field.

- If you made any policy changes, click **Apply** to save the changes.

Click **Revert** to discard any policy changes and revert to the existing policy.

13.3.3 Manage Audit Policies with WLST

This section explains how to view and update audit policies using the Oracle WebLogic Scripting Tool (WLST) command-line tool:

- [View Audit Policies with WLST](#)
- [Update Audit Policies with WLST](#)
- [Example 1: Configuring an Audit Policy for Users with WLST](#)
- [Example 2: Configuring an Audit Policy for Events with WLST](#)
- [Custom Configuration is Retained when the Audit Level Changes](#)

Note: When running audit WLST commands, you must invoke the WLST script from the Oracle Common home. See "Using Custom WLST Commands" in the *Oracle Fusion Middleware Administrator's Guide* for more information.

13.3.3.1 View Audit Policies with WLST

Take these steps to view audit policies with WLST:

Note: This discussion assumes that you are invoking WLST interactively. For details about WLST and the different options for invoking the tool, see "Getting Started Using the Oracle WebLogic Scripting Tool (WLST)" in the *Oracle Fusion Middleware Administrator's Guide*.

- Connect to the WebLogic Server using the following commands:

```
java weblogic.WLST
connect('servername', 'password', 'localhost:portnum')
```

- Use the `getAuditPolicy` command to view the audit policy configuration. For example:

```
wls:/mydomain/serverConfig> getAuditPolicy()
```

- For system components:

- obtain the MBean name using the `getNonJava EEAuditMBeanName` command. See [Section C.4.1, "getNonJava EEAuditMBeanName"](#) for details.
- Use the `getAuditPolicy` command and include the MBean name to view the audit policy configuration. For example:

```
wls:/mydomain/serverConfig> getAuditPolicy
(on="oracle.security.audit.test:type=CSAAuditMBean,name=CSAAuditProxyMBean")
```

13.3.3.2 Update Audit Policies with WLST

Take these steps to update audit policies with the Oracle WebLogic Scripting Tool (WLST) command-line tool:

Note: This discussion assumes that you are invoking WLST interactively. For details about WLST and the different options for invoking the tool, see "Getting Started Using the Oracle WebLogic Scripting Tool (WLST)" in the *Oracle Fusion Middleware Administrator's Guide*.

- Connect to the WebLogic Server using the following commands:

```
java weblogic.WLST
connect('servername', 'password', 'localhost:portnum')
```

- Navigate the bean hierarchy to access the domain of interest. For example, if the domain is called mydomain:

```
wls:/mydomain/serverConfig>
```

- Use the `setAuditPolicy` command to update the audit policy configuration.
- For components that manage their policy locally, use the `setAuditPolicy` command and include an MBean name to update the audit policy configuration.
- Explicitly call `save` after issuing a `setAuditPolicy`, or `importAuditConfig`, command.

If you do not invoke `save`, the new settings will not take effect.

For an example of this call, see *Managing Auditing by Using WLST* in the *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*, which demonstrates this call for Oracle Internet Directory auditing.

See Also: The WLST command reference for details about WLST commands for audit.

13.3.3.3 Example 1: Configuring an Audit Policy for Users with WLST

In this scenario, the domain's current policy audits a user named user1. We would like to add two names, user2 and user3, to the list of users who are always audited, and remove user1 from the list.

The following invocation of `setAuditPolicy` performs this task:

```
setAuditPolicy
(filterPreset="None", addSpecialUsers="user2,user3", removeSpecialUsers="user1")
```

See Also: [Section C.4.3, "setAuditPolicy"](#)

13.3.3.4 Example 2: Configuring an Audit Policy for Events with WLST

In this scenario, the domain's current policy audits user logout events. We would like to remove the logout events from the policy and instead, audit login events.

The following invocation of `setAuditPolicy` performs this task:

Note: This example uses the component type OHS for Oracle HTTP Server. Substitute the relevant component type when using the command.

```
setAuditPolicy
```

```
(filterPreset="Custom",addCustomEvents="OHS:UserLogin",  
removeCustomEvents="OHS:UserLogout")
```

Notice that we had to set the `Custom` filter preset to add and remove events.

13.3.3.5 Custom Configuration is Retained when the Audit Level Changes

When auditing is configured at the custom audit level, and you subsequently use WLST to switch to a different (non-custom) audit level, the custom audit settings are retained unless you explicitly remove those custom settings.

Note: This behavior only occurs when using WLST; if you use Fusion Middleware Control to manage audit configuration, the custom audit settings are cleared when you switch from the custom audit level to a different audit level.

An example illustrates this behavior:

1. Custom audit level is set for a component's policy. An audit filter is specified as part of the configuration.
2. At run-time, audit data is collected according to the specified filter.
3. The component's audit policy is now changed from custom audit level to, say, the low audit level using the WLST `setauditpolicy` command. However, the filter that was set up as part of the custom audit level persists in the audit configuration.
4. Audit data is now collected based on the low audit level, not the custom level.
5. The component's audit policy is changed back to custom level. An additional filter is added; this filter is appended to the originally configured filter. Unless the original filter is explicitly deleted, it remains part of the configuration.
6. At run-time, audit data is collected based on all prevailing filters at the custom level.

13.3.4 Manage Audit Policies Manually

This section explains how to configure auditing policies and other features by manually updating:

- the platform configuration file `jps-config.xml` for Java components
- component-specific files for system components

This section contains these topics:

- [Location of Configuration Files for Java Components](#)
- [Audit Service Configuration Properties in jps-config.xml for Java Components](#)
- [Switching from Database to File for Java Components](#)
- [Manually Configuring Audit for System Components](#)

13.3.4.1 Location of Configuration Files for Java Components

The `jps-config.xml` domain configuration file can be found at this location:

```
$DOMAIN_HOME/config/fmwconfig/jps-config.xml
```

13.3.4.2 Audit Service Configuration Properties in jps-config.xml for Java Components

The Audit Service Configuration in `jps-config.xml` consists of the properties shown in [Table F-9](#). Taken together, the set of properties and their values are known as the audit policy.

Example jps-config.xml file

Here is a sample file illustrating an audit policy:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<jpsConfig xmlns="http://xmlns.oracle.com/oracleas/schema/11/jps-config-11_1.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.oracle.com/oracleas/schema/11/jps-config-11_
1.xsd" schema-major-version="11" schema-minor-version="1">

    <serviceProviders>
        <serviceProvider name="audit.provider" type="AUDIT"
class="oracle.security.jps.internal.audit.AuditProvider">
            </serviceProvider>
        </serviceProviders>

    <serviceInstances>
        <serviceInstance name="audit" provider="audit.provider">
            <property name="audit.filterPreset" value="Low"/>
            <property name="audit.specialUsers" value="admin, fmwadmin" />
            <property name="audit.customEvents" value="JPS:CheckAuthorization,
                CreateCredential; OIF:UserLogin"/>
            <property name="audit.loader.jndi" value="jdbc/AuditDB"/>
            <property name="audit.loader.interval" value="15" />
            <property name="audit.maxDirSize" value="102400" />
            <property name="audit.maxFileSize" value="10240" />
            <property name="audit.loader.repositoryType" value="Db" />
        </serviceInstance>
    </serviceInstances>
    <jpsContexts default="default">
        <jpsContext name="default">
            <serviceInstanceRef ref="audit"/>
        </jpsContext>
    </jpsContexts>
</jpsConfig>
```

13.3.4.3 Switching from Database to File for Java Components

In rare instances, you may wish to revert from using a (database) data store to using a file for audit records. This requires manual configuration of the property `audit.loader.repositoryType` described in [Table F-9](#).

To switch from database to file, set the `audit.loader.repositoryType` to `File`.

When you switch from database to file, events that were collected in the database are not transferred back to the file system. If this switch is temporary, the audit events collected in the file are automatically pushed to the database when you switch to a database store again.

13.3.4.4 Manually Configuring Audit for System Components

System components do not use the `jps-config.xml` file to store the audit configuration. Instead:

- Oracle HTTP Server uses the `auditconfig.xml` file which is located in:
`ORACLE_INSTANCE/instance_name/config/OHS/<ohs_name>/auditconfig.xml`
- Oracle Web Cache uses the `auditconfig.xml` file which is located in:
`ORACLE_INSTANCE/instance_name/config/WebCache/<webcache_name>/auditconfig.xml`
- Oracle Reports uses the `jps-config-jse.xml` file which is located in:
`$DOMAIN_HOME/config/fmwconfig/jps-config-jse.xml`
- Oracle Virtual Directory uses `jps-config.-jse.xml` file which is located in:
`ORACLE_INSTANCE/instance_name/config/JPS/jps-config-jse.xml`
- Oracle Internet Directory's audit configuration is stored in the database.

Format of the `auditconfig.xml` File

Here is the format of the `auditconfig.xml` file:

```
<AuditConfig xmlns="http://xmlns.oracle.com/ias/audit/audit.xsd">
  <Filters>
    <!-- FilterPreset can be None,Low,Medium,All or Custom. Default value: None -->
    -->
    <FilterPreset>Low</FilterPreset>

    <!-- Comma separated list of special users for whom auditing is always turned
    on. Default value: no users -->
    <SpecialUsers>u1,u2</SpecialUsers>

    <!-- In case of custom, a comma separate list of events that are to be enabled
    for auditing. Default value: no events -->
    <CustomEvents>e1,e1</CustomEvents>

  </Filters>
  <LogsDir>

    <!-- Maximum dir size of the log directory (busstop). 0 implies unlimited
    size. Default value: 0 -->
    <MaxDirSize>0</MaxDirSize>

    <!-- Maximum file size of each audit.log file. Default value: 100MB -->
    <MaxFileSize>104857600</MaxFileSize>

  </LogsDir>
</AuditConfig>
```

13.4 Audit Logs

Fusion Middleware Audit Framework provides a set of log files to help with audit administration. You can use these logs to trace errors and for diagnostic purposes when the audit framework is not functioning properly.

This section contains the following topics:

- [Location of Audit Logs](#)
- [Audit Log Timestamps](#)

13.4.1 Location of Audit Logs

For a listing of all audit log locations, how to configure the loggers, and how to use the logs to diagnose issues, see [Section L.1.1.6, "Audit Loggers"](#).

13.4.2 Audit Log Timestamps

Time stamps in the audit logs are recorded in Coordinated Universal Time. This may differ from the machine time depending on the machine's time zone setting.

13.5 Advanced Management of Database Store

The audit schema is created through the Repository Creation Utility (RCU). This section explains the organization of the audit schema and contains the following topics related to maintaining the schema:

See Also: For more information on RCU, see *Oracle Fusion Middleware Repository Creation Utility User's Guide*.

13.5.1 Schema Overview

The Oracle Fusion Middleware Audit Framework schema consists of the following:

- A base table: IAU_BASE
- A translation table: IAU_DISP_NAMES_TL
- A set of component-specific tables of audit data, for example OVDCOMPONENT, OIDCOMPONENT, JPS and so on

When generated, audit records are stored in a file; if an audit database store is configured, the audit loader stores each audit record in one row of the base table and one row of a component table:

- General information (such as Time, EventType, and EventStatus) is written into the base table
- component-specific information (such as CodeSource) is written into the component table

Note: The attribute `ComponentType` in the bus-stop file determines which component table stores the record.

The audit loader assigns unique sequential numbers to all records during storage.

Here is a sample bus-stop file for Oracle Platform Security Services. By default, this file is maintained in the directory

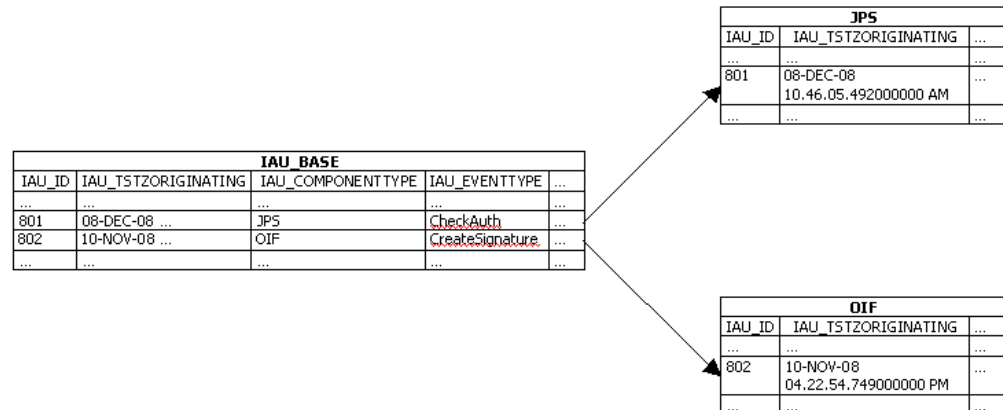
```
WebLogic Domain Home/servers/server_
name/diagnostics/auditlogs/JPS/audit.log
```

```
#Fields:Date Time Initiator EventType EventStatus MessageText HomeInstance ECID
RID ContextFields SessionId TargetComponentType ApplicationName EventCategory
ThreadId InitiatorDN TargetDN FailureCode RemoteIP Target Resource Roles
CodeSource InitiatorGUID Principals PermissionAction PermissionClass mapName key
#Remark Values:ComponentType="JPS"
2008-12-08 10:46:05.492 - "CheckAuthorization" true "Oracle Platform Security
Authorization Check Permission SUCCEEDED." - - - - - "Authorization" "48" - -
"true" - - "(oracle.security.jps.service.policystore.PolicyStoreAccessPermission
context=APPLICATION,name=SimpleServlet getApplicationPolicy)" -
```

```
"file:/oracle/work/middleware/oracle_common/modules/oracle.jps_
11.1.1/jps-internal.jar" - "[]" - - - -
```

Figure [Figure 13–1](#) shows the data in the base table and how it relates to the component-specific tables.

Figure 13–1 Audit Schema



The average record size in the base table **IAU_BASE** is approximately 0.3 KB. When you plan for tablespace sizing:

- use this number as a guideline for the average record size
- monitor how audit database size is growing based on the audit policy selected and the level of activity
- take into account the period of time for which the audit data is being stored.

13.5.2 Table Attributes

The attributes of the base table and the component-specific tables respectively are derived from these files:

```
$ORACLE_HOME/modules/oracle.iau_11.1.1/components/generic/generic_events.xml
```

for the base table, and

```
$ORACLE_HOME/modules/oracle.iau_11.1.1/components/componentName/component_
events.xml
```

for each component table.

[Table 13–1](#) lists a few important attributes defined in the base table **IAU_BASE**. The first four attributes are common in that table and all component tables. The primary key is defined as **IAU_ID** + **IAU_TSTZORIGINATING**.

See Also: [Section C.3, "The Audit Schema"](#)

Table 13–1 Attributes of Base Table IAU_BASE

Attribute	Description
IAU_ID	A unique sequential number for every audit record
IAU_TstzOriginating	Date and time when the audit event was generated (data type TIMESTAMP)

Table 13–1 (Cont.) Attributes of Base Table IAU_BASE

Attribute	Description
IAU_EventType	The type (name) of the audit event
IAU_EventCategory	The category of the audit event
IAU_EventStatus	The outcome of the audit event - success or failure
IAU_MessageText	Description of the audit event
IAU_Initiator	UID of the user who was doing the operation

Note: A `SEQUENCE`, an Oracle database object, is created to coordinate the assignment of sequential numbers (`IAU_ID`) for audit records.

You can use the `listAuditEvents` WLST command to get a list of all attribute names for individual component tables.

See Also:

- [Section C.4, "WLST Commands for Auditing"](#).
- [Section C.3, "The Audit Schema"](#)

13.5.3 Indexing Scheme

For efficient queries, an index is created by default on the Timestamp (`IAU_TSTZORIGINATING`) in the base table and on each of the component-specific tables.

The default index in `IAU_BASE` is named `EVENT_TIME_INDEX`, and in the component tables it is named `tableName_INDEX` (such as `OVDCOMPONENT_INDEX`, `OIDCOMPONENT_INDEX`, `JPS_INDEX` and so on).

13.5.4 Backup and Recovery

Compliance regulations require that audit data be stored for long periods. A backup and recovery plan is needed to protect the data.

A good backup plan takes account of these basic guidelines:

- Growth rate of Audit Events

The number of audit events generated depends on your audit policy. The number of audit events generated daily determines, in turn, how often you want to perform backups to minimize the loss of your audit data.
- Compliance regulations

Consult you organization's compliance regulations to determine the frequency of backups and number of years for which audit data storage is mandatory.
- Online or Offline Data Management

Consult you organization's compliance regulations to determine the frequency of backups and the portion of audit data that needs to be easily accessible.

Oracle Database uses Oracle Recovery Manager (RMAN) for backup and recovery. For details, see:

http://www.oracle.com/technology/deploy/availability/htdocs/BR_Overview.htm

http://www.oracle.com/technology/deploy/availability/htdocs/rman_overview.htm

Note: The translation table, `IAU_DISP_NAMES_TL`, needs to be backed up only once, since it should not change over time.

13.5.5 Importing and Exporting Data

You can import and export the audit schema to migrate data if you started with multiple audit databases and wish to combine them into a single audit data store, or if you wish to change the database to scale up.

Oracle Database sites can utilize the utilities of Oracle Data Pump to import and export data. For details, refer to:

http://www.oracle.com/technology/products/database/utilities/htdocs/data_pump_overview.html

13.5.6 Partitioning

Not all database systems support partitioning, all the tables in the audit schema are unpartitioned by default.

Since audit data is cumulative and older data is never removed, if you store a high volume of audit data you should consider partitioning the audit schema, as it will allow for easier archiving.

Benefits of partitioning include:

- **Improved Performance:** If a table is range-partitioned by Timestamps, for example, queries by Timestamps can be processed on the partitions within that time-frame only.
- **Better Manageability:** Partitions can be created on separate tablespaces (thus different disks). This enables you to move older data to slower and larger disks, while keeping newer data in faster and smaller disks.

In addition, partitioning makes archival much easier. For example, you can compress a single partition rather than having to partition the entire table.

- **Increased Availability:** If a single partition is unavailable, for example, and you know that your query can eliminate this partition from consideration, the query can be successfully processed without needing to wait for the unavailable partition.

13.5.6.1 Partition Tables

In this example, `IAU_BASE` is used as an example to demonstrate how to convert the unpartitioned tables in the audit schema into partitioned tables.

It is recommended that partitioning is done before using this schema for an audit data store to minimize the application down time.

Note: Two sample SQL scripts are shipped with the product:

- \$RCU_
HOME/rcu/integration/iau/scripts/convertPartitionedTables.sql (linux) or %RCU_
HOME\rcu\integration\iau\scripts\convertPartitionedTables.sql (Windows) converts the base and component tables in audit schema into partitioned tables
 - \$RCU_
HOME/rcu/integration/iau/scripts/createPartitionsByQuarter.sql (linux) or %RCU_
HOME\rcu\integration\iau\scripts\createPartitionsByQuarter.sql (Windows) creates partitions by quarter for the base and component tables in the audit schema
-
-

The partitioning steps are as follows:

Note: It is recommended that you deactivate the audit loader prior to partitioning. See [Section 13.2.4.1, "Deconfigure the Audit Data Store"](#) for details.

1. Rename the existing unpartitioned table. For example:

```
RENAME IAU_BASE TO IAU_BASE_NONPART;
```

2. Create a new partitioned table that follows the table structure of the unpartitioned table. This example uses the range-partitioning (by Timestamp) scheme:

```
CREATE TABLE IAU_BASE  
PARTITION BY RANGE (IAU_TSTZORIGINATING)  
(  
    PARTITION IAU_BASE_DEFAULT VALUES LESS THAN (MAXVALUE)  
)  
AS SELECT * FROM IAU_BASE_NONPART;
```

3. Enable row movement to allow data to automatically move from partition to partition when new partitions are created. For example:

```
ALTER TABLE IAU_BASE  
ENABLE ROW MOVEMENT;
```

4. Create a local prefix index for the partitioned table. For example:

```
ALTER INDEX EVENT_TIME_INDEX  
RENAME TO EVENT_TIME_INDEX_NONPART;  
  
CREATE INDEX EVENT_TIME_INDEX  
ON IAU_BASE(IAU_TSTZORIGINATING) LOCAL;
```

5. Partitions can now be created. In this example partitions are created by calendar quarter:

```
ALTER TABLE IAU_BASE  
SPLIT PARTITION IAU_BASE_DEFAULT AT (TO_DATE('01/04/2008', 'DD/MM/YYYY')) INTO  
(PARTITION IAU_BASE_Q1_2008, PARTITION IAU_BASE_DEFAULT)  
UPDATE INDEXES;
```

```
ALTER TABLE IAU_BASE
SPLIT PARTITION IAU_BASE_DEFAULT AT (TO_DATE('01/07/2008', 'DD/MM/YYYY')) INTO
(PARTITION IAU_BASE_Q2_2008, PARTITION IAU_BASE_DEFAULT)
UPDATE INDEXES;
```

```
ALTER TABLE IAU_BASE
SPLIT PARTITION IAU_BASE_DEFAULT AT (TO_DATE('01/10/2008', 'DD/MM/YYYY')) INTO
(PARTITION IAU_BASE_Q3_2008, PARTITION IAU_BASE_DEFAULT)
UPDATE INDEXES;
```

```
ALTER TABLE IAU_BASE
SPLIT PARTITION IAU_BASE_DEFAULT AT (TO_DATE('01/01/2009', 'DD/MM/YYYY')) INTO
(PARTITION IAU_BASE_Q4_2008, PARTITION IAU_BASE_DEFAULT)
UPDATE INDEXES;
```

Note: New partitions should be created periodically for new quarters.

13.5.6.2 Backup and Recovery of Partitioned Tables

Backup and recovery were discussed in [Section 13.5.4, "Backup and Recovery"](#). Note that read-only tablespaces can be excluded from whole database backup, so long as a backup copy was created. Thus, you can avoid unnecessarily repeating backups for the partitions of archived data residing on those tablespaces, improving performance.

13.5.6.3 Import, Export, and Data Purge

Import and export were discussed in [Section 13.5.5, "Importing and Exporting Data"](#). Keep in mind that with a range-partitioned table it is much more efficient to drop a partition when you want to remove old data, rather than deleting the rows individually.

```
ALTER TABLE IAU_BASE DROP PARTITION IAU_BASE_Q4_2008;
```

It is also easy to load a partition of new data without having to modify the entire table. However, you have to remove the default partition of "values less than (MAXVALUE)" first, and add it back once finished, using a command like the following:

```
ALTER TABLE IAU_BASE ADD PARTITION IAU_BASE_Q4_2008 VALUES LESS THAN
('01-JAN-2009');
```

Once partitions are created, you can purge/backup a particular partition. Refer to your database documentation for details.

In the database mode, the audit loader automatically manages bus-stop files.

13.5.6.4 Tiered Archival

Partitioning enables individual partitions (or groups of partitions) to be stored on different storage tiers. You can create tablespaces in high-performance or low-cost disks, and create partitions in different tablespaces based on the value of the data or other criteria. It is also easy to move data in partitions between the tablespaces (storage tiers).

Here is an example:

```
ALTER TABLE IAU_BASE MOVE PARTITION IAU_BASE_Q1_2008
TABLESPACE AUDITARCHIVE UPDATE INDEXES;
```

Note : Partitions can be moved only in Range, List, System, and Hash partitioning schemes.

The Oracle Information Lifecycle Management (ILM) Assistant is a free tool that shows you how to partition tables and advise you when it is the time to move partitions. For details, refer to:

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

13.6 Metadata Schema Overview

Awaiting content from dev (Sam).

Using Audit Analysis and Reporting

This chapter describes how to configure audit reporting and how to view audit reports. It contains these topics:

- [Setting up Oracle Business Intelligence Publisher for Audit Reports](#)
- [Organization of Audit Reports](#)
- [View Audit Reports](#)
- [Example of Oracle Business Intelligence Publisher Reports](#)
- [Audit Report Details](#)
- [Customizing Audit Reports](#)

14.1 Setting up Oracle Business Intelligence Publisher for Audit Reports

When your audit data resides in a database, you can run pre-defined Oracle Business Intelligence Publisher reports and create your own reports on the data. This section contains these topics about configuring your environment for audit reports:

- [About Oracle Business Intelligence Publisher](#)
- [Install Oracle Business Intelligence Publisher](#)
- [Set Up Oracle Reports in Oracle Business Intelligence Publisher](#)
- [Set Up Audit Report Templates](#)
- [Set Up Audit Report Filters](#)
- [Configure Scheduler in Oracle Business Intelligence Publisher](#)

See Also: Oracle Business Intelligence Publisher Enterprise documentation at:

http://www.oracle.com/technology/documentation/bi_pub.html

14.1.1 About Oracle Business Intelligence Publisher

Reports help auditors determine whether there are any violations with respect to various industry regulations such as HIPPA, SOX, and other regulatory compliance demands. Oracle Fusion Middleware Audit Framework is integrated with Oracle Business Intelligence Publisher for out-of-the box reports.

Pre-defined reports are available as part of the Oracle Fusion Middleware Audit Framework. These reports are integrated with Oracle Business Intelligence Publisher to work in conjunction with the audit data in the audit store.

Oracle Fusion Middleware Audit Framework ships with over twenty pre-built reports in 11g Release 1 (11.1.1). For convenience, the reports are grouped in Oracle Business Intelligence Publisher according to functional areas and by component.

The functional areas consists of the following:

- Error and Exception reports like authentication and authorization failures
- User Activities including transaction history and authorization history
- Operational reports including created, deleted, and locked-out users
- Audit Service Events

The component-specific reports, as the name implies, are grouped based on the components themselves, for example, Oracle HTTP Server reports and Oracle Identity Federation reports.

Other features of Oracle Business Intelligence Publisher include:

- Flexible Report Displays

You can view reports online, change report parameters, change output types (pdf, html, rtf, excel and others), modify the appearance of reports, export to the desired format, and send to an E-mail address, fax or other destination.

- Report Filters

You can filter audit records to be included in the report using a range of options including the ability to modify the SQL used to extract records from the audit repository.

- Scheduling Reports

You can schedule reports to be run based on a range of criteria such as filters, templates, formats, locale, viewing restrictions and so on.

See Also: For more information about scheduling features, see the Oracle Business Intelligence Publisher Enterprise documentation at:

http://www.oracle.com/technology/documentation/bi_pub.html

- Custom Reporting

You can design your own reports and specify the data model, layout, parameters, bursting (for example, you can enable delivery based on delivery preference).

See Also:

- [Section L.18, "Troubleshooting Oracle Business Intelligence Reporting"](#) for troubleshooting tips and other useful information about Oracle Business Intelligence

- Oracle Business Intelligence Publisher Enterprise documentation at:

http://www.oracle.com/technology/documentation/bi_pub.html

All the auditing reports available in Oracle Business Intelligence Publisher provide these report filtering and formatting options:

- View - View the report using the current parameters.

- Schedule - Set up a schedule for the report along with job parameters and data filters.
- History
- Edit - Modify the query and parameter display formats.
- Configure - Set up runtime configuration controls.
- Export

14.1.2 Install Oracle Business Intelligence Publisher

If you already have Oracle Business Intelligence Publisher 10.1.3.4 or later installed at your site, you can skip this section and go to [Section 14.1.3, "Set Up Oracle Reports in Oracle Business Intelligence Publisher"](#).

If you need to install Oracle Business Intelligence Publisher, follow the instructions provided with the Oracle Business Intelligence Publisher Companion CD.

See Also: Oracle Business Intelligence Publisher Enterprise documentation at:

http://www.oracle.com/technology/documentation/bi_pub.html

14.1.3 Set Up Oracle Reports in Oracle Business Intelligence Publisher

In this section you configure Oracle Business Intelligence Publisher to work with the audit datasources.

Note: 11g Release 1 (11.1.1.4.0) PS3 reports can work only with an 11g Release 1 (11.1.1.4.0) PS3 schema; they cannot work with an earlier schema such as 11g Release 1 (11.1.1).

Details about upgrading schemas with the Patch Set Assistant are available in the *Oracle Fusion Middleware Patching Guide*.

Take these steps to set up Oracle Business Intelligence Publisher for use with audit reports:

1. Navigate to the Reports folder in your Oracle Business Intelligence Publisher installation. By default, the Reports folder is at %BIP_HOME%\XMLP\Reports.
2. Unjar the AuditReportTemplates.jar into your Reports folder. You should see a new folder called Oracle_Fusion_Middleware_Audit. You can find AuditReportTemplates.jar at:

```
$MW_ORA_HOME/oracle_common/modules/oracle.iau_11.1.1/reports/
AuditReportTemplates.jar
```

3. Set up the data source for audit repository as follows:
 - Navigate to the Admin tab.
 - If you deployed on Oracle WebLogic Server in Step 1, set up JNDI as follows:
 - Click **JNDI Connection**.
 - Click **Add DataSource**.
 - Specify the DataSource details:

Name the Data Source `Audit`.

Note: The reports refer to the audit data source, so the naming convention is important.

JNDI Name - `'jdbc/AuditDB'`

- Test for a successful connection. If the connection is not successful, check the values you entered.
- Press **Apply** to save your changes.
- If you deployed on Oracle Containers for Java EE in Step 1, set up JNDI as follows:
 - Click **JDBC Connection**.
 - Click **Add DataSource**.
 - Specify the DataSource details:

Name the Data Source `Audit`.

Note: The reports refer to the audit data source, so the naming convention is important.

Enter the details for the URL, username, and password for the audit schema. (Note: The username and password consist of the audit schema name including a prefix, for example, username: `dev_iau` or `test_iau`.)

- Test for a successful connection. If the connection is not successful, check the values you entered.
- Press **Apply** to save your changes.

14.1.4 Set Up Audit Report Templates

You can use the standard audit reports in their default formats out-of-the-box. However, if you wish to customize the appearance and other related aspects of the reports, you do so by setting up audit report templates.

From a report's **Edit** dialog, you can click the **Layout** option in the left panel to control layouts and output formats. Using this feature, you can:

- Customize the report template and design your own layout; for example you can rearrange fields and highlight selected field labels.
- Restrict the formats to which the report output is generated - by default, a large number of output formats are available including HTML, PDF, Excel spreadsheet, RTF, and others.

See Also: *Oracle Business Intelligence Publisher User's Guide*.

14.1.5 Set Up Audit Report Filters

You can use the standard audit reports in their default formats out-of-the-box. However, if you wish to customize the scope of data and other related aspects of the reports, you do so by setting up audit report filters.

Oracle Business Intelligence Publisher provides both basic and advanced filtering options for your audit reports.

See Also: *Oracle Business Intelligence Publisher User's Guide.*

Basic Filters

Clicking on the report's **Schedule** button brings up a page which you can use to schedule and administer the report.

In the Report Parameters area you can provide high-level filters to restrict the report:

- Date Filters
 - show only recent audit records such as last hour or last week
 - show records generated within a specified starting and ending dates
 - limit number of records returned
- Selected Report Fields

For example, the Authentication Failures report can be filtered by:

 - Username
 - Component Type
 - Component Name
 - Application Name
 - Domain Name

Advanced Filters

Clicking on the report's **Edit** button brings up a page at which you can specify more detailed report filters and properties. This page consists of two panels. The left panel lets you select what element of the report is to be modified through these options. For each element you select, the right panel displays the corresponding information.

- Data Model - This contains the SQL query that fetches the raw data for the report. The query can be modified according to your needs.
- List of Values - Shows all the report columns. Selecting on a column displays the underlying SQL query that filters data for the attribute. You can modify the query as needed; for example you can specify more restrictive filter values.
- Parameters - Shows all the report columns, and lets you select any column to modify display settings for that column. For example, you can specify a date display format for timestamp fields.
- Layouts and output formats - This feature is described in [Section 14.1.6, "Configure Scheduler in Oracle Business Intelligence Publisher"](#).

14.1.6 Configure Scheduler in Oracle Business Intelligence Publisher

Clicking on the report's **Schedule** button brings up a page which you can use to schedule and administer the report. Information you can specify on this page includes:

Note: This feature assumes that the Oracle Business Intelligence Publisher repository is already configured.

- Report Parameters - filters to restrict the data included in the report, for example records for the last hour only.

See Also: [Section 14.1.5, "Set Up Audit Report Filters"](#)

- Job Properties - the job name, formatting locale and time zone, and so on.
- Notification - one or more users to be notified by E-mail when the job completes or fails.
- Time - report scheduling options; the report can be scheduled to run periodically or on a one-time basis.
- Delivery - deliver the report to one or more users

14.2 Organization of Audit Reports

Oracle Fusion Middleware Audit Framework ships with a set of pre-defined reports that are designed to work, out-of-the-box, with Oracle Fusion Middleware components. These reports are organized into two main categories:

- Common Reports

These reports capture common events such as authentication success and failures, account-related status (lockout, disabled, and so on). Many components have implemented audit capability for these common events. The common reports are located under the Common Reports subfolder of the Audit Reports, and all audit-enabled events from across the components are captured in these reports.

For example, "Authentication History" displays authentication history across all the components where authentication events are being captured.

You can use these reports to examine audit records for a specific area across components or to examine the audit records of a single user across multiple components for that specific area.

- Component-specific Reports

These reports focus on individual components. They are needed because not all audit events may be relevant to each component. The Component Specific folder serves two purposes. First, it identifies the valid reports among the Common Reports that are relevant to the component and show only the audit records for that component. Secondly, for some components, component-specific reports have been defined to suit the specific needs of that component. While audit records themselves are generic for all the components, the representation of an audit record may have component-specific requirements. For example, an access policy may need to be shown in a format to be useful.

For example, you can locate the Authentication History report in the Common folder, where it displays authentication events for all components. You can also find the same report under a component-specific folder, where it displays authentication events for that component only.

- There is also a generic report at the top level called "All Events", which shows all the events across all audit-enabled components. The "All Events" report is also available in each component-specific folder, to show all the events for individual components.

This report can be used to query audit data.

14.3 View Audit Reports

This section explains how to view audit reports using Oracle Business Intelligence Publisher.

Take these steps to view an audit report:

1. Log in to Oracle Business Intelligence Publisher using a URL of the form:
`http://host.domain.com:port/xmlpserver/`
2. On the main page, click **Oracle Fusion Middleware Audit** under Shared Folders.
3. The audit reports are organized into:
 - reports that are common to multiple components; these are further organized by report types
 - reports that are specific to a component; these are further organized by component

See Also: [Table 14-1](#) for a description of the standard reports.

4. Navigate to the report of interest; for example, you can click on the Common Reports folder, then Errors and Exceptions, then click on All Errors and Exceptions.

The report is displayed.

5. The report display page contains these major areas:
 - Filters at the top of the page enable you to determine the type, scope, and number of records to include in the report. These filters include:
 - User
 - Start and End Dates
 - Last n time period
 - Component type and name
 - Application Name
 - Domain Name

Use relevant filters to limit the report to the desired records.

Note: Initially, the report is displayed with default filter values that you can modify.

- Format control buttons enable you to determine:
 - the template type, which can be:
 - HTML - This is the default display format.
 - PDF - Displays a printable PDF view.
 - Data - Displays an unformatted XML data set.

To change the template type while viewing a report, select the type from the drop-down list and click **View**.

- output format

- delivery options
 - The report record display area. The appearance and number of columns depend on previously selected options and filters.
Each column header also acts as a sort option.
6. View, save or export the report as desired.

14.4 Example of Oracle Business Intelligence Publisher Reports

This section uses a common scenario to demonstrate how Oracle Business Intelligence Publisher reports are used to view audit data generated by Oracle Platform Security Services events.

In this example, some activity is generated on the credential store for an Oracle WebLogic Server domain. We then use Oracle Business Intelligence Publisher to take a look at the relevant report to see the audit records. Subsequently, a few other reports are examined.

1. As the system administrator, locate the domain whose credentials are to be managed.
2. Use the relevant commands to generate some credential management records; for example, create and delete some user credentials.

See Also: [Section 10.3, "Managing the Credential Store"](#) for details about credential management.

3. Log in to Oracle Business Intelligence Publisher using a URL of the form:
`http://host.domain.com:port/xmlpserver/`
4. Under the **Reports** tab, click on Shared Folders, and select Fusion Middleware Audit.
5. On the main page, click **Fusion Middleware Audit** under Shared Folders.
6. The audit report menu appears. Audit reports are organized in various folders by type.
7. To view audit records for Oracle Platform Security Services, for example, navigate to the Component Specific folder, then **Oracle Platform Security Services**.
8. The Oracle Platform Security Services folder contains several reports. Click **All Events**.

The report shows activity in a default time range. Modify the time range to show only the day's events.

The activity performed on that day appears on the page.

Credential Management							
User ID	Component Name	Application Name	Timestamp	Map	Key	Message	Event
weblogic			1/11/2009 5:58:46 PM	initiatorMap	initiatorKey	Setting credential succeeded.	CreateCredential Details
weblogic			1/11/2009 5:58:46 PM	mapSimpleServlet	keySimpleServlet	Setting credential succeeded.	CreateCredential Details
weblogic			1/11/2009 5:58:46 PM	*	*	Deleting all credentials from the store succeeded.	DeleteCredential Details
weblogic			1/11/2009 5:58:46 PM	*	*	Deleting all credentials from the store succeeded.	DeleteCredential Details
			1/11/2009 6:05:24 PM	MAP2		Setting credential map succeeded.	CreateCredential Details
			1/11/2009 6:05:24 PM	MAP1	KEY2	Setting credential succeeded.	CreateCredential Details

Observe the different regions of the report and their functions: report filters, format control, scheduling, and the data display itself.

- In each report, the last data column is a Detail column. Click on a detail to view all the attributes of the specific audit record.

Setting credential succeeded.
(View Related Audit Events ...)

Process Details

Host Network Address	140.87.22.71
Host Id	stane09

Event Details

Event Category	CredentialManagement
Event Type	CreateCredential
Event Status	SUCCESS

Request Details

Component Type	JFS
Thread Id	14

Oracle Platform Security

Credential Map	mapSimpleServlet
Credential Key	keySimpleServlet

- Return to the main folder to view some other reports of interest. For example, in the Common Reports folder, navigate to the Account Management folder, and click **Account Profile History**.

The Account Profile History report appears.

Account Profile History						
User ID	Timestamp	Component/ Application Name	Administrator ID	Message	Event	Event Details
cn=fwu	2/27/2009 12:36:37 PM	oidl	cn=orcladmin	Operation name: add	Create ACCOUNT	Detail s...
cn=inda balls,ou=tresury,cnf=accounting,cn=americas,ou=us	2/27/2009 10:00:40 AM	uidl	cn=orcladmin	Operation name: add	Create ACCOUNT	Detail s...
cn=lincoln,ou=us,ou=tresury,ou=accounting,ou=us,ou=americas,ou=us	2/27/2009 10:00:40 AM	oidl	cn=orcladmin	Operation name: add	Create ACCOUNT	Detail s...
cn=lincoln,ou=us,ou=tresury,ou=accounting,cn=americas,ou=us	2/27/2009 10:00:40 AM	oidl	cn=orcladmin	Operation name: add	Create ACCOUNT	Detail s...

- Click on the Event Details for an event of interest:

Operation name: add
(View Related Audit Events ...)

Process Details

Host Network Address	140.87.54.16
Host Id	stalb08
Process Id	196

Event Details

Event Category	Account Management
Event Type	Create Account
Event Status	SUCCESS

Request Details

Event ID	View Related Audit Events ...
Component Type	OID
Component Name	oidl
Client IP Address	140.87.7.191
Session Id	27

Oracle Internet Directory

Event Operation	add
-----------------	-----

- Finally, return to the Common Reports folder and select Errors and Exceptions. Select the **All Errors and Exceptions** report.

- A number of records are displayed. To narrow the report to records of interest, use the Event drop-down to select checkPermission events.

One row is returned showing an authorization check failure:

Event Category: All
 Event: Access Credential
 Sort By: Timestamp
 Order: Descending

Template: Simple | HTML | View | Export | Send | Schedule | Analyzer | Analyzer for Excel | [Link to this report](#)

All Errors and Exceptions						
User ID	Timestamp	Component/ Application Name	Client IP Address	Message	Event	Event Details
	1/29/2009 10:22:33 PM	em		Getting credential failed. Reason access denied (oracle.security.jps.service.credstore.CredentialAccessPermission context=SYSTEM,mapName=EM,keyName=PROXY_INFO read)	Access Credential	Details...

- Click Details to obtain more information:

Getting credential failed. Reason access denied (oracle.security.jps.service.credstore.CredentialAccessPermission context=SYSTEM,mapName=EM,keyName=PROXY_INFO read)

(View Related Audit Events ...)

Process Details

Host Network Address	152.68.65.148
Host Id	stbbm01

Event Details

Event Category	Credential Management
Event Type	Access Credential
Event Status	FAILURE
Timestamp	2009-01-29 10:22:33 PM GMT

14.5 Audit Report Details

This section provides detailed reference information about the standard (pre-built) audit report.

The standard audit reports are grouped as follows:

- The All Events report
 - This report contains all audit records generated in a pre-defined interval.
- Common Reports
 - These are reports that contain audit records across multiple components.
- Component-Specific Reports
 - Each report is dedicated to a specific component.

Common Reports

Common reports are organized as follows:

- Account Management
 - Account Profile History
 - Accounts Deleted
 - Accounts Enabled

- Accounts Disabled
- Accounts Created
- Accounts Locked Out
- Password Changes
- dashboard
- User Activities
 - Authentication History
 - Multiple Logins from Same IP
 - Authorization History
 - Event Details
 - Related Audit Events
 - Dashboard
- Errors and Exceptions
 - All Errors and Exceptions
 - Authorization Failures
 - Authentication Failures
 - Dashboard

Important: Run the Event Details report only against an 11g Release 1 (11.1.1.4.0) PS3 (patch set 3) schema.

Component-Specific Reports

For a list of reports, see [Section C.2.2, "Component-Specific Audit Reports"](#).

14.5.1 List of Audit Reports in Oracle Business Intelligence Publisher

[Table 14–1](#) provides a brief description of each audit report in Oracle Business Intelligence Publisher.

Note: The folder path shown in the column titled "Located in Folder" is relative to the Oracle Fusion Middleware Audit folder. To get to this folder, log in to Oracle Business Intelligence Publisher, and navigate to Shared Folders, then Oracle Fusion Middleware Audit.

Table 14–1 List of Audit Reports

Report	Description	Located in Folder
Accounts Created	shows accounts created in various components	Common Reports, then Account Management. Also in Component Specific folders.
Accounts Deleted	shows accounts deleted in various components	Common Reports, then Account Management. Also in Component Specific folders.

Table 14–1 (Cont.) List of Audit Reports

Report	Description	Located in Folder
Accounts Disabled	shows accounts disabled in various components	Common Reports, then Account Management. Also in Component Specific folders.
Accounts Enabled	shows accounts enabled in various components	Common Reports, then Account Management. Also in Component Specific folders.
Accounts Locked Out	shows accounts locked out due to excessive authentication failures	Common Reports, then Account Management. Also in Component Specific folders.
Account Profile History	shows profile changes in accounts, such as change in address and password changes	Common Reports, then Account Management. Also in Component Specific folders.
All Errors and Exceptions	captures all errors and exceptions across components	Common Reports, then Errors and Exceptions. Also in Component Specific folders.
All Events	displays all audit events	Oracle Fusion Middleware Audit. Also in Component Specific folders.
Application Policy Management	displays application level policy management	Component Specific, then Oracle Platform Security Services.
Application Role Management	shows application role to enterprise role mappings	Component Specific, then Oracle Platform Security Services.
Assertion Activity	Assertion Activity in Oracle Identity Federation	Component Specific, then Oracle Identity Federation.
Assertion Template Management	lists assertion Template management operations in Oracle Web Services Manager	Component Specific, then Oracle Web Services Manager, then Policy Management
Authentication Failures	authentication errors and exceptions; can be cross-component or specific to a component.	Common Reports, then Errors and Exceptions. Also in Component Specific folders.
Authentication History	Authentications across all components	Common Reports, then User Activities. Also in Component Specific folders.
Authorization Failures	captures authorization failures	Common Reports, then Errors and Exceptions. Also in Component Specific folders.
Authorization History	Authorizations across all components	Common Reports, then User Activities. Also in Component Specific folders.
Confidentiality Enforcements	lists enforcements related to confidentiality in Oracle Web Services Manager	Component Specific, then Oracle Web Services Manager, then Policy Enforcements
Configuration Changes	configuration changes made in Fusion Middleware Audit Framework.	Component Specific, then Oracle Fusion Middleware Audit Framework
Credential Access	displays credential accesses by users and applications in Oracle Platform Security Services	Component Specific, then Oracle Platform Security Services.

Table 14–1 (Cont.) List of Audit Reports

Report	Description	Located in Folder
Credential Management	displays credential management operations performed in Oracle Platform Security Services.	Component Specific, then Oracle Platform Security Services.
Federation User Activity	lists federation user activities in Oracle Identity Federation	Component Specific, then Oracle Identity Federation.
Message Integrity Enforcements	shows enforcements related to message integrity in Oracle Web Services Manager	Component Specific, then Oracle Web Services Manager, then Policy Enforcements
Multiple Logins from Same IP	lists machines from where successful logins are made into different user accounts.	Common Reports, then User Activities.
Password Changes	shows password changes done in various accounts.	Common Reports, then Account Management. Also in Component Specific folders.
Policy Attachments	shows Policy to web service endpoint attachments	Component Specific, then Oracle Web Services Manager
Policy Enforcements	general policy enforcements for Oracle Web Services Manager	Component Specific, then Oracle Web Services Manager, then Policy Enforcements
Profile Management Events	shows changes to Directory Integration Platform's profiles.	Component Specific, then Directory Integration Platform.
Request Response	shows requests sent and responses received from web services	Component Specific, then Oracle Web Services Manager
System Policy Management	displays system level policy management operations	Component Specific, then Oracle Platform Security Services.
Violations	Enforcement violations.	Component Specific, then Oracle Web Services Manager, then Policy Enforcements
Web Services Policy Management	shows policy management operations.	Component Specific, then Oracle Web Services Manager, then Policy Management

14.5.2 Attributes of Audit Reports in Oracle Business Intelligence Publisher

[Table 14–2](#) lists the attributes that appear in the various audit reports. When viewing a report, you can use this table to learn more about the attributes that appear in the report.

Note the following:

- Not all attributes appear in each report.
- The user or users attribute, which appears in each report, can mean different things in different reports; see [Table 14–1](#) for an explanation of this attribute.
- Not all the attributes are displayed in Oracle Business Intelligence Publisher audit reports. If you wish to include some additional attributes in your custom reports, see [Appendix C, "Oracle Fusion Middleware Audit Framework Reference"](#).

Table 14–2 Attributes of Audit Reports

Attribute	Description
Activity	The type of action, either user- or system-initiated.
Application Name	The complete application path and name.
Application Server Instance	The instance of the application server in use.
Attempted	The action that was attempted, for example, a single sign-on attempted by the user.
Component Name	The name of the component instance.
Component Type	The type of component, for example Oracle Identity Federation.
Domain Name	Oracle WebLogic Server domain name.
ECID	The execution context ID.
Event Type	The type of event that occurred, for example, account creation.
Initiator	The user who initiated the event.
Internet Protocol Address, IP Address	The IP address of the user's machine from which the action was initiated.
Message Text	The text of the message; a description of the event.
Policy Name	The name of the policy involved in the action.
Time Range	The time range which allows you to limit your data set to a specific time interval, for example, the last 24 hours.
Timestamp	The date and time of the event.
Transaction ID	The transaction identifier.

14.6 Customizing Audit Reports

This section discusses advanced report generation and creation options:

- [Using Advanced Filters on Pre-built Reports](#)
- [Creating Custom Reports](#)

14.6.1 Using Advanced Filters on Pre-built Reports

Clicking on the report's **Edit** button brings up a page at which you can specify more detailed report filters and properties. This page consists of two panels. The left panel lets you select what element of the report is to be modified through these options. For each element you select, the right panel displays the corresponding information.

- **Data Model** - This contains the SQL query that fetches the raw data for the report. The query can be modified according to your needs.
- **List of Values** - Shows all the report columns. Selecting on a column displays the underlying SQL query that filters data for the attribute. You can modify the query as needed; for example you can specify more restrictive filter values.
- **Parameters** - Shows all the report columns, and lets you select any column to modify display settings for that column. For example, you can specify a date display format for timestamp fields.
- **Layouts and output formats** - This feature is described in the following section.

14.6.2 Creating Custom Reports

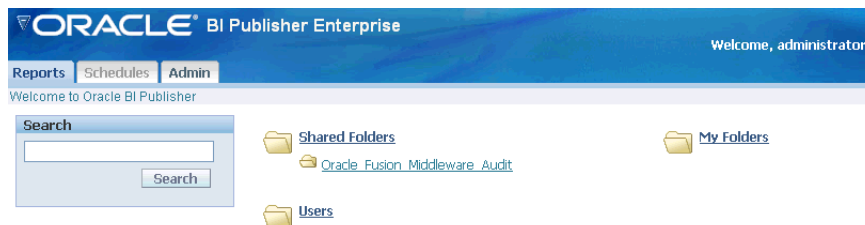
Oracle Business Intelligence Publisher provides a complete set of capabilities for designing and creating custom reports.

See Also:

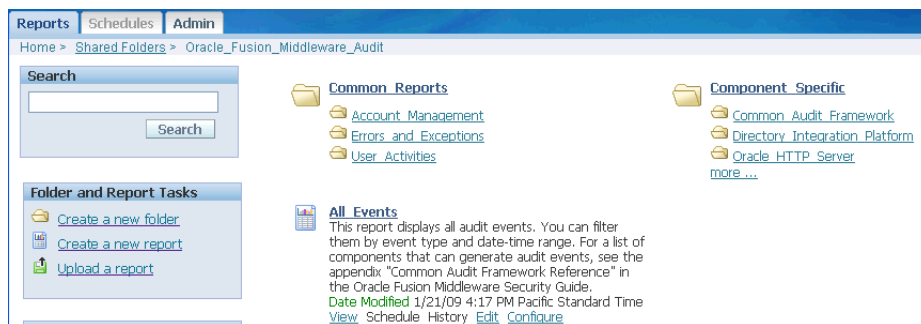
- *Oracle Business Intelligence Publisher User's Guide.*
- [Section C.3, "The Audit Schema"](#)

Here is a simple example illustrating the basic steps to customize an existing audit report with Oracle Business Intelligence Publisher.

1. Log in to Oracle Business Intelligence Publisher as administrator.

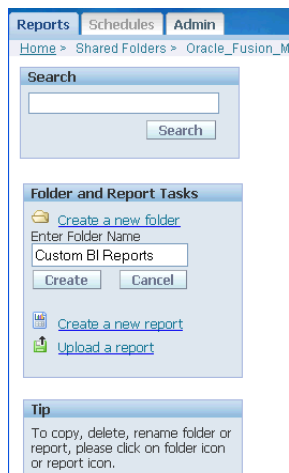


2. Navigate to the Oracle Fusion Middleware Audit folder.



3. Create a folder to maintain your custom reports. Under **Folder and Event Tasks**, click **New Folder**.

Enter a folder name.



- The new folder, Custom BI Reports, appears on the main audit reports folder.

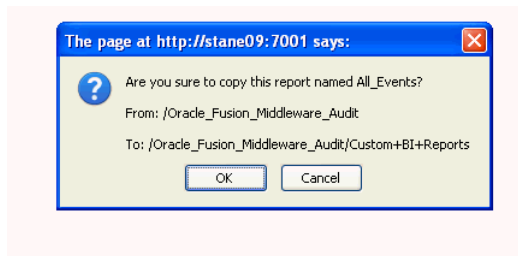


- Select an existing report that will be a starting point to create a custom report, by clicking the icon to the left of the report. In this example the All Events report is selected:



Click **Copy this report**.

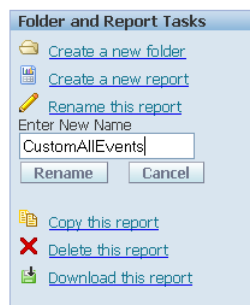
- This action copies the report to the clipboard. To send it to the new folder:
 - Select the Custom BI Reports folder.
 - Under **Folder and Report Tasks**, click **Paste from clipboard**.
 - A dialog box appears requesting confirmation. Click **Yes**.



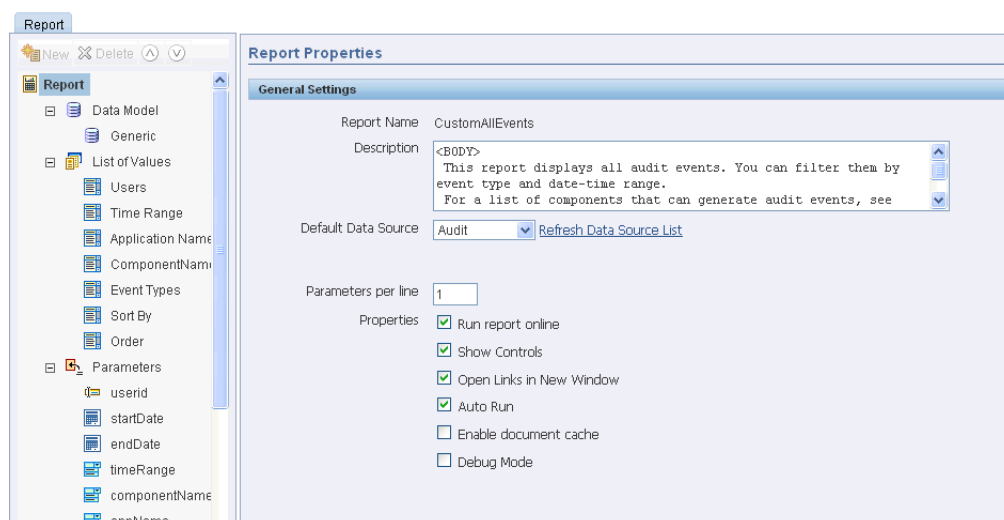
The report is now moved from the clipboard to the custom folder:



- Provide a descriptive name for the new report by selecting the icon to the left of the report, and clicking **Rename this report**.



7. Now you are ready to customize the report. Click **Edit** from the menu choices under the report title.
8. The Edit page appears.

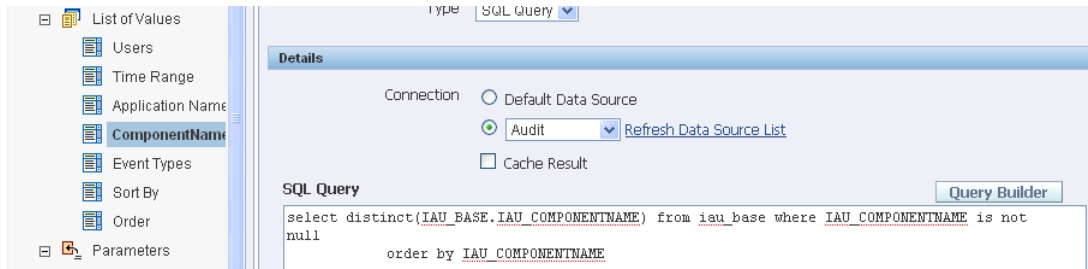


Two panels are displayed; on the main panel titled General Settings, you can control basic features like the report title and runtime controls. To the left of the main panel, a second panel displays two sets of information that you can use to create relevant content for your report:

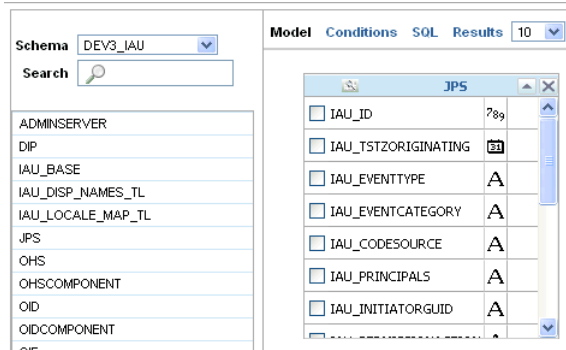
- List of Values shows the fields that are being used currently in the report. When you click on a field, the main panel automatically displays the name and the SQL query used to select the values to include for that field.
- Parameters shows the available parameters from which you can choose the ones to include in the report. Notice that a subset of the parameters is already in the report; for example, `userid` (which is the initiator of the audit event) provides the Users data, while `timeRange` provides the Time Range data.

The palette of choices on the left panel is context-sensitive and provides information to help you build the report.

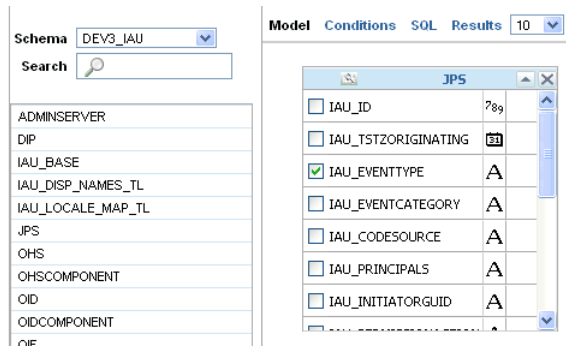
9. You can use the Query Builder to customize the data to include in your report. For example, to include only login events for a component, you can:
 - Select `ComponentName` from the list of values and click **Query Builder**.



- A table appears listing the available components. Select the component, say JPS. A second table appears showing the component event fields:



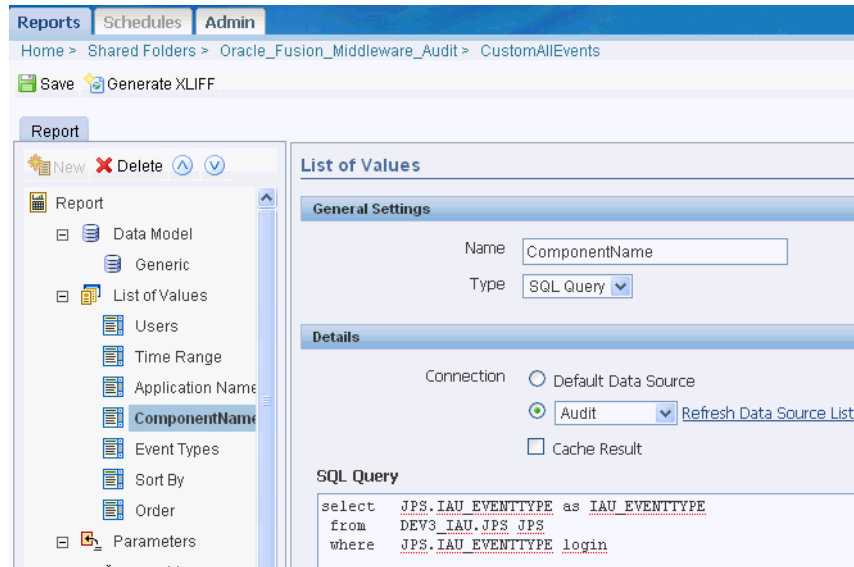
- In the JPS table select IAU_EVENTTYPE.



- Click **Conditions**, enter the condition login and click **Save**.



10. The condition is now included in the report. Be sure to click **Save** again on the upper left corner to commit your changes to the report definition.



11. You can now return to the report in the Custom BI Reports folder and view the data.

Part IV

Single Sign-On Configuration

This part describes how to configure single sign-on in Oracle Fusion Middleware in the following chapters:

- [Chapter 15, "Introduction to Single Sign-On in Oracle Fusion Middleware"](#)
- [Chapter 16, "Configuring Single Sign-On with Oracle Access Manager 11g"](#)
- [Chapter 17, "Configuring Single Sign-On Using Oracle Access Manager 10g"](#)
- [Chapter 18, "Configuring Single Sign-On using OracleAS SSO 10g"](#)

Introduction to Single Sign-On in Oracle Fusion Middleware

The chapter outlines a set of recommended single sign-on solutions for Oracle Fusion Middleware. This chapter includes the following major sections:

- [Choosing the Right SSO Solution for Your Deployment](#)
- [Introduction: OAM Authentication Provider for WebLogic Server](#)
- [Setting Up Debugging in the WebLogic Administration Console](#)

15.1 Choosing the Right SSO Solution for Your Deployment

Oracle Platform Security Services comprise Oracle WebLogic Server's internal security framework. A WebLogic domain uses a separate software component called an Authentication Provider to store, transport, and provide access to security data. Authentication Providers can use different types of systems to store security data. The Authentication Provider that WebLogic Server installs uses an embedded LDAP server.

Oracle Fusion Middleware 11g supports new single sign-on solutions that applications can use to establish and enforce perimeter authentication:

- Oracle Access Manager solutions
- Oracle Single Sign-On (OSSO) solution

Customers must carefully choose the solution appropriate to their needs. Selecting the right SSO solution requires careful consideration and depends upon your requirements. This section outlines some general information and guidelines to help you choose the best solution for your needs.

Note: Oracle recommends that you consider upgrading to Oracle Access Manager 11g Single Sign on solution to take advantage of additional functionality and architecture.

See Also:

- *Oracle Fusion Middleware Security Overview*
- *Oracle Fusion Middleware Administrator's Guide for Oracle Access Manager with Oracle Security Token Service*

- **Development or Small Stand-Alone Environment:** Oracle recommends a light-weight SSO solution when deployed applications are not integrated into an enterprise-level single sign-on framework.

In such cases, a SAML-based solution that uses the Oracle WebLogic Server SAML Credential Mapping Provider is best. The embedded LDAP server is used as the default user repository. Alternatively, an LDAP Authenticator can be configured to leverage an external LDAP server as a user repository.

See Also: "Configuring Single Sign-On with Web Browsers and HTTP Clients" in *Oracle Fusion Middleware Securing Oracle WebLogic Server*

- **Enterprise-Level SSO with Oracle Fusion Middleware 11g:** Oracle Access Manager supports:
 - A wide variety of LDAP vendors as the user and group repository and also works with Oracle Virtual Directory
 - Integration with non-Oracle application server vendors and Web Tier components on a large variety of OS platforms to provide a flexible solution.
 - Oracle Access Manager 11g supports out-of-the-box integration with Oracle Fusion Middleware applications

Oracle Access Manager 11g (Release 1): Oracle recommends Oracle Access Manager 11g whether:

- You are new to Oracle Fusion Middleware
- You are considering a migration from OSSO
- You are considering an enterprise-level SSO solution
- You want to implement Identity Propagation with the OAM Token, as described in the *Oracle Fusion Middleware Administrator's Guide for Oracle Access Manager with Oracle Security Token Service*

Oracle Access Manager 10g (10.1.4.3): You can continue using this when you have:

- Existing Oracle Access Manager 10g implementations
- An enterprise-level SSO solution

Selecting the right Oracle Access Manager solution (11g versus 10g (10.1.4.3)) as your enterprise-level Single-Sign-on solution depends upon your requirements. Refer to product documentation in this chapter and in the respective administration guides to evaluate the release that best meets your overall requirements.

See Also: ["Introduction: OAM Authentication Provider for WebLogic Server"](#) on page 15-4

- **Existing OSSO 10g Customers:** Oracle Single Sign-On is part of the 10g Oracle Application Server suite. OSSO is an enterprise-level single sign-on solution that works with the OC4J application server in conjunction with Oracle Internet Directory and Oracle HTTP Server 11g.

If OSSO is already in place as the enterprise solution for your existing Oracle deployment, Oracle Fusion Middleware continues to support the existing OSSO as a solution. However, Oracle recommends that you consider upgrading to Oracle Access Manager 11g Single Sign on solution, which is a strategic Oracle SSO

solution. For more information when planning your upgrade, check the Lifetime Support Middleware Policy for the OSSO end of support dates at:

<http://www.oracle.com/support/lifetime-support-policy.html>

See Also:

- ["Introduction: OAM Authentication Provider for WebLogic Server" on page 15-4](#)
- *Oracle Fusion Middleware Upgrade Planning Guide*
- *Oracle Fusion Middleware Upgrade Guide for Oracle Identity Management*—For information about the types of Java EE environments available in 10g and instructions for upgrading those environments to Oracle Fusion Middleware 11g
- **Portal, Forms, Reports, and Discoverer 11g:** Oracle Access Manager 11g is certified with Oracle Portal, Forms, Reports, and Discover 11g. With Oracle classic components, Oracle Delegated Administration Services 10g is a required and important feature of the Oracle Identity Management infrastructure.

See the *Oracle Identity Management Guide to Delegated Administration* in the Oracle Identity Management 10g (10.1.4.0.1) Online Documentation Library at:

<http://www.oracle.com/technology/documentation/oim1014.html>

See the Oracle Fusion Middleware Supported System Configurations page for more details:

http://www.oracle.com/technology/software/products/ias/files/fusion_certification.html

See Also: The following topics and other 11g manuals:

- ["Introduction: OAM Authentication Provider for WebLogic Server" on page 15-4](#)
- [Chapter 18, "Configuring Single Sign-On using OracleAS SSO 10g"](#)
- *Oracle Fusion Middleware Administrator's Guide for Oracle Portal*
- *Oracle Fusion Middleware Forms Services Deployment Guide*
- *Oracle Fusion Middleware Publishing Reports to the Web with Oracle Reports Services*
- *Oracle Fusion Middleware Administrator's Guide for Oracle Business Intelligence Discoverer*
- **Oracle Access Manager Integration with OSSO:** Oracle recommends Oracle Access Manager 11g as the recommended enterprise-wide solution. If applications (Oracle Portal for example) are deployed that previously required OracleAS Single Sign-On, you can delegate the authentication (from OSSO 10g) to Oracle Access Manager 11g. Oracle Internet Directory is needed for applications that require integrating Oracle Access Manager and OSSO.

See Also:

- ["Introduction: OAM Authentication Provider for WebLogic Server"](#) on page 15-4
 - *Oracle Fusion Middleware Administrator's Guide for Oracle Access Manager* for details about registering OSSO (mod_osso) Agents with Oracle Access Manager 11g to delegate authentication and for details about co-existence with Oracle Access Manager 11g during the OSSO 10g upgrade.
 - *Oracle Fusion Middleware Upgrade Guide for Java EE*—For information about the types of Java EE environments available in 10g and instructions for upgrading those environments to Oracle Fusion Middleware 11g
 - "Integrating with Oracle Application Servers" in the 10g (10.1.4.3) *Oracle Access Manager Integration Guide*.
-
- **Windows Native Authentication for Microsoft Clients:** OSSO and Oracle Access Manager 11g both support this integration. Oracle WebLogic Server can be configured to use the Simple and Protected Negotiate (SPNEGO) mechanism for authentication to provide Windows Native Authentication support.

See Also:

- The chapter on configuring Oracle Access Manager 11g to use Windows Native Authentication for Microsoft Clients in the *Oracle Fusion Middleware Integration Guide for Oracle Access Manager*
- "Configuring Single Sign-On with Microsoft Clients" in *Oracle Fusion Middleware Securing Oracle WebLogic Server*

15.2 Introduction: OAM Authentication Provider for WebLogic Server

Unless explicitly stated, information here applies equally to both Oracle Access Manager 11g and 10g deployments.

The Oracle Access Manager Authentication Provider is one of several Providers that operate with Oracle WebLogic Server. The Oracle Access Manager Authentication Provider does not require the entire Oracle WebLogic Suite nor Oracle Java Required Files (JRF) to operate with Oracle Access Manager 11g or 10g.

In a WebLogic Server domain where JRF is installed, the JRF template is present as part of the domain in an Oracle Fusion Middleware product. In this case, the OAM Identity Asserter and OAM Authentication Provider are automatically available for configuration. If JRF is not installed in your WebLogic domain, you must add the OAMAuthnProvider.jar to a specific location in your domain as described later.

Note: The JRF template is present as part of the domain in an Oracle Fusion Middleware product.

You can use the OAM Authentication Provider for WebLogic Server when you have:

- Applications that are (or will be) deployed in a WebLogic container outside the Identity Management domain
- WebGate is (or will be) deployed in front of the Authentication Provider

The Authentication Provider can be configured to provide either (or both) of the following functions for WebLogic users:

- **Identity Asserter for Single Sign-on** function
- **Authenticator** function

Identity Asserter for Single Sign-on Function

When the application is protected using a perimeter Webgate, the identity of the authenticated user that is communicated to the WebLogic Server is made available to container security layers using the Oracle Access Manager identity asserter. The Identity Asserter only asserts the incoming identity and then passes control to the configured Authentication Providers to continue with the rest of the authentication process (populating the subject with the right principals).

Note: A Web-only applications implementation handles nearly all SSO use cases. The exception is when you have Oracle Web Services Manager protected Web services. In this case, there is no trusted WebGate. Instead the AccessGate provided with the Identity Asserter is contacted and interacts with your OAM 10g Access Server or 11g OAM Server; all other processing is essentially the same.

Oracle provides the following mechanisms, each with slightly different characteristics and requirements:

- **Trusted Header Assertion:** This newest mechanism, for use with Oracle Access Manager 11.1.1.5.2 or later (and either a 10g or 11g Webgate), is triggered for the OAM_IDENTITY_ASSERTION token present for applications protected by 11g or 10g WebGate. This provides maximum security and is easy to configure.
- **Clear Text Header:** This default mechanism is triggered for the OAM_REMOTE_USER token present for applications protected by 10g or 11g WebGate.
- **Session Token:** This mechanism is available for use with only perimeter 10g Webgates and either the 10g Access Server or 11g OAM Server.

Table 15–1 lists the benefits and requirements for each.

See Also: ["About Using the Identity Asserter Function with Oracle Access Manager"](#) on page 15-6

Table 15–1 Summary: Identity Assertion Mechanisms for Oracle Access Manager

Mechanism	Benefits	Requirements
Trusted Header Assertion OAM_IDENTITY_ASSERTION	Maximum security Easy configuration	Oracle Access Manager 11.1.1.5.2 or later 10g or 11g Webgate
Clear Text Header OAM_REMOTE_USER	Maximum performance Default Mechanism	Oracle Access Manager 11.1.1.5.0 10g or 11g Webgate
Session Token (ObSSOCookie) To be deprecated	10g Webgate with either OAM 10g or 11g Server	Oracle Access Manager 11.1.1.3.0 Oracle Access Manager 10.1.4.3 10g Webgate

Authenticator Function

The Authenticator function does not provide single sign-on. The Authenticator requests credentials from the user based on the authentication method specified in the application configuration file, `web.xml`, not according to the Oracle Access Manager authentication scheme. However, an Oracle Access Manager authentication scheme is required for the application domain.

Note: You can skip this topic if you are using the Identity Asserter function.

For more information, see the following topics:

- [About Using the Identity Asserter Function with Oracle Access Manager](#)
- [About Using the Authenticator Function with Oracle Access Manager](#)
- [Choosing Applications for Oracle Access Manager SSO Scenarios and Solutions](#)

15.2.1 About Using the Identity Asserter Function with Oracle Access Manager

This topic describes and illustrates the use of the Identity Asserter function with Oracle Access Manager 11g and 10g WebGates. Processing is similar, with few exceptions, whether you have OAM 11g with 11g (or 10g) WebGates or OAM 10g with 10g WebGates). For instance, with Oracle Access Manager 11g, the Access Server is known as the OAM Server.

All requests are first routed to a reverse proxy Web server and requests are intercepted by WebGate. The user is challenged for credentials based on the authentication scheme that is configured within Oracle Access Manager. Oracle recommends Form (form-based login) as the authentication scheme.

The Identity Asserter function relies on perimeter authentication performed by WebGate on the Web Tier. Triggering the Identity Asserter function requires the appropriate chosen Active Type for your WebGate release.

After triggering the Identity Asserter function, configured Authentication Providers (Login Modules) for constructing the Subject and populating it with the appropriate Principals are invoked.

Note: The only difference between using the Identity Asserter function with 11g WebGates versus 10g WebGates is the provider's chosen Active Type.

Chosen Active Types

The Identity Asserter function's Active Type configuration parameter lists supported values under the Available UI section. One of the following must be selected as the "Chosen" type to trigger the Identity Asserter function:

See Also: [Table 15–1, "Summary: Identity Assertion Mechanisms for Oracle Access Manager"](#)

- `Identity Assertion`: Triggers Identity Assertion based on the trusted header `OAM_IDENTITY_ASSERTION`.
- `OAM_REMOTE_USER`: Triggers Identity Assertion based on `OAM_REMOTE_USER` header.

- `obSSOCookie`: Triggers Identity Assertion based on the `obSSOCookie`.

`OAM_REMOTE_USER` header includes the uid of the logged in user. Configuring `OAM_REMOTE_USER` as the chosen Active Type for the Identity Asserter requires Oracle Access Manager policies that set `OAM_REMOTE_USER` as part of the authorization success response headers.

Authentication Processing and the Identity Assertion Function

Unless explicitly stated, information here applies equally to Oracle Access Manager 11g and Oracle Access Manager 10g.

WebGate, using the configured authentication scheme, authenticates the user, and then:

- WebGate:
 - 11g WebGate sets the `OAMAuthnCookie` and triggers the token (either `OAM_IDENTITY_ASSERTION` or `OAM_REMOTE_USER`).
 - 10g WebGate triggers assertion based on the `obSSOCookie` or `OAM_REMOTE_USER` or `OAM_IDENTITY_ASSERTION` are possible
- The OHS Web server `mod_weblogic` module forwards the request to Oracle WebLogic Server

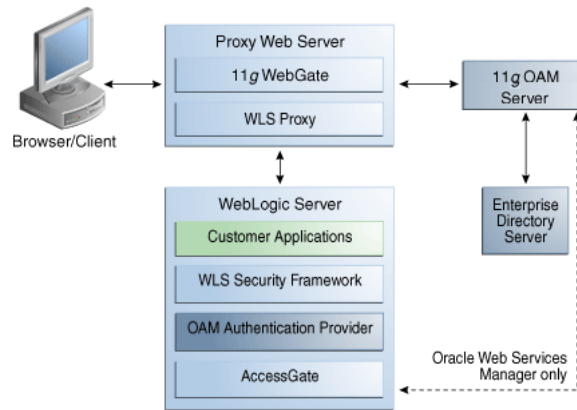
Note: `mod_weblogic` is the generic name of the WebLogic Server plug-in for Apache. For Oracle HTTP Server 11g, the name of this plug-in is `mod_wl_ohs`; the actual binary name is `mod_wl_ohs.so`.

- The Identity Asserter is invoked when the configured Active token type is present in the request coming into the container: `OAM_REMOTE_USER` (default), `obSSOCookie`, `OAM_IDENTITY_ASSERTION`.
- After Assertion Processing: Authentication Providers configured in the security realm are invoked to populate the 'Subject' with Principals (Users and Groups)

[Figure 15-1](#), and the overview that follows, describe processing between components when the Identity Asserter function is used with Web-only applications. This implementation handles nearly all SSO use cases. **Exception:** Oracle Web Services Manager protected Web services. In this case, there is no trusted WebGate. Instead the AccessGate provided with the Identity Asserter (dotted line in [Figure 15-1](#)) is contacted and interacts with the 11g OAM Server (or 10g OAM Access Server); all other processing is essentially the same.

For more information, see "[Oracle Access Manager Authentication Provider Parameter List](#)" on page 17-14.

[Figure 15-1](#) illustrates the processing overview using the Identity Asserter configuration with Oracle Access Manager 11g and

Figure 15–1 Identity Asserter Configuration with Oracle Access Manager and WebGates

Assertion takes place based on which token type is configured in the authorization policy. Alone, the presence of token in the request is not sufficient to invoke the asserter. Simply configuring a particular active token type in WebLogic is not sufficient. `OAM_IDENTITY_ASSERTION` will be set in the request if it is configured in the authorization policy.

Process overview: Identity Assertion with OAM 11g, 11g WebGate, and Web-only applications

1. A user attempts to access an Oracle Access Manager protected Web application that is deployed on the Oracle WebLogic Server.
2. WebGate on a reverse proxy Web server intercepts the request and queries the OAM Server to determine whether the requested resource is protected.
3. If the requested resource is protected, WebGate challenges the user for credentials based on the type of Oracle Access Manager authentication scheme configured for the resource (Oracle recommends Form Login). The user presents credentials such as user name and password.
4. WebGate forwards the authentication request to the OAM Server.
5. OAM 11g Server validates user credentials against the primary user identity store and returns the response to WebGate (OAM 10g Access Server validates user credentials against configured user directories). Upon:
 - **Successful Authentication:** Processing continues with Step 6.
 - **Authentication Not Successful:** The login form appears asking the user for credentials again; no error is reported.
6. OAM Server generates the session token and sends it to the WebGate:

11g WebGate: Sets and returns the `OAMAuthn` cookie and triggers the `OAM_REMOTE_USER` (or `OAM_IDENTITY_ASSERTER`) token when policies are configured for this.

10g WebGate: Sets and returns `OAM_REMOTE_USER` or `OAM_IDENTITY_ASSERTION` headers in the request when policies are configured for this.

The Web server forwards this request to the proxy, which in turn forwards the request to the Oracle WebLogic Server using the `mod_weblogic` plug-in.

`mod_weblogic` forwards requests as directed by its configuration.

Note: `mod_weblogic` is the generic name of the WebLogic Server plug-in for Apache For Oracle HTTP Server 11g, the name of this plug-in is `mod_wl_ohs`.

7. WebLogic Server security service invokes the Oracle Access Manager Identity Asserter which is configured to accept tokens of type "OAM_REMOTE_USER" (or "OAM_IDENTITY_ASSERTER"). The Identity Asserter initializes a `CallbackHandler` with the header. In addition, the Identity Asserter sets up `NameCallback` with the username for downstream LoginModules.
8. Oracle WebLogic Security service authorizes the user and allows access to the requested resource.
9. A response is sent back to the reverse proxy Web server.
10. A response is sent back to the browser.

15.2.2 About Using the Authenticator Function with Oracle Access Manager

This topic describes and illustrates use of the Authenticator configured to protect access to Web and non-Web resources with Oracle Access Manager.

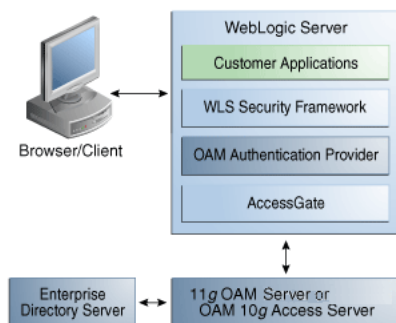
Note: Unless explicitly stated, information applies equally to Oracle Access Manager 11g and Oracle Access Manager 10g.

The Authenticator function relies on Oracle Access Manager services to authenticate users who access applications deployed in WebLogic Server. Users are authenticated based on their credentials, such as a user name and password.

When a user attempts to access a protected resource, the Oracle WebLogic Server challenges the user for credentials according to the authentication method specified in the application's `web.xml` file. Oracle WebLogic Server then invokes the Authentication Provider, which passes the credentials to Oracle Access Manager Access Server for validation through the enterprise directory server.

Figure 15-2 illustrates the distribution of components and flow of information for Oracle Access Manager authentication for Web and non-Web resources. Details follow the figure. In this case, the Authenticator communicates with the 11g OAM Server (or the OAM 10g Access Server) through a custom `AccessGate`.

Figure 15-2 Authenticator for Web and non-Web Resources



Surrounding text describes this graphic.

Process overview: Authenticator Function for Web and non-Web Resources

1. A user attempts to access a Java EE application (secured with the authentication mechanism in the application's web.xml file) that is deployed on the Oracle WebLogic Server.
2. Oracle WebLogic Server intercepts the request.
3. Oracle Access Manager Authentication Provider LoginModule is invoked by the Oracle WebLogic security service. The LoginModule uses the OAP library to communicate with the 11g OAM Server (or 10g Access Server) and validate the user credentials.
 - If the user identity is authenticated successfully, WLSUserImpl and WLSGroupImpl principals are populated in the Subject.
 - If Oracle Access Manager LoginModule fails to authenticate the identity of the user, it returns a LoginException (authentication failure) and the user is not allowed to access the Oracle WebLogic resource.
4. Oracle Access Manager Authenticator supports Oracle WebLogic Server UserNameAssertion.
5. Oracle Access Manager Authenticator can be used with any Identity Asserter. In this case, the Oracle Access Manager Authenticator performs user name resolution and gets the roles and groups associated with the user name.

See Also:

- ["Configuring the Authenticator Function for Oracle Access Manager 11g"](#) on page 16-29
- ["Configuring the Authenticator for Oracle Access Manager 10g"](#) on page 17-49

15.2.3 Choosing Applications for Oracle Access Manager SSO Scenarios and Solutions

This section introduces choosing applications to use Oracle Access Manager and the Authentication Provider according to current application setup. Details are similar whether you plan to use Oracle Access Manager 11g or 10g with the Authentication Provider:

- [Applications Using Oracle Access Manager for the First Time](#)
- [Applications Migrating from Oracle Application Server to Oracle WebLogic Server](#)
- [Applications Using OAM Security Provider for WebLogic SSPI](#)

15.2.3.1 Applications Using Oracle Access Manager for the First Time

If your application is to use Oracle Access Manager Authentication Provider for the first time, proceed based on the functionality that you want to use:

- **Identity Asserter for Single Sign-On:** The Web-only applications implementation handles nearly all SSO use cases. See ["Installing the Authentication Provider with Oracle Access Manager 11g"](#) on page 16-8.

Oracle Web Services Manager-Protected Web Services: This requires the AccessGate that is provided with the Identity Asserter to interact with the OAM

Server. See ["Configuring Identity Assertion for Oracle Web Services Manager and OAM 11g"](#) on page 16-35.

- **Authenticator:** No single sign-on is provided. The Authenticator requests credentials from the user based on the authentication method specified in the application configuration file, `web.xml`. See ["Configuring the Authenticator Function for Oracle Access Manager 11g"](#) on page 16-29.

15.2.3.2 Applications Migrating from Oracle Application Server to Oracle WebLogic Server

If your application has been deployed on the old Oracle Application Server (OC4J), you can perform a few steps to make the application use the Authentication provider with Oracle WebLogic Server, proceed as follows:

- Remove all OC4J-specific settings from the application configuration
- **Identity Asserter for Single Sign-On:** The Web-only applications implementation handles nearly all SSO use cases. See the appropriate topic for your environment:
 - OAM 11g: ["Configuring Identity Assertion for SSO with Oracle Access Manager 11g"](#) on page 16-16
 - OAM 10g: ["Configuring OAM Identity Assertion for SSO with Oracle Access Manager 10g"](#) on page 16-16
- **Oracle Web Services Manager-Protected Web Services:** Require the AccessGate provided with the Identity Asserter. See the appropriate topic for your environment:
 - OAM 11g: ["Configuring Identity Assertion for Oracle Web Services Manager and OAM 11g"](#) on page 16-35
 - OAM 10g: ["Configuring Identity Assertion for Oracle Web Services Manager and OAM 10g"](#) on page 17-60
- **Authenticator:** No single sign-on is provided. The Authenticator requests credentials from the user based on the authentication method specified in the application configuration file, `web.xml`. See the appropriate topic for your environment:
 - OAM 11g: ["Configuring the Authenticator Function for Oracle Access Manager 11g"](#) on page 16-29
 - OAM 10g: ["Configuring the Authenticator for Oracle Access Manager 10g"](#) on page 17-49

15.2.3.3 Applications Using OAM Security Provider for WebLogic SSPI

The Oracle Access Manager Security Provider for WebLogic SSPI provides authentication, authorization, and single sign-on across Java EE applications that are deployed in the WebLogic platform. The Security Provider for WebLogic SSPI enables WebLogic administrators to use Oracle Access Manager to control user access to business applications.

Note: Security Provider for WebLogic SSPI is also known as "Security Provider" in the 10g (10.1.4.3) Oracle Access Manager Integration Guide.

The Oracle Access Manager Security Provider for WebLogic SSPI provides authentication to Oracle WebLogic Portal resources and supports single sign-on between Oracle Access Manager and Oracle WebLogic Portal Web applications. Apart from this, the Security Provider for WebLogic SSPI also offers user and group management functions.

The Oracle Access Manager Authentication Provider is more easily installed and configured than the Security Provider for WebLogic SSPI. The Authentication Provider offers authentication and single sign-on (SSO) services, and also works with all platforms supported by Oracle WebLogic Server.

If your application has been using the Oracle Access Manager Security Provider for WebLogic SSPI for only authentication and SSO, the deployment is a good candidate for the latest Authentication Provider. However, if your application relies on features other than those offered by the latest Oracle Access Manager Authentication Provider, you can continue to use the Oracle Access Manager 10g Security Provider for WebLogic SSPI.

Note: WebLogic SSPI connector can be used with Oracle Access Manager 10g but is not supported with Oracle Access Manager 11g

See Also: ["Applications Using OAM Security Provider for WebLogic SSPI"](#) on page 15-11

15.2.4 Implementation: Using the Provider with OAM 11g versus OAM 10g

With a very few differences, implementing solutions is similar whether you are using OAM 11g or OAM 10g to protect for applications in a WebLogic container.

[Table 15–2](#) outlines the differences when deploying the Authentication Provider with OAM 11g versus OAM 10g. Topic headings are highlighted.

Table 15–2 Differences in Authentication Provider Implementation Tasks for OAM 11g versus OAM 10g

OAM 11g Implementation Details	OAM 10g Implementation Details
<p>Included in the OAM 11g implementation are the following tasks, which are described in the <i>Oracle Fusion Middleware Administrator's Guide for Oracle Access Manager with Oracle Security Token Service</i>:</p> <ul style="list-style-type: none"> ▪ Installing the Authentication Provider with Oracle Access Manager 11g ▪ Previewing Pre-Seeded OAM 11g Policies for Use by the 10g AccessGate ▪ Session Token: Provisioning an OAM Agent with Oracle Access Manager 11g (ObSSOCookie only) <p>Note: The OAM 11g remote registration tool automates provisioning WebGates and policies. For WebLogic Server resources, a wl_authen resource type is created by default.</p> <ul style="list-style-type: none"> ▪ Configuring Identity Assertion for SSO with Oracle Access Manager 11g <p>Or</p> <p>Configuring Identity Assertion for Oracle Web Services Manager and OAM 11g</p> <p>Or</p> <p>Configuring the Authenticator Function for Oracle Access Manager 11g</p> <ul style="list-style-type: none"> ▪ Configuring Centralized Log Out for Oracle Access Manager 11g 	<p>Tasks for implementing SSO solutions with OAM 10g are described in this chapter:</p> <ul style="list-style-type: none"> ▪ Installing and Setting Up Authentication Providers for OAM 10g ▪ Configuring OAM Identity Assertion for SSO with Oracle Access Manager 10g <p>Note: OAM 10g OAMCfgTool automates provisioning WebGates and policies.</p> <p>Install 10g WebGate: <i>Oracle Fusion Middleware Administrator's Guide for Oracle Access Manager with Oracle Security Token Service</i>.</p> <ul style="list-style-type: none"> ▪ Configuring the Authenticator for Oracle Access Manager 10g requires manual policy domain creation ▪ Configuring Identity Assertion for Oracle Web Services Manager and OAM 10g ▪ Configuring Global Logout for Oracle Access Manager 10g and 10g WebGates

15.2.5 Requirements for the Provider with Oracle Access Manager

The required components and files for implementing the Authentication Provider are nearly identical whether you have OAM 11g or OAM 10g as the SSO solution. The few exceptions are noted in the following list:

- An enterprise directory server (Oracle Internet Directory or Oracle Sun One directory server) for Oracle Access Manager and Oracle WebLogic Server.
- Oracle WebLogic Server 10.3.1+ to be configured to use the Oracle Access Manager Authentication Provider as described later in this chapter.
- **Optional:** A Fusion Middleware product (Oracle Identity Manager, Oracle SOA Suite, or Oracle Web Center for example).
- **Authentication Provider:** For applications deployed in a WebLogic container, Oracle Access Manager JAR and WAR files are available when you install an Oracle Fusion Middleware product (Oracle Identity Management, Oracle SOA Suite, or Oracle WebCenter).

Note: With a stand-alone Oracle WebLogic Server (no Fusion Middleware), you must obtain the Authentication Provider JAR and WAR files from Oracle Technology Network as described in Step 1 of procedures later in this chapter.

- **oamAuthnProvider.jar:** Includes files for both the Oracle Access Manager Identity Asserter for single sign-on and the Authenticator for Oracle WebLogic Server 10.3.1+. A custom Oracle Access Manager AccessGate is also provided to process requests for Web and non-Web resources (non-HTTP) from users or applications.
- **oamauthenticationprovider.war:** Restricts the list of providers that you see in the Oracle WebLogic Server Console to only those needed for use with Oracle Access Manager.

When you deploy the extension, the WebLogic Administration Console creates an in-memory union of the files and directories in its WAR file with the files and directories in the extension WAR file. Once the extension is deployed, it is a full member of the WebLogic Administration Console: it is secured by the WebLogic Server security realm, it can navigate to other sections of the Administration Console, and when the extension modifies WebLogic Server resources, it participates in the change control process. For more information, see the *Oracle Fusion Middleware Extending the Administration Console for Oracle WebLogic Server*.

- **Oracle Access Manager 11g:** A remote registration command-line utility streamlines WebGate provisioning and creates a fresh application domain with security policies. Administrators can specify WebGate parameters and values using a template.
- **Oracle Access Manager 10g:** The platform-agnostic OAMCfgTool and scripts (oamcfgtool.jar) automate creation of the Oracle Access Manager form-based authentication scheme, policy domain, access policies, and WebGate profile for the Identity Asserter for single sign-on. OAMCfgTool requires JRE 1.5 or 1.6. Internationalized login forms for Fusion Middleware applications are supported with the policies protecting those applications.
- **OHS 11g** must be configured as a reverse proxy for the WebGate (required by the Oracle Access Manager Identity Asserter)

- **Oracle Access Manager:**
 - OAM 11g:** Deployed with initial configuration using the Oracle Fusion Middleware Configuration Wizard, as described in Oracle Fusion Middleware Installation Guide for Oracle Identity Management. See "[Deploying the Oracle Access Manager 11g SSO Solution](#)" on page 16-7.
 - OAM 10g:** Installed with initial setup as described in *Oracle Access Manager Installation Guide*. See "[Deploying SSO Solutions with Oracle Access Manager 10g](#)" on page 17-1.
- **WebGate/AccessGate:** Whether you need to provision a WebGate or an AccessGate with Oracle Access Manager depends on your use of the OAM Authentication Provider:
 - Identity Asserter for Single Sign-On:** Requires a separate WebGate for each application to define perimeter authentication.
 - Authenticator (or Oracle Web Services Manager):** Requires the custom 10g AccessGate that is available with the Authentication Provider.

15.3 Setting Up Debugging in the WebLogic Administration Console

The Authentication Providers use messages with verbose descriptions of low-level activity within the application when Debug mode issued. Ordinarily, you do not need this much information. However, if you must call Oracle Support, you might be advised to set up debugging. When set, Authentication Providers messages appear in the Oracle WebLogic Server default log location.

To set up debugging

1. Log into WebLogic Administration Console.
2. Go to Domain, Environment, Servers, *yourserver*.
3. Click the Debug tab.
4. Under Debug Settings for this Server, click to expand the following: **weblogic, security, atn**.
5. Click the option beside DebugSecurityAtn to enable it.
6. Save Changes.
7. Restart the Oracle WebLogic Server.
8. In the Oracle WebLogic Server default log location, search for SSOAssertionProvider. For example:

```
####<Apr 10, 2009 2:32:16 AM PDT> <Debug> <SecurityAtn> <sta00483>
<AdminServer> <[ACTIVE]>
ExecuteThread: '0' for queue: 'weblogic.kernel.Default (self-tuning) '>
<<WLS Kernel>> <> <> <1239355936490> <BEA-000000>
<SSOAssertionProvider:Type          = Proxy-Remote-User>
```

Configuring Single Sign-On with Oracle Access Manager 11g

The chapter provides information on configuring single sign-on using Oracle Access Manager 11g. It includes the following major sections:

- [Introduction to Oracle Access Manager 11g SSO](#)
- [Deploying the Oracle Access Manager 11g SSO Solution](#)
- [Configuring Centralized Log Out for Oracle Access Manager 11g](#)
- [Synchronizing the User and SSO Sessions: SSO Synchronization Filter](#)
- [Troubleshooting Tips](#)

16.1 Introduction to Oracle Access Manager 11g SSO

Oracle Access Manager 11g is part of Oracle's enterprise class suite of security products. Intended for use in new and existing SSO deployments, Oracle Access Manager 11g provides a full range of Web perimeter security functions that include Web single sign-on; authentication and authorization; policy administration, and more.

Oracle Access Manager 11g single sign-on (SSO) and single log-out (SLO) supports a variety of application platforms including:

- SOA
- WebCenter

Oracle Access Manager 11g supports integration with a variety of applications, as described in the *Oracle Fusion Middleware Integration Guide for Oracle Access Manager*.

- Oracle Identity Navigator
- Oracle Identity Federation
- Oracle Identity Manager
- Oracle Adaptive Access Manager

As described in the *Oracle Fusion Middleware Administrator's Guide for Oracle Access Manager with Oracle Security Token Service*, Oracle Access Manager 11g differs from Oracle Access Manager 10g in that identity administration features have been transferred to Oracle Identity Manager 11g. This includes user self-service and self registration, workflow functionality, dynamic group management, and delegated identity administration.

Console Protection for Oracle Identity Management Applications

Oracle Access Manager 11g and other Oracle Identity Management applications are deployed in a WebLogic container. Individual administration consoles include Oracle Access Manager, Oracle Adaptive Access Manager, Oracle Identity Navigator, Oracle Identity Manager, Oracle WebLogic Server, and Oracle Entitlements Server.

These are protected by default using pre-configured Authentication Providers in the WebLogic Administration Console and a pre-registered IAMSuiteAgent with Oracle Access Manager 11g. OAM 11g SSO policies are pre-seeded. No further configuration is needed for the consoles.

See Also: *Oracle Fusion Middleware Administrator's Guide for Oracle Access Manager with Oracle Security Token Service*

Preview of OAM 11g Deployments

You can configure Oracle Access Manager in a new WebLogic administration domain or in an existing WebLogic administration domain using the Oracle Fusion Middleware Configuration Wizard.

See "[Requirements for the Provider with Oracle Access Manager](#)" on page 15-13

See Also: *Oracle Fusion Middleware Installation Guide for Oracle Identity Management*

Oracle Access Manager 11g provides new server-side components that maintain backward compatibility with new or existing policy-enforcement agents. Dynamic Server-initiated updates are performed for any policy or configuration changes.

- Oracle Access Manager Console (installed on WebLogic Administration Server) replaces the OAM 10g Policy Manager
- OAM Server (installed on a WebLogic Managed Server; replaces the OAM 10g Access Server)

Oracle Access Manager 11g provides single sign-on (SSO), authentication, authorization, and other services to registered Agents (in any combination) protecting resources:

- 11g WebGates
- 10g WebGates
- Java-based IAMSuiteAgent
- OSSO Agents (10g mod_osso)

You can integrate with Oracle Access Manager 11g, any Web applications currently using Oracle ADF Security and the OPSS SSO Framework.

Only users with sufficient privileges can log in to the Oracle Access Manager Administration Console or use OAM administrative command-line tools. Your enterprise might require independent sets of administrators: one set of users responsible for OAM administration and a different set for WebLogic administration. For more information, see "Defining a New OAM Administrator Role" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Access Manager with Oracle Security Token Service*.

Overview of OAM 11g

The following outlines some of the basic features of Oracle Access Manager 11g:

Provisioning/Remote Registration: A new remote registration tool enables administrators inside or outside the network to register agents and policies. A username and password must be set in the primary User Identity Store for OAM 11g.

Authentication: Oracle Access Manager 11g application domains aggregate resources and security policies (one policy per resource). Oracle Access Manager 11g authentication policies include a specific scheme. Supported authentication modules include LDAP, X.509, and Kerberos. Authentication user mapping is performed against the primary user-identity provider by the centralized credential collector.

Authorization: Oracle Access Manager 11g performs authorization based on security policies defined in the application domain and persisted in the database. Authorization policies define the resource and constraint evaluation.

Responses: Administrators can set session attributes using authentication and authorization Responses. Aside from session attributes, a Response can also obtain user-related data and request-related data. Responses, once set, are then sent as either HTTP Headers or Cookies to the agent that helps manifest them. For cookie values and header variables, Responses can retrieve session attributes previously set by another Response. For example, session attributes set by a Response upon authentication can be retrieved as a header value during authorization.

Session Management: Oracle Access Manager 11g session management services track active user sessions through a high performance distributed cache system based on technology from Oracle Coherence. Each Oracle Access Manager runtime instance is a node within the distributed cache system. Secure communication between the nodes is facilitated using a symmetric key. The Oracle Access Manager runtime instances move user session data in the local cache into the distributed cache for other nodes to pick up. Each Oracle Access Manager runtime instance can also configure the replication factor and determine how session data is distributed.

Administrators can configure the session lifecycle, locate and remove specific active sessions, and set a limit on the number of concurrent sessions a user can have at any time. Out-of-band session termination prevents unauthorized access to systems when a user has been terminated.

Keys: The Oracle Access Manager 11g runtime is deployed as an application to a WebLogic Managed Server or Cluster. New Oracle Access Manager 11g WebGates support a shared secret per agent trust model. 11g WebGates use agent/host specific cookies, which offers superior security. Oracle Access Manager 11g WebGates are all trusted at the same level; a cookie specific for the WebGate is set and cannot be used to access any other WebGate-protected applications on a user's behalf. Cookie-replay types of attacks are prevented.

SSO and SLO: The Oracle Access Manager 11g Server Session Token forms the basis for SSO between Oracle Access Manager and OSSO Agents. Logout is driven through Oracle Access Manager 11g Server Global Logout, which terminates the central session and logs out the user from each agent that was visited.

- With Oracle Access Manager 10g WebGates, logout removes the ObSSOCookie and then redirects to the Global Logout page.
- With Oracle Access Manager 11g WebGates and mod_osso agents, logout redirects to the Global Logout page and each agent is called back to remove the agent-specific cookie.

Logging and Auditing: Oracle Access Manager 11g components use the same logging infrastructure and guidelines as any other component in Oracle Fusion Middleware 11g. Oracle Access Manager 11g provides agent and server monitoring functions. Oracle Access Manager 11g auditing functions are based on the Common Audit

Framework; audit-report generation is supported using Oracle Business Intelligence Publisher.

Access Tester: The new Oracle Access Manager 11g Access Tester enables IT professionals and administrators to simulate interactions between registered Oracle Access Manager Agents and Servers. This is useful when testing security policy definitions or troubleshooting issues involving agent connections.

Transition from Test to Production: Oracle Access Manager 11g enables moving configuration or policy data from one Oracle Access Manager 11g deployment to another (from a small test deployment to a production deployment, for example). Support for the creation of new topologies is based on templates. You can also copy and move policy changes.

Co-existence and Upgrades for OSSO 10g: The Oracle-provided Upgrade Assistant scans the existing OracleAS 10g SSO server configuration, accepts as input the 10g OSSO policy properties file and schema information, and transfers configured partner applications into the destination Oracle Access Manager 11g SSO.

See Also:

- *Oracle Fusion Middleware Administrator's Guide for Oracle Access Manager with Oracle Security Token Service* for an "Introduction to Post-Upgrade Co-existence Between Oracle Access Manager 11g and OSSO 10g Servers"
- *Oracle Fusion Middleware Upgrade Planning Guide*
- *Oracle Fusion Middleware Upgrade Guide for Oracle Identity Management*

16.1.1 Previewing Pre-Seeded OAM 11g Policies for Use by the 10g AccessGate

This topic is required for only 10g custom AccessGates. Skip this topic if it does not apply to your environment.

The `Application Authenticator` application domain is delivered with OAM 11g. It is pre-seeded with the policy objects that enables integration with applications deployed in WebLogic environments using the OAM Authentication Provider as the security provider. It is not associated with WebGate provisioning. When you provision a WebGate or AccessGate to use this (or another existing application domain), you will decline having policies created automatically.

The `Application Authenticator` application domain comes into play with the custom 10g AccessGate used with the OAM Authenticator (and the Identity Asserter for Oracle Web Services Manager). In this case, the custom AccessGate (not WebGate) contacts the WebLogic Server directly with a token to authenticate the user before OAM 11g is contacted.

The `Application Authenticator` application domain protects only resources of type `wl_authen` and is seeded with two authentication policies and one authorization policy. The following `wl_authen` resources are also seeded in this domain:

- `/Authen/Basic`
- `/Authen/SSOToken`
- `/Authen/UsernameAssertion` protected by `LDAPNoPasswordValidationScheme`

Note: Only resources of type `wl_authen` are allowed in this domain; no other resource types can be added. Policies and Responses for `wl_authen` resources can be added. However, ideally, you will not need to modify this domain.

Figure 16–1 illustrates details of the seeded `Application Authenticator` application domain in the OAM 11g Administration Console. The page shown describes the pre-seeded `User ID Assertion` authentication policy, which protects the `/Authen/UsernameAssertion` resource. The authentication scheme for this policy is also shown along with the resources that are protected by the policy.

Figure 16–1 Pre-seeded Resources in the User ID Assertion Authentication Policy

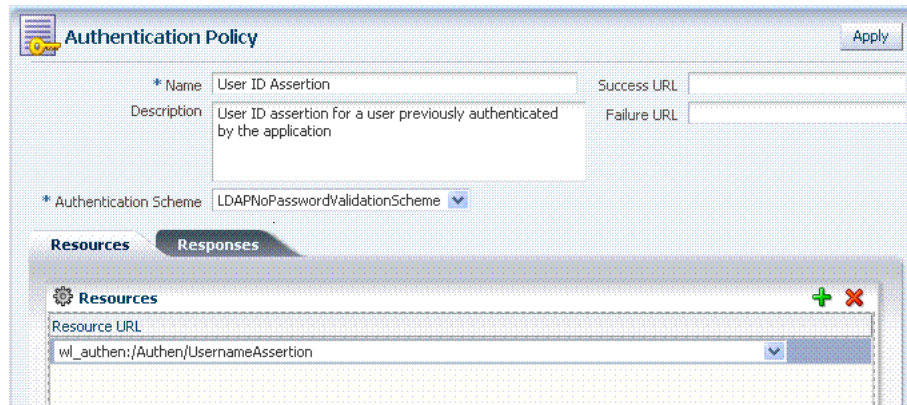


Figure 16–2 illustrates pre-seeded Responses for the `User ID Assertion` authentication policy. For more information about Responses, see the *Oracle Fusion Middleware Administrator's Guide for Oracle Access Manager with Oracle Security Token Service*.

Figure 16–2 Pre-seeded Responses in the User ID Assertion Policy

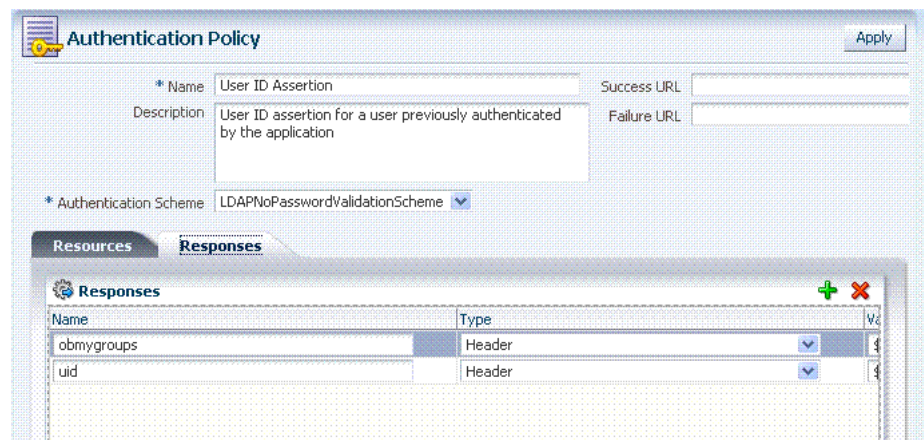


Figure 16–3 illustrates the pre-seeded `Application SSO` authentication policy, the resources protected by this policy, and the authentication scheme.

Figure 16–3 Pre-seeded Application SSO Authentication Policy and Resources

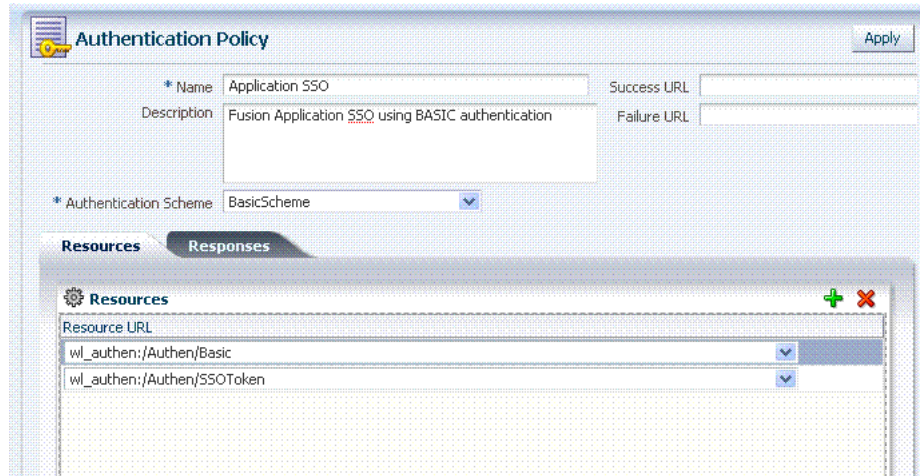


Figure 16–4 illustrates Pre-seeded Responses for the Application SSO authentication policy in the application domain.

Figure 16–4 Pre-seeded Responses for the Application SSO Authentication Policy

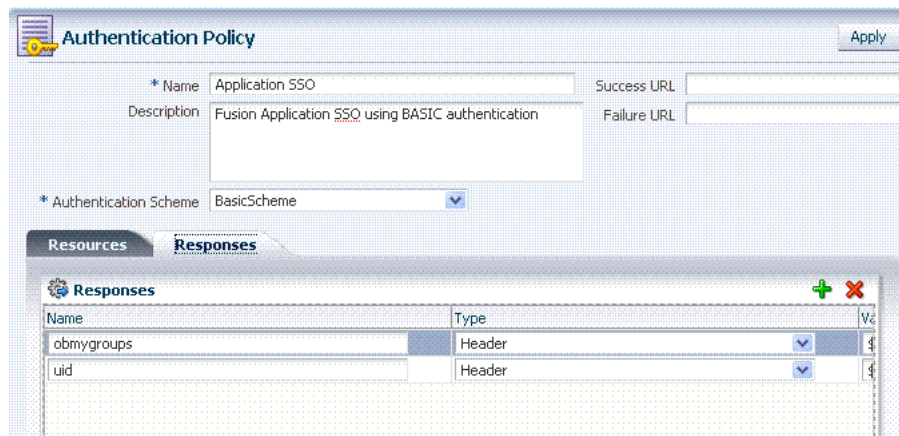
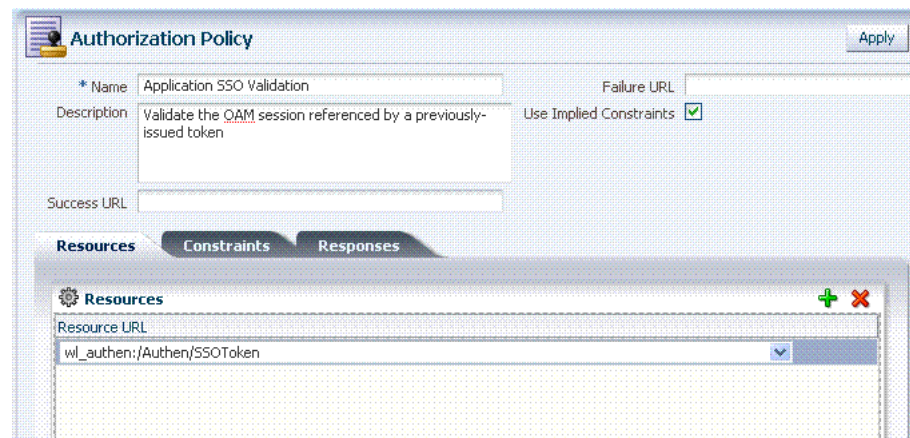


Figure 16–5 illustrates the pre-seeded Application SSO authorization policy and Resources in the application domain.

Figure 16–5 Pre-seeded Application SSO Authorization Policy and Resources

Authorization Constraints: There are no pre-seeded Application SSO authorization policy Constraints in this application domain. However, you can add constraints as described in the *Oracle Fusion Middleware Administrator's Guide for Oracle Access Manager with Oracle Security Token Service*.

Authorization Responses: There are no pre-seeded Application SSO authorization policy Responses in the application domain. However, you can add responses as described in the *Oracle Fusion Middleware Administrator's Guide for Oracle Access Manager with Oracle Security Token Service*.

See Also:

- [Configuring the Authenticator Function for Oracle Access Manager 11g](#)
- [Configuring Identity Assertion for Oracle Web Services Manager and OAM 11g](#)

16.2 Deploying the Oracle Access Manager 11g SSO Solution

This section introduces how to implement OAM 11g with the Authentication Provider when you have applications that are (or will be) deployed in a WebLogic container.

This section provides the following topics to help you implement OAM 11g SSO when you have applications deployed in a WebLogic container. Aside from these uniquely OAM 11g methods, implementing OAM solutions are the same whether you have OAM 11g or OAM 10g:

- [Installing the Authentication Provider with Oracle Access Manager 11g](#)
- [Session Token: Provisioning an OAM Agent with Oracle Access Manager 11g](#)
- [Configuring Identity Assertion for SSO with Oracle Access Manager 11g](#)
- [Configuring the Authenticator Function for Oracle Access Manager 11g](#)
- [Configuring Identity Assertion for Oracle Web Services Manager and OAM 11g](#)
- [Configuring Centralized Log Out for Oracle Access Manager 11g](#)

See Also: *Oracle Fusion Middleware Administrator's Guide for Oracle Access Manager with Oracle Security Token Service* for details about the scenario for Identity Propagation with the OAM Token.

16.2.1 Installing the Authentication Provider with Oracle Access Manager 11g

The following overview outlines the tasks that must be completed to install the required components and files for the Oracle Access Manager 11g SSO solution using the Authentication Provider. While many of these tasks are nearly the same for Oracle Access Manager 11g and Oracle Access Manager 10g, there are a few differences.

See Also: *Oracle Fusion Middleware Installation Guide for Oracle Identity Management* for installation and initial configuration details for Oracle Access Manager 11g.

Task overview: Installing components for use with the Authentication Provider and OAM 11g

1. Install and set up Oracle Internet Directory for Oracle Access Manager.

See Also:

- *Oracle Fusion Middleware Installation Guide for Oracle Identity Management*
- *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*

2. Install and set up Oracle WebLogic Server 10.3.1+.

See Also: Item 3 in this list, and the *Oracle Fusion Middleware Getting Started With Installation for Oracle WebLogic Server*

3. **Optional:** Install a Fusion Middleware product (Oracle Identity Manager, Oracle SOA Suite, or Oracle Web Center for example):

Note: Without a Fusion Middleware application, you must acquire the required JAR and WAR files as described in later procedures.

4. Install OHS 11g for the Oracle Access Manager WebGate, if needed:

- **Identity Asserter:** Requires Oracle HTTP Server 11g Web server configured as a reverse proxy in front of Oracle WebLogic Server.

WebGate: For identity assertion with the OAM Identity Asserter, a perimeter Webgate is required (installed and configured) on the OHS Web Server.

- **Authenticator or Oracle Web Services Manager:** No Web server is required for the custom AccessGate. The protected resource is accessed using its URL on the Oracle WebLogic Server.

5. **Authentication Provider Files:** Confirm the required JAR and WAR files as follows:

- a. Confirm the location of required JAR files in the following Fusion Middleware path:

ORACLE_INSTANCE/modules/oracle.oamprovider_11.1.1/oamAuthnProvider.jar

- b. Locate the console-extension WAR file in the following path:

ORACLE_INSTANCE/modules/oracle.oamprovider_11.1.1/oamauthenticationprovider.war

- c. Copy the WAR file to the following path in the WebLogic Server home:

```
WL_HOME/server/lib/console-ext/autodeploy/oamauthenticationprovider.war
```

6. Oracle Access Manager 11g:

- a. Install Oracle Access Manager and perform initial configuration as described in *Oracle Fusion Middleware Installation Guide for Oracle Identity Management*.
- b. **Trusted Header Assertion:** Go to My Oracle Support, retrieve Bundle Patch 02 (Oracle Access Manager Bundle Patch 11.1.1.5.2), and apply it as described in the companion readme file: <http://support.oracle.com>.

7. AccessGate for the Authenticator (or for Oracle Web Services Manager):

- You can provision the 10g AccessGate as described in "Session Token: Provisioning an OAM Agent with Oracle Access Manager 11g" on page 16-12 (or refer to an existing OAM Agent registration when configuring the Authentication Provider).
- Deploy the custom 10g AccessGate available in `oamAuthnProvider.jar`

16.2.2 Converting Oracle Access Manager Certificates to Java Keystore Format

Oracle recommends that all Java components and applications use JKS as the keystore format. This topic provides steps to convert Oracle Access Manager X.509 certificates to Java Keystore (JKS) format. These steps, when followed properly, generate the JKS stores that can be used while the Java NAP client wants to communicate with an OAM Server in Simple or Cert (certificate) mode.

Note: This procedure is required regardless of the SSO mechanism you choose.

When communicating in Simple or Cert mode, the OAM Server uses a key, server certificate, and CA chain files:

- `aaa_key.pem`: the random key information generated by the certificate-generating utilities while it sends a request to a Root CA. This is your private key. The certificate request for WebGate generates the certificate-request file `aaa_req.pem`. You must send this WebGate certificate request to a root CA that is trusted by the OAM Server. The root CA returns the WebGate certificates, which can then be installed either during or after WebGate installation.
- `aaa_cert.pem`: the actual certificate for the OAM Server, signed by the Root CA.
- `aaa_chain.pem`: the public certificate of the Root CA. This is used when peers communicating in Simple or Cert mode perform an SSL handshake and exchange their certificates for validity. In Simple Mode, the `aaa_chain.pem` is the OpenSSL certificate located in `OAMServer_install_dir/access/oblix/tools/openssl/simpleCA/cacert.pem`

Here, *aaa* is the name you specify for the file (applicable only to Cert and chain files).

You can edit an existing certificate with a text editing utility to remove all data except that which is contained within the `CERTIFICATE` blocks. You then convert the edited certificate to JKS format, and import it into the keystore. Java KeyTool does not allow you to import an existing Private Key for which you already have a certificate. You must convert the PEM format files to DER format files using the OpenSSL utility.

To convert an Oracle Access Manager certificate to JKS format and import it

1. Install and configure Java 1.6 or the latest version.
2. Copy the following files before editing to retain the originals:
 - `aaa_chain.pem`
 - `aaa_cert.pem`
 - `cacert.pem`, only if configuring for Simple mode
3. Edit `aaa_chain.pem` using TextPad to remove all data except that which is contained within the `CERTIFICATE` blocks, and save the file in a new location to retain the original.

```
-----BEGIN CERTIFICATE-----  
...  
CERTIFICATE  
...  
-----END CERTIFICATE-----
```

4. Run the following command for the edited `aaa_chain.pem`:

```
JDK_HOME\bin\keytool" -import -alias root_ca -file aaa_chain.pem -keystore  
rootcerts
```

Here you are assigning an alias (short name) **root_ca** to the key. The input file `aaa_chain.pem` is the one that you manually edited in step 3. The keystore name is `rootcerts`.

You must give a password to access the keys stored in the newly created keystore.

Note: To ensure security, Oracle recommends that you allow the keytool to prompt you to enter the password. This prompt occurs automatically when the "-storepass" flag is omitted from the command line.

5. Enter the keystore password, when asked. For example:

```
Enter keystore password: <keystore_password>  
Re-enter new keystore password: <keystore_password>
```

6. Enter Yes when asked if you trust this tool:

```
Trust this certificate? [no]: yes
```

7. Confirm that the certificate has been imported to the JKS format by executing the following command and then the password.

```
JDK_HOME\bin\keytool" -list -v -keystore "rootcerts"  
Enter keystore password: <keystore_password>
```

8. Look for a response like the following:

```
Keystore type: JKS  
Keystore provider: SUN  
Your keystore contains n entries  
Alias name: root_ca  
Creation date: April 19, 2009  
Entry type: trustedCertEntry
```

```
Owner: CN=NetPoint Simple Security CA - Not for General Use, OU=NetPoint,
O="Obliv, Inc.", L=Cupertino, ST= California , C=US
```

```
Issuer: CN=NetPoint Simple Security CA - Not for General Use, OU=NetPoint,
O="Obliv, Inc.", L=Cupertino, ST= California ,C=US
```

```
Serial number: x
Valid from: Tue Jul 25 23:33:57 GMT+05:30 2000 until: Sun Jul 25 23:33:57
GMT+05:30 2010
```

Certificate fingerprints

```
MD5: CE:45:3A:66:53:0F:FD:D6:93:AD:A7:01:F3:C6:3E:BC
SHA1: D6:86:9E:83:CF:E7:24:C6:6C:E1:1A:20:28:63:FE:FE:43:7F:68:95
Signature algorithm name: MD5withRSA
Version: 1
*****
```

9. Repeat steps 3 through 7 for the other PEM files (except `aaa_chain.pem` unless there is a chain).
10. Convert the `aaa_key.pem` file to DER format using the OpenSSL utility in the OAM Server installation directory path. For example:

```
OAM_Server_HOME\access\oblix\tools\openssl>openssl pkcs8 -topk8
-nocrypt -in aaa_key.pem -inform PEM -out aaa_key.der -outform DER
```

Here the input file is `aaa_key.pem` and the output file is `aaa_key.der`. Additional options include:

Table 16–1 Options to Create DER Format Files from PEM

Option	Description
-topk8	Reads a traditional format private key and writes a PKCS#8 format key. This reverses the default situation where a PKCS#8 private key is expected on input and a traditional format private key is written.
-nocrypt	An unencrypted PrivateKeyInfo structure is expected for output.
-inform	Specifies the input format. If a PKCS#8 format key is expected on input, then either a DER or PEM encoded version of a PKCS#8 key is expected. Otherwise the DER or PEM format of the traditional format private key is used.
-outform	Specifies the output format. If a PKCS#8 format key is expected on output, then either a DER or PEM encoded version of a PKCS#8 key is expected. Otherwise the DER or PEM format of the traditional format private key is used.

11. **Simple or Cert Mode:** In the PEM file (in this case, `aaa_cert.pem`), enter the passphrase for the OAM Server if it is configured for Simple or Cert mode.

```
Passphrase for the certificate
```

12. Run the following command to convert the `aaa_cert.pem` file to DER format.

```
AccessServer_install_dir\access\oblix\tools\openssl>openssl x509 -in
aaa_cert.pem -inform PEM -out aaa_cert.der -outform DER
```

13. Import the DER format files into a Java keystore using the `ImportKey` utility. For example:

```
Java_install_dir\doc>java -Dkeystore=jks certs ImportKey aaa_key.der
```

```
aaa_cert.der
```

14. Review the results in the window, which should look something like the following example:

```
Using keystore-file : jkscerts
One certificate, no chain
Key and certificate stored
Alias:importkey Password:your_password
```

16.2.3 Session Token: Provisioning an OAM Agent with Oracle Access Manager 11g

This task is required for only the session token mechanism (ObSSOCookie). If you are implementing either a trusted header assertion or clear text header mechanism, skip this topic.

Provisioning is the process of registering an agent and creating an application domain to use OAM 11g authentication and authorization services. You must provision a WebGate with OAM 11g whether you are preparing to install a fresh 11g or 10g instance or you have a legacy 10g WebGate installed.

The term WebGate is used for WebGates (and for the custom 10g AccessGates used with the Authenticator and the Identity Asserter for Oracle Web Services Manager). Unless explicitly stated, topics apply equally to both.

When you have multiple agents, each one can be provisioned independently or you can use a single OAM Agent registration for multiple agents.

Note: The Application Authenticator application domain is pre-seeded and delivered with OAM 11g. When you provision an OAM Agent to use this (or another existing) application domain, decline the option of having policies automatically created.

The following topics are provided:

- [About WebGate Provisioning Methods for Oracle Access Manager 11g](#)
- [Provisioning a WebGate with Oracle Access Manager 11g](#)

16.2.3.1 About WebGate Provisioning Methods for Oracle Access Manager 11g

This task is required for only the session token mechanism (ObSSOCookie). If you are implementing either a trusted header or clear text header mechanism, skip this topic.

[Table 16–2](#) outlines the methods and tools you can use to provision WebGates for use with OAM 11g. The remote registration tool enables you to specify a small amount or all WebGate parameters using templates.

Table 16–2 Provisioning Methods for OAM 11g

Method	Description
Oracle Access Manager Administration Console	Enables OAM Administrators to manually enter information and set parameters directly in Oracle Access Manager. This method is required if you are using the Authenticator, or if you have Oracle Web Services Manager policies protecting Web services.

Table 16–2 (Cont.) Provisioning Methods for OAM 11g

Method	Description
Remote Registration	<p>Application administrators who are implementing the Identity Asserter for single sign-on, can register the WebGate using the command line. This also creates a new application domain with security policies for a fresh or existing Web Tier.</p> <p>Required parameters are provisioned using values for your environment specified in a template. Default values are accepted for non-required parameters. After registration, values can be modified in the Oracle Access Manager Console.</p>

During remote registration, you must provide the details discussed in [Table 16–3](#).

See Also: *Oracle Fusion Middleware Administrator's Guide for Oracle Access Manager with Oracle Security Token Service* for a complete list of WebGate parameters

Table 16–3 Required Registration Details for OAM Agents

OAM Agent Element	Description
<serverAddress>	Points to a running instance of the Oracle Access Manager Administration Console, including the host and port.
<webDomain> OSSO requests only	Defines the Web server domain under which the Agent Base URL is stored internally.
<agentName>	<p>Defines a unique identifier for the agent on the OAM (Administration) Server.</p> <p>For every agent on the same server instance, this tag must be unique to avoid re-registering the same agent.</p> <p>Re-registering an agent on the same server instance is not supported.</p>
<hostIdentifier>	This identifier represents the Web server host. The field is filled in automatically when you specify a value for the OAM Agent Name. If the agent name or host identifier of the same name already exists, an error occurs during registration.
<protectedResourcesList>	Specifies the resource URLs that you want the OAM Agent to protect with some authentication scheme. The resource URLs should be relative paths to the agentBaseUrl.
<publicResourcesList>	Specifies the resource URLs that you want to keep public (not protected by the OAM Agent). The resource URLs should be relative paths to the agentBaseUrl. For instance, you might want to specify the Home page or the Welcome page of your application

16.2.3.2 Provisioning a WebGate with Oracle Access Manager 11g

This task is required for only the session token mechanism (ObSSOCookie). If you are implementing either a trusted header or clear text header mechanism, skip this topic.

Provisioning a WebGate or AccessGate involves the same steps. You can provision a new instance for use with the Authentication Provider or you can refer to an existing registration when configuring the provider.

In this example, an OAM 10g WebGate is provisioned using the OAMRequest_short.xml template. The registered agent is named *my-wl-agent1*, protecting */.../**, and declaring a public resource, */public/index.html*. Your values will be different.

Note: When provisioning an OAM 11g WebGate, use the OAM11gRequest_short.xml template.

See Also: *Oracle Fusion Middleware Administrator's Guide for Oracle Access Manager with Oracle Security Token Service*

To provision a WebGate with OAM 11g

1. Acquire the Tool: On the computer to host the WebGate, acquire the remote registration tool and set up the script for your environment. For example:
 - a. Locate RREG.tar.gz file in the following path:


```
WLS_home/Middleware/domain_home/oam/server/rreg/client/RREG.tar.gz
```
 - b. Untar RREG.tar.gz file to any suitable location. For example:


```
rreg/bin/oamreg.
```
 - c. In the oamreg script, set the following environment variables based on your situation (client side or server side) and information in Table 6–7 in the *Oracle Fusion Middleware Administrator's Guide for Oracle Access Manager with Oracle Security Token Service*:


```
OAM_REG_HOME = exploded_dir_for_RREG.tar/rreg
JDK_HOME = Java_location_on_the_computer
```

2. Create the registration request:
 - a. Locate the *Request_short.xml file and copy it to a new location and name. For example:


```
WLS_home/Middleware/domain_home/oam/server/rreg/bin/oamreg/
```

Copy: OAMRequest_short.xml (or OAM 11gRequest.xml)

To: *my-wl-agent1.xml*
 - b. Edit *my-wl-agent1.xml* to include details for your environment, and set automatic policy creation to false. For example:


```
<OAMRegRequest>
  <serverAddress>http://sample.us.oracle.com:7001</serverAddress>
  <hostIdentifier>my-wl</hostIdentifier>
  <agentName>my-wl-agent1</agentName>
  <primaryCookieDomain>.us.example.com</primaryCookieDomain>
  <autoCreatePolicy>false</autoCreatePolicy>
  <logoutUrls><url>/oamssso/logout.html</url></logoutUrls>
</OAMRegRequest>
```

See Also: "Creating the Registration Request" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Access Manager with Oracle Security Token Service*

3. Provision the agent. For example:
 - a. Locate the remote registration script.


```
Linux: rreg/bin/oamreg.sh
Ensure the script has executable permission: chmod +x oamreg.sh

Windows: rreg\bin\oamreg.bat
```
 - b. From the directory containing the script, execute the script using inband mode. For example:

```
$ ./bin/oamreg.sh inband input/my-wl-agent1.xml
```

```
Welcome to OAM Remote Registration Tool!
Parameters passed to the registration tool are:
Mode: inband
Filename: ...
```

- c. When prompted, enter the following information using values for your environment:

```
Enter your agent username: username
Username: username
Enter agent password: *****
Do you want to enter a Webgate password?(y/n)
n
iv.Do you want to import an URIs file?(y/n)
n
```

- d. Review the final message to confirm that this was a successful registration:

```
Inband registration process completed successfully! Output artifacts are
created in the output folder"
```

4. Confirm in the Console: Log in to the Oracle Access Manager Console and review the new registration:

- a. From the OAM 11g Console System Configuration tab, Access Manager Settings section, expand the SSO Agents nodes to search for the agent you just provisioned:

```
Access Manager Settings
  SSO Agents
    OAM Agents
      Search
```

- b. In the Search Results table, click the agent's name to display the registration page and review the details, which you will use later. For example:

Agent Name—During WebGate installation, enter this as the WebGate ID. If you deploy the custom 10g AccessGate, enter this as the AccessGate Name when configuring the OAM Authentication Provider in the WebLogic Administration Console.

Access Client Password—During WebGate installation, enter this as the WebGate password. If no password was entered, you can leave the field blank.

Access Server Host Name—Enter the DNS host name for the primary OAM 11g Server with which this WebGate is registered.

- c. **OAM Proxy Port**—From the Oracle Access Manager Console, System Configuration tab, Common Configuration section, open Server Instances and locate the port on which the OAM Proxy is running.

5. Ignore the Obaccessclient.xml file, which is created during provisioning, for now.

6. Proceed as needed for your environment:

- **Agent is Installed:** Go to the appropriate topic for your implementation:
 - [Configuring Identity Assertion for SSO with Oracle Access Manager 11g](#)
 - [Configuring the Authenticator Function for Oracle Access Manager 11g](#)

- [Configuring Identity Assertion for Oracle Web Services Manager and OAM 11g](#)
- [Configuring Centralized Log Out for Oracle Access Manager 11g](#)
- **Agent is Not Installed:**
 - 11g WebGate: See *Oracle Fusion Middleware Installation Guide for Oracle Identity Management*.
 - 10g WebGate: See *Oracle Fusion Middleware Administrator's Guide for Oracle Access Manager with Oracle Security Token Service*.

16.2.4 Configuring Identity Assertion for SSO with Oracle Access Manager 11g

This section describes the unique steps needed to configure Oracle Access Manager 11g Identity Assertion for Single Sign-On with your application.

Task overview: Deploying the Identity Asserter for SSO with OAM 11g includes

1. Finishing all prerequisite tasks for the mechanism you are implementing:
 - [Installing the Authentication Provider with Oracle Access Manager 11g](#)
 - [Converting Oracle Access Manager Certificates to Java Keystore Format](#)
 - [Session Token: Provisioning an OAM Agent with Oracle Access Manager 11g](#)
2. [Establishing Trust with Oracle WebLogic Server](#)
3. [Configuring Providers in the WebLogic Domain](#)
4. [Trusted Header Assertion: Configuring Digital Signature Verification](#)
5. [Trusted Header Assertion: Configuring Policies](#)
6. [Configuring Centralized Log Out for Oracle Access Manager 11g](#)
7. [Synchronizing the User and SSO Sessions: SSO Synchronization Filter](#)
8. [Testing Oracle Access Manager Identity Assertion for Single Sign-on](#)

16.2.4.1 Establishing Trust with Oracle WebLogic Server

The following topics explain the tasks you must perform to set up the application for single sign-on with the Oracle Access Manager Identity Asserter.

Task overview: Establishing Trust with Oracle WebLogic Server

1. [Setting Up the Application Authentication Method for Identity Asserter for Single Sign-On](#)
2. [Confirming mod_weblogic for Oracle Access Manager Identity Asserter](#)
3. [Clear Text Header: Establishing Trust between Oracle WebLogic Server and Other Entities](#)

16.2.4.1.1 Setting Up the Application Authentication Method for Identity Asserter for Single Sign-On This topic describes how to create the application authentication method for Oracle Access Manager Identity Assertion.

See Also: *Oracle Fusion Middleware Deploying Applications to Oracle WebLogic Server*

When you use the Oracle Access Manager Identity Asserter, all `web.xml` files in the application EAR file must specify `CLIENT-CERT` in the element `auth-method` for the appropriate realm.

You can add comma separated values here when you want applications accessed directly over the WebLogic Server `host:port` to be authenticated by the container. For instance: `<auth-method>CLIENT-CERT, FORM</auth-method>`.

The `auth-method` can use `BASIC`, `FORM`, or `CLIENT-CERT` values. While these look like similar values in Oracle Access Manager, the `auth-method` specified in `web.xml` files are used by Oracle WebLogic Server (not Oracle Access Manager).

To specify authentication in `web.xml` for the Identity Asserter

1. Locate the `web.xml` file in the application EAR file:

```
my_app/WEB-INF/web.xml
```

2. Locate the `auth-method` in `login-config` and enter `CLIENT-CERT`.

```
<login-config>
  <auth-method>CLIENT-CERT</auth-method>
</login-config>
```

3. Save the file.
4. Redeploy and restart the application.
5. Repeat for each `web.xml` file in the application EAR file.
6. Proceed to "[Confirming mod_weblogic for Oracle Access Manager Identity Asserter](#)".

16.2.4.1.2 Confirming mod_weblogic for Oracle Access Manager Identity Asserter Oracle Oracle HTTP Server includes the `mod_weblogic` plug-in module (`mod_wl_ohs.so` in 11g) which is already enabled. You can perform the following procedure to confirm this or skip this procedure.

With Oracle HTTP Server 11g, the `mod_weblogic` configuration is present in `mod_wl_ohs.conf` by default, and the path of this file is included in `httpd.conf`. If the `mod_weblogic` configuration is not present then you must edit `httpd.conf`.

To configure mod_weblogic for the Oracle Access Manager Identity Asserter

1. Locate `httpd.conf`. For example:

```
ORACLE_INSTANCE/config/OHS/<ohs_name>/httpd.conf
```

2. Confirm that the following statement is in the file with appropriate values for your deployment (add or uncomment this, if needed):

```
<IfModule mod_weblogic.c>
  WebLogicHost myHost.myDomain.com
  WebLogicPort myWlsPortNumber
</IfModule>

<Location http://request-uri-pattern>
  SetHandler weblogic-handler
</Location>
```

3. Save the file.
4. Proceed as needed for your implementation:

- [Clear Text Header: Establishing Trust between Oracle WebLogic Server and Other Entities](#)
- [Configuring Providers in the WebLogic Domain](#)

16.2.4.1.3 Clear Text Header: Establishing Trust between Oracle WebLogic Server and Other Entities The Oracle WebLogic Connection Filtering mechanism must be configured for creating access control lists and for accepting requests from only the hosts where Oracle HTTP Server and the front-end Web server are running.

Note: This filter is required for security when you use Identity Assertion with the Clear Text Header mechanism. This task is optional when you use one of the other mechanisms.

A *network connection* filter is a component that controls the access to network level resources. It can be used to protect resources of individual servers, server clusters, or an entire internal network. For example, a filter can deny non-SSL connections originating outside of a corporate network. A network connection filter functions like a firewall since it can be configured to filter protocols, IP addresses, or DNS node names. It is typically used to establish trust between Oracle WebLogic Server and foreign entities.

To configure a connection filter to allow requests from only `mod_weblogic` and the host where OHS 11g is running, perform the procedure here.

Note: This chapter uses the generic name of the WebLogic Server plug-in for Apache: `mod_weblogic`. For Oracle HTTP Server 11g, the name of this plug-in is `mod_wl_ohs`; the actual binary name is `mod_wl_ohs.so`. Examples show exact syntax for implementation.

WebLogic Server provides a default connection filter: `weblogic.security.net.ConnectionFilterImpl`. This filter accepts all incoming connections and also provides static factory methods that allow the server to obtain the current connection filter. To configure this connection filter to deny access, simply enter the connection filters rules in the WebLogic Server Administration Console.

You can also use a custom connection filter by implementing the classes in the `weblogic.security.net` package. Like the default connection filter, custom connection filters are configured in the WebLogic Server Administration Console.

Connection Filter Rules: The format of filter rules differ depending on whether you are using a filter file to enter the filter rules or you enter the filter rules in the Administration Console. When entering the filter rules on the Administration Console, enter them in the following format:

```
targetAddress localAddress localPort action protocols
```

Table 16–4 provides a description of each parameter in a connection filter.

Table 16–4 Connection Filter Rules

Parameter	Description
target	Specifies one or more systems to filter
localAddress	Defines the host address of the WebLogic Server instance. (If you specify an asterisk (*), the match returns all local IP addresses.)

Table 16–4 (Cont.) Connection Filter Rules

Parameter	Description
localPort	Defines the port on which the WebLogic Server instance is listening. (If you specify an asterisk, the match returns all available ports on the server.)
action	Specifies the action to perform. This value must be allow or deny
protocols	Is the list of protocol names to match. The following protocols may be specified: http, https, t3, t3s, giop, giops, dcom, ftp, ldap. If no protocol is defined, all protocols match a rule.

The Connection Logger Enabled attribute logs successful connections and connection data in the server. This information can be used to debug problems relating to server connections.

See Also: "Configuring Security in a WebLogic Domain" in *Oracle Fusion Middleware Securing Oracle WebLogic Server*

To configure a connection filter to allow requests from Oracle HTTP Server host

1. Log in to the Oracle WebLogic Administration Console.
2. Click Domain under Domain Configurations.
3. Click the Security tab, click the Filter tab.
4. Click the Connection Logger Enabled attribute to enable the logging of accepted messages for use when debugging problems relating to server connections.
5. Specify the connection filter to be used in the domain:
 - Default Connection Filter: In the Connection Filter attribute field, specify `weblogic.security.net.ConnectionFilterImpl`.
 - Custom Connection Filter: In the Connection Filter attribute field, specify the class that implements the network connection filter, which should also be specified in the CLASSPATH for Oracle WebLogic Server.
6. Enter the appropriate syntax for the connection filter rules.
7. Click Save.
8. Restart the Oracle WebLogic Server.
9. Proceed to ["Configuring Providers in the WebLogic Domain"](#).

16.2.4.2 Configuring Providers in the WebLogic Domain

The information here applies equally to OAM 11g and OAM 10g. This topic is divided as follows:

- [About Oracle WebLogic Server Authentication and Identity Assertion Providers](#)
- [About the Oracle WebLogic Scripting Tool \(WLST\)](#)
- [Configuring Oracle WebLogic Server for a Web Application Using ADF Security, OAM SSO, and OPSS SSO](#)
- [Setting Up Providers for Oracle Access Manager 11g Identity Assertion](#)

16.2.4.2.1 About Oracle WebLogic Server Authentication and Identity Assertion Providers This topic introduces only a few types of Authentication Providers for a WebLogic security realm, if you are new to them.

Each WebLogic security realm must have one at least one Authentication Provider configured. The WebLogic Security Framework is designed to support multiple Authentication Providers (and thus multiple LoginModules) for multipart authentication. As a result, you can use multiple Authentication Providers as well as multiple types of Authentication Providers in a security realm. The Control Flag attribute determines how the LoginModule for each Authentication Provider is used in the authentication process.

Oracle WebLogic Server offers several types of Authentication and Identity Assertion providers including, among others:

- The default WebLogic Authentication Provider (Default Authenticator) allows you to manage users and groups in one place, the embedded WebLogic Server LDAP server. This Authenticator is used by the Oracle WebLogic Server to login administrative users.
- Identity Assertion uses token-based authentication; the Oracle Access Manager Identity Asserter is one example. This must be configured to use the appropriate action for the installed WebGate (either 10g or 11g).
- LDAP Authentication Providers store user and group information in an external LDAP server. They differ primarily in how they are configured by default to match typical directory schemas for their corresponding LDAP server.

Oracle WebLogic Server 10.3.1+ provides OracleInternetDirectoryAuthenticator.

When you configure multiple Authentication Providers, use the JAAS Control Flag for each provider to control how the Authentication Providers are used in the login sequence. You can choose the following the JAAS Control Flag settings, among others:

- **REQUIRED**—The Authentication Provider is always called, and the user must always pass its authentication test. Regardless of whether authentication succeeds or fails, authentication still continues down the list of providers.
- **SUFFICIENT**—The user is not required to pass the authentication test of the Authentication Provider. If authentication succeeds, no subsequent Authentication Providers are executed. If authentication fails, authentication continues down the list of providers.
- **OPTIONAL**—The user is allowed to pass or fail the authentication test of this Authentication Provider. However, if all Authentication Providers configured in a security realm have the JAAS Control Flag set to **OPTIONAL**, the user must pass the authentication test of one of the configured providers.

When additional Authentication Providers are added to an existing security realm, the Control Flag is set to **OPTIONAL** by default. You might need to change the setting of the Control Flag and the order of providers so that each Authentication Provider works properly in the authentication sequence.

See Also: "Configuring Authentication Providers" in *Oracle Fusion Middleware Securing Oracle WebLogic Server* for a complete list of Authentication Providers and details about configuring the Oracle Internet Directory provider to match the LDAP schema for user and group attributes

16.2.4.2.2 About the Oracle WebLogic Scripting Tool (WLST) This topic introduces WLST, if you are new to it.

You can add providers to a WebLogic domain using either the Oracle WebLogic Administration Console or Oracle WebLogic Scripting Tool (WLST) command-line tool.

WLST is a Jython-based command-line scripting environment that you can use to manage and monitor WebLogic Server domains. Generally, you can use this tool online or offline. You can use this tool interactively on the command line in batches supplied in a file (Script Mode, where scripts invoke a sequence of WLST commands without requiring your input), or embedded in Java code.

When adding Authentication Providers to a WebLogic domain, you can use WLST online to interact with an Authentication Provider and add, remove, or modify users, groups, and roles.

When you use WLST offline to create a domain template, WLST packages the Authentication Provider's data store along with the rest of the domain documents. If you create a domain from the domain template, the new domain has an exact copy of the Authentication Provider's data store from the domain template. However, you cannot use WLST offline to modify the data in an Authentication Provider's data store.

Note: You cannot use WLST offline to modify the data in an Authentication Provider's data store.

See Also:

- ["Configuring Oracle WebLogic Server for a Web Application Using ADF Security, OAM SSO, and OPSS SSO"](#)
- *Oracle Fusion Middleware Oracle WebLogic Scripting Tool*
- *Oracle Fusion Middleware WebLogic Scripting Tool Command Reference* "Infrastructure Security Commands" chapter

16.2.4.2.3 Configuring Oracle WebLogic Server for a Web Application Using ADF Security, OAM SSO, and OPSS SSO

On the Oracle WebLogic Server, you can run a Web application that uses Oracles Application Development Framework (Oracle ADF) security, integrates with Oracle Access Manager Single Sign On (SSO), and uses Oracle Platform Security Services (OPSS) SSO for user authentication. However before the Web application can be run, you must configure the domain-level `jps-config.xml` file on the application's target Oracle WebLogic Server for the Oracle Access Manager security provider.

The domain-level `jps-config.xml` file is in the following path and should not be confused with the deployed application's `jps-config.xml` file:

```
domain_home/config/fmwconfig/jps-config.xml
```

You can use an Oracle Access Manager-specific WLST script to configure the domain-level `jps-config.xml` file, either before or after the Web application is deployed. This Oracle JRF WLST script is named as follows:

Linux: `wlst.sh`

Windows: `wlst.cmd`

The Oracle JRF WLST script is available in the following path if you are running through JDev:

```
$JDEV_HOME/oracle_common/common/bin/
```

In a standalone JRF WebLogic installation, the path is:

```
$Middleware_home/oracle_common/wlst
```

Note: The Oracle JRF WLST script is required. When running WLST for Oracle Java Required Files (JRF), do not use the WLST script under `$JDEV_HOME/wlserver_10.3/common/bin`.

Command Syntax

```
addOAMSSOProvider(loginuri, logouturi, autologinuri)
```

[Table 16–5](#) defines the expected value for each argument in the `addOAMSSOProvider` command line.

Table 16–5 *addOAMSSOProvider Command-line Arguments*

Argument	Definition
loginuri	Specifies the URI of the login page
autologinuri	Specifies the URI of the autologin page.
logouturi	Specifies the URI of the logout page

See Also:

- [Oracle Fusion Middleware Oracle WebLogic Scripting Tool](#)
- [Oracle Fusion Middleware WebLogic Scripting Tool Command Reference "Infrastructure Security Commands" chapter](#)

Prerequisites

[Configuring Providers in the WebLogic Domain](#)

To modify domain-level `jps-config.xml` for a Fusion Web application with Oracle ADF Security enabled

1. On the computer hosting the Oracle WebLogic Server and the Web application using Oracle ADF security, locate the Oracle JRF WLST script. For example:

```
cd $ORACLE_HOME/oracle_common/common/bin
```

2. Connect to the computer hosting the Oracle WebLogic Server:

```
connect login_id password hostname:port
```

For example, the Oracle WebLogic Administration Server host could be `localhost` using port `7001`. However, your environment might be different.

3. Enter the following command-line arguments with values for the application with ADF security enabled:

```
addOAMSSOProvider(loginuri="/${app.context}/adfAuthentication",
logouturi="/oamssso/logout.html", autologinuri="/obrar.cgi")
```

4. Stop and start the Oracle WebLogic Server.
5. Perform the following tasks as described in:
 - [Setting Up Providers for Oracle Access Manager 11g Identity Assertion](#)
 - [Configuring Centralized Log Out for Oracle Access Manager 11g](#)
 - [Testing Oracle Access Manager Identity Assertion for Single Sign-on](#)

16.2.4.2.4 Setting Up Providers for Oracle Access Manager 11g Identity Assertion This topic describes how to configure providers in the WebLogic security domain to perform single sign-on with the Oracle Access Manager Identity Asserter. Several Authentication Provider types must be configured and ordered:

- OAM Identity Asserter: REQUIRED (also specify a chosen Active Type for the mechanism you are using (Table 15-1))
- OID Authenticator: SUFFICIENT
- DefaultAuthenticator: SUFFICIENT

See Also: "About Oracle WebLogic Server Authentication and Identity Assertion Providers" on page 16-19

The following procedure uses the WebLogic Administration Console.

Note: With an Oracle Fusion Middleware application installed, you have the required provider JAR file. Skip Step 1.

To set up Providers for Oracle Access Manager single sign-on in a WebLogic domain

1. **No Oracle Fusion Middleware Application:** Obtain the Oracle Access Manager provider:
 - a. Log in to Oracle Technology Network at:

http://www.oracle.com/technology/software/products/middleware/docs/111110_fmws.html
 - b. Locate the oamAuthnProvider ZIP file with Access Manager WebGates (10.1.4.3.0):

`oamAuthnProvider<version number>.zip`
 - c. Extract and copy oamAuthnProvider.jar to the following path on the computer hosting Oracle WebLogic Server:

`BEA_HOME/wlserver_10.x/server/lib/mbeantypes/oamAuthnProvider.jar`
2. **With Oracle Fusion Middleware Application Installed:**
 - a. Locate oamauthenticationprovider.war in the following path:

`ORACLE_INSTANCE/modules/oracle.oamprovider_11.1.1/oamauthenticationprovider.war`
 - b. Copy oamauthenticationprovider.war to the following location:

`BEA_HOME/wlserver_10.x/server/lib/console-ext/autodeploy/oamauthenticationprovider.war`
3. Log in to the WebLogic Administration Console.
4. Click **Security Realms**, *Default Realm Name*, and click **Providers**.
5. **OAM Identity Asserter:** Perform the following steps to add this provider:
 - a. Click **New**, and then enter a name and select a type:

Name: *OAM Identity Asserter*

Type: **OAMIdentityAsserter**

OK

- b. In the Authentication Providers table, click the newly added authenticator.
- c. Click the **Common** tab, set the Control Flag to **REQUIRED**.
- d. On the **Common** tab, specify one Chosen Active Type for your SSO mechanism (Table 15-1). For example:

OAM_IDENTITY_ASSERTION

- e. Save the configuration.
6. **OID Authenticator:** Perform the following steps to add this provider.

- a. Click **Security Realms, Default Realm Name**, and click **Providers**.
- b. Click **New**, enter a name, and select a type:

Name: *OID Authenticator*

Type: *OracleInternetDirectoryAuthenticator*

OK

- c. In the Authentication Providers table, click the newly added authenticator.
- d. On the Settings page, click the **Common** tab, set the Control Flag to **SUFFICIENT**, and then click **Save**.
- e. Click the **Provider Specific** tab and specify the following required settings using values for your own environment:

Host: Your LDAP host. For example: *localhost*

Port: Your LDAP host listening port. For example: *6050*

Principal: LDAP administrative user. For example: *cn=orcladmin*

Credential: LDAP administrative user password.

User Base DN: Same searchbase as in Oracle Access Manager.

All Users Filter: For example: *(&(uid=*)(objectclass=person))*

User Name Attribute: Set as the default attribute for username in the LDAP directory. For example: *uid*

Group Base DN: The group searchbase (same as User Base DN)

Do not set the All Groups filter as the default works fine as is.

Save.

- 7. **Default Authenticator:** Perform the following steps to set up the Default Authenticator for use with the Identity Asserter:
 - a. Go to **Security Realms, Default Realm Name**, and click **Providers**.
 - b. Click **Authentication**, Click **DefaultAuthenticator** to see its configuration page.
 - c. Click the **Common** tab and set the Control Flag to **SUFFICIENT**.
 - d. Save.
- 8. Reorder Providers:
 - a. Click **Security Realms, Default Realm Name, Providers**.

- b. On the Summary page where providers are listed, click the **Reorder** button
 - c. On the **Reorder Authentication Providers** page, select a provider name and use the arrows beside the list to order the providers as follows:
 - OAM Identity Asserter (REQUIRED)
 - OID Authenticator (SUFFICIENT)
 - Default Authenticator (SUFFICIENT)
 - d. Click OK to save your changes
9. **Activate** Changes: In the Change Center, click Activate Changes.
 10. Reboot Oracle WebLogic Server.
 11. Proceed as follows:
 - **Successful:** Proceed as needed or your implementation.
 - [Trusted Header Assertion: Configuring Digital Signature Verification](#)
 - [Testing Oracle Access Manager Identity Assertion for Single Sign-on](#)
 - **Not Successful:** Confirm that all providers have the proper specifications for your environment, are in the proper order, and that `oamAuthnProvider.jar` is in the correct location.

As mentioned earlier, a login form shipped with 10g WebGate is used only with OAM 10g Access Server. For OAM 11g, neither the 10g WebGate nor 11g WebGate provide a login page.

Note: The OAM 11g Server displays a login page. No set up is needed.

16.2.4.3 Trusted Header Assertion: Configuring Digital Signature Verification

This is a manual task. The Oracle Access Manager certificate public key is required for digital signature verification. The certificate, which is consumed by the Identity Asserter, must be in the `.oamkeystore`.

For the SSO Sync Filter to consume the certificate, you need to provide the truststore to the filter. SSO Sync Filter behavior can be altered for application requirements by passing various over-riding system properties to WebLogic. To do this, you add a property in Oracle WebLogic startup script (`setDomainEnv.sh`) under `EXTRA_JAVA_PROPERTIES`. The truststore location can be provided as the system property. By default filter will look for keystore at `ssofilter.jar` location. If not found then it looks in system property.

The following procedure guides as you retrieve the `.oamkeystore` password required to perform export and import operations. After you export and import the required OAM certificate, you provision the keystore to enable the Identity Asserter to consume the certificate. Finally, you choose the `OAM_IDENTITY_ASSERTION` token type, provision the certificate in the SSO Sync Filter, and confirm that the authorization policy enables Identity Assertion.

To configure digital signature verification for trusted header assertion

1. Retrieve the `.oamkeystore` password using WLST script tool as follows:
 - a. Locate the WLST tool in your `$MW_HOME/Oracle_IDM1/common/bin`.

b. Execute `wst.sh: $./wst.sh`.

c. Confirm execution with the following onscreen messages:

```
Initializing WebLogic Scripting Tool (WLST) ...
```

```
Jython scans all the jar files it can find at first startup. Depending on
the system, this process may take a few minutes to complete, and WLST may
not return a prompt right away
```

```
Welcome to WebLogic Server Administration Scripting Shell
```

```
Type help() for help on available commands
```

d. Execute `wls:/offline> connect()` and supply information for your environment (WebLogic Administrator username and password and AdminServer URL). For example:

```
Please enter your username: weblogic
Please enter your password: password
Please enter your server URL //localhost:7001
not return a prompt right away
```

```
Connecting to ...
Successfully connected to Admin Server 'AdminServer' ... domain 'base_
domain'.
```

```
Warning: An insecure protocol was used to connect to the server.
To ensure on-the-wire security, the SSL port or Admin port should be used
instead.
```

```
wls:/base_domain/serverConfig
```

e. Execute `wls:/base_domain/serverConfig> domainRuntime()` and check the following onscreen messages. For example:

```
Location changed to domainRuntime tree. This is a read-only tree with
DomainMBean as the root
```

```
For more help, use help(domainRuntime)
```

```
wls:/base_domain/domainRuntime>
```

f. Execute `wls:/base_domain/domainRuntime> listCred(map="OAM_STORE",key="jks")`. For example:

```
Already in Domain Runtime Tree
PASSWORD: 11eoi4sbkpo3bj8fg55k7jgbgh
wls:/base_domain/domainRuntime>
```

2. Export the alias to a certificate file using the JDK6 keytool, as follows:

```
jdk/bin]$ keytool -exportcert -alias "assertion-cert" -keystore .oamkeystore
-storepass gtml6es9qderjc66f76hvtqm5a -storetype JCEKS -file assertion.cer
```

3. Import the certificate file using the JDK6 keytool, as follows:

Note: The keystore alias `oam.assertion.cert` and the keystore name `oamiap-keystore.jks` are fixed. Use those names only.

```
jdk/bin $ keytool -importcert -trustcacerts -alias "oam.assertion.cert" -file
```

```
assertion.cer -keystore /scratch/oamiap-keystore.jks -storepass password
-storetype JKS
```

4. Provision the Identity Asserter keystore for consumption of the OAM certificate with the public key in oamiap-keystore.jks:
 - a. From the WebLogic Console, Security Realm, Identity Asserter entry, add the absolute path of oamiap-keystore.jks in the provider-specific configuration.
 - b. Select token type OAM_IDENTITY_ASSERTION in provider-specific configuration.
 - c. Save this configuration.

5. Provision .oamkeystore in the SSO Sync Filter, as follows:

By default, the filter looks for the keystore in the ssofilter.jar location. If not found there, the system property is checked.

- a. **Default configuration:** Place the keystore file oamiap-keystore.jks in the same location as ssofilter.jar. For example: `$MW_HOME/oracle_common/modules/oracle.ssofilter_11.1.1`
- b. **Fallback Mechanism:** Set the keystore file oamiap-keystore.jks as a systemproperty in setDomainEnv.sh (`$MW_HOME/user_projects/domains/base_domain/bin/setDomainEnv.sh`):


```
-Dsso.filter.oam.keystore=/scratch/keystore/oamiap-keystore.jks
```

6. Set System Properties for OAM_IDENTITY_ASSERTION, as follows:

- a. Stop the WebLogic Server.
- b. Open the file setDomainEnv.sh in `$MW_HOME/user_projects/domains/base_domain/bin/setDomainEnv.sh`

- c. Add the following property under EXTRA_JAVA_PROPERTIES, and save the file:

```
-Dssso.filter.sstoken=OAM_IDENTITY_ASSERTION
```

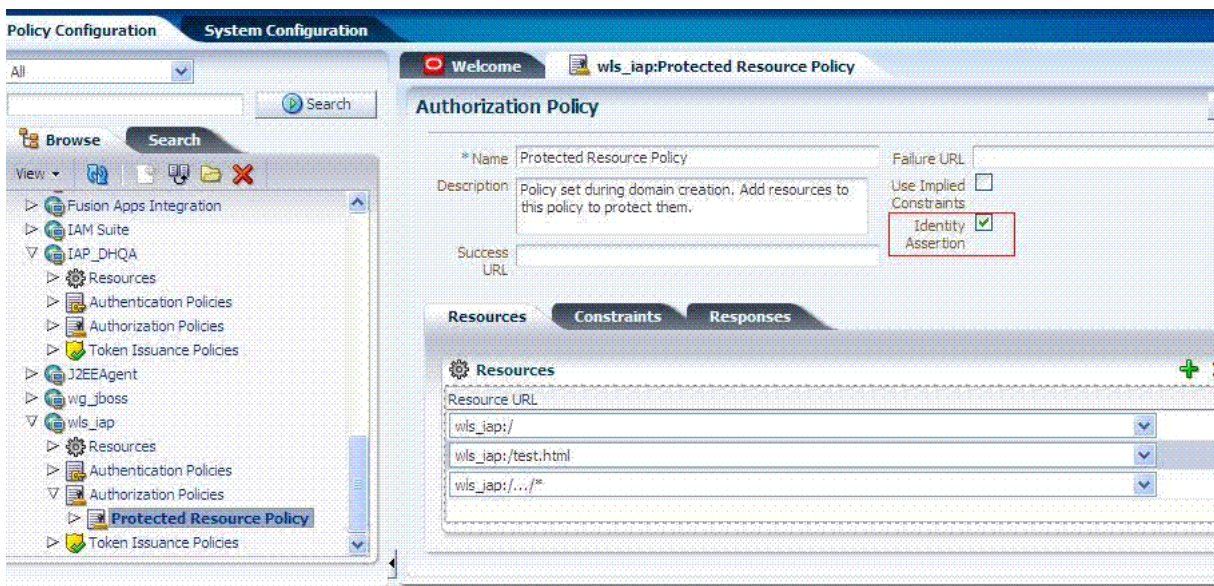
- 7. Start the WebLogic Server.
- 8. Proceed to "Trusted Header Assertion: Configuring Policies".

16.2.4.4 Trusted Header Assertion: Configuring Policies

To use OAM_IDENTITY_ASSERTION as a token type for the assertion, the Identity Assertion option must be enabled within the authorization policy that protects the resources. Default policies are generated during agent registration. You can also create policies manually using the Oracle Access Manager Console.

Figure 16–6 provides an example of an authorization policy for the Trusted Header Assertion mechanism.

Figure 16–6 Sample Authorization Policy for Trusted Header Assertion



The following procedure provides the steps to enable Identity Assertion within the Oracle Access Manager 11g authorization policy that protects the resources.

See Also: Oracle Fusion Middleware Administrator's Guide for Oracle Access Manager with Oracle Security Token Service

To enable Identity Assertion for Trusted Header Assertion

1. From the Policy Configuration tab, navigation tree, open the following nodes:
 - Application Domains
 - Desired Domain*
 - Authorization Policies
 - PolicyName*
2. Enable Identity Assertion (check the box).
3. **Resources:**

- On the Resource tab, confirm the desired resources are protected by this policy.
 - Add or remove resources as needed.
4. Click Apply to save changes and close the Confirmation window.
 5. Close the page when you finish.
 6. Proceed with "[Testing Oracle Access Manager Identity Assertion for Single Sign-on](#)".

16.2.4.5 Testing Oracle Access Manager Identity Assertion for Single Sign-on

The following procedure describes how to test your Oracle Access Manager Identity Assertion setup, regardless of the mechanism you are using.

Alternatively, you can run Access Tester in Oracle Access Manager to test your policy domain, as described in the *Oracle Fusion Middleware Administrator's Guide for Oracle Access Manager with Oracle Security Token Service*.

To validate Oracle Access Manager Identity Assertion for Single Sign-on

1. Enter the URL to access the protected resource in your environment. For example:

```
http://ohs_server:port/<protected url>
```
2. Provide appropriate credentials when the login form appears.
 - Successful: The implementation works.
 - Not Successful: See "[Troubleshooting Tips](#)" on page 16-42.

16.2.5 Configuring the Authenticator Function for Oracle Access Manager 11g

With the Authenticator function, the user is challenged for credentials based on the authentication method that is configured within the application web.xml. However, an Oracle Access Manager authentication scheme is required and available in the pre-seeded application domain that is delivered with Oracle Access Manager 11g. It protects the following resources (resource type wl_authen):

- /Authen/Basic
- /Authen/SSOToken
- /Authen/UsernameAssertion

You can add Responses and Constraints to policies. However, no other configuration is needed.

For more information about the pre-seeded application domain, see "[Previewing Pre-Seeded OAM 11g Policies for Use by the 10g AccessGate](#)" on page 16-4.

Prerequisites

- [Installing the Authentication Provider with Oracle Access Manager 11g](#)
- [Session Token: Provisioning an OAM Agent with Oracle Access Manager 11g](#)

Note: You can provision the custom 10g AccessGate for the Authenticator or simply refer to an existing OAM Agent registration when configuring providers for the Authenticator.

Tasks to configure the Oracle Access Manager Authenticator are described in the following overview.

Task overview: Configuring the Authenticator function for OAM includes

1. Ensuring that all prerequisite tasks have been performed
2. [Configuring Providers for the Authenticator in a WebLogic Domain](#)
3. [Configuring the Application Authentication Method for the Authenticator](#)
4. [Mapping the Authenticated User to a Group in LDAP](#)
5. [Configuring Centralized Log Out for Oracle Access Manager 11g](#)
6. [Testing the Oracle Access Manager Authenticator Implementation](#)

16.2.5.1 Configuring Providers for the Authenticator in a WebLogic Domain

This topic includes a procedure that you can use to add and configure the appropriate Authentication providers in a WebLogic domain.

The Oracle Access Manager Authenticator must be configured along with the Default Authentication Provider in a WebLogic domain.

- DefaultAuthenticator: SUFFICIENT
- OAM Authenticator: OPTIONAL

The following procedure describes this task using the WebLogic Administration Console. You can also add these using the Oracle WebLogic Scripting Tool (WLST).

See Also:

- ["About Oracle WebLogic Server Authentication and Identity Assertion Providers"](#) on page 16-19
- *Oracle Fusion Middleware Oracle WebLogic Scripting Tool*
- *Oracle Fusion Middleware WebLogic Scripting Tool Command Reference*

Note: When an Oracle Fusion Middleware application is installed, you have the required files and can skip Step 1.

To configure providers for the Oracle Access Manager Authenticator in a WebLogic domain

1. **No Oracle Fusion Middleware Application:** Obtain the Oracle Access Manager provider if you have no Oracle Fusion Middleware application.

- a. Log in to Oracle Technology Network at:

http://www.oracle.com/technology/software/products/middleware/htdocs/111110_fmws.html

- b. Locate the oamAuthnProvider ZIP file with Access Manager WebGates (10.1.4.3.0). For example:

oamAuthnProvider<version>.zip

- c. Extract and copy the oamAuthnProvider.jar to the following path on the computer hosting Oracle WebLogic Server:

BEA_HOME/wlserver_10.x/server/lib/mbeantypes/oamAuthnProvider.jar

2. Go to the Oracle WebLogic Administration Console.
3. **With Oracle Fusion Middleware Application Installed:**
 - a. Locate oamauthenticationprovider.war in the following path:

ORACLE_INSTANCE/modules/oracle.oamprovider_11.1.1/oamauthenticationprovider.war
 - b. Copy oamauthenticationprovider.war to the following location:

BEA_HOME/wlserver_10.x/server/lib/console-ext/autodeploy/oamauthenticationprovider.war
4. Go to the Oracle WebLogic Administration Console.
5. Click **Lock & Edit**, if desired.
6. **OAM Authenticator:**
 - a. Click **Security Realms** and select the realm you want to configure.
 - b. Select **Providers, Authentication**, and click **New** to display the Create a New Authentication Provider page
 - c. Enter a name and select a type:

Name *OAMAuthN*

Type: **OAMAuthenticator**

OK
 - d. Click the name of the Authentication provider you have just created to display the Provider Configuration page.
 - e. In the Provider Configuration page, set the required values as follows:

Access Gate Name: The name of the AccessGate used by the Provider. This must match exactly the name of an OAM Agent registration in the Oracle Access Manager Console.

Note: You can have one or more 10g OAM Agents registered with OAM 11g. Be sure to choose the correct Agent registration name.

Access Gate Password: The same password, if any, that is as defined for the Agent registration (see the Oracle Access Manager Console).

Primary Access Server: The *host:port* of the primary OAM Server that is associated with this AccessGate in the Oracle Access Manager Console.

Advanced Configuration: Following are several advanced configuration values.

Transport Security: The communication mode between OAM Server and AccessGate: open, simple, or cert.

If transport security is Simple or Cert, include the following parameters and values:

Trust Store: The absolute path of JKS trust store used for SSL communication between the provider and the OAM Server.

Key Store: The absolute path of JKS key store used for SSL communication between the provider and the OAM Server.

Key Store Pass Phrase: The password to access the key store.

Simple mode pass phrase: The password shared by AccessGate and OAM Server for simple communication modes.

Secondary OAM Server: The *host:port* of the secondary OAM Server that is associated with this AccessGate in the Oracle Access Manager Console.

Maximum OAM Server Connections in Pool: The maximum number of connections that the AccessGate opens to the OAM Server. The default value is 10.

Note: The Maximum OAM Server Connections in Pool (or Minimum OAM Server Connections in Pool) settings in the WebLogic Administration Console are different from the Maximum (or Minimum) Connections specified in the Oracle Access Manager Console.

Minimum Access Server Connections in Pool: The minimum number of connections that the Authentication provider uses to send authentication requests to the OAM Server. The default value is 5.

See Also: "[Oracle Access Manager Authentication Provider Parameter List](#)" on page 17-14 for descriptions and values of the common and provider-specific parameters

- f. Ensure that the parameter **Control Flag** is set to OPTIONAL initially.

Note: Do not set the parameter **Control Flag** to REQUIRED until you have verified that the Authentication Provided is operational and configured correctly.

7. In the Change Center, click **Activate Changes**.
8. **DefaultAuthenticator:** Under the Providers tab, select **DefaultAuthenticator**, which changes its control flag to SUFFICIENT.
9. **Reorder:** Under the Providers tab, reorder the providers so that DefaultAuthenticator is first (**OAMAuthenticator** follows **DefaultAuthenticator**).

Note: If the Oracle Access Manager Authenticator flag is set to REQUIRED, or if Oracle Access Manager Authenticator is the only Authentication provider, perform the next step to ensure that the LDAP user who boots Oracle WebLogic Server is included in the administrator group that can perform this task. By default the Oracle WebLogic Server Admin Role includes the Administrators group.

10. **Oracle Access Manager Authenticator REQUIRED or the Only Authenticator:** Perform the following steps to set user rights for booting Oracle WebLogic Server.
 - a. Create an Administrators group in the directory server, if one does not already exist (or any other group for which you want boot access).

Note: To provide access to any other group, you must create that group in the directory server and add the user who boots WebLogic Server in that group.

- b. Confirm that the LDAP user who boots Oracle WebLogic Server is included in the Administrators (or other) group.
 - c. From the WebLogic Administration Console, go to **Security Realms**, *myrealm*, Roles and Policies, Global Roles.
 - d. Select **View Conditions** for the Admin Role.
 - e. Add the group and click Save.
11. Reboot the WebLogic Server.
 12. Once the server has started, reset the Authentication Provider parameter **Control Flag** to the appropriate value (REQUIRED, OPTIONAL, or SUFFICIENT).

Note: The recommended value is REQUIRED. To prevent a known issue, see "JAAS Control Flag" on page 17-74.

13. Proceed with "[Configuring the Application Authentication Method for the Authenticator](#)".

16.2.5.2 Configuring the Application Authentication Method for the Authenticator

This topic describes how to create the application authentication method for Oracle Access Manager Authenticator.

See Also: *Oracle Fusion Middleware Deploying Applications to Oracle WebLogic Server*

When you use the Oracle Access Manager Authenticator, all `web.xml` files in the application EAR file must specify BASIC in the element `auth-method` for the appropriate realm.

The `auth-method` can use BASIC or FORM values. While these look like similar values in Oracle Access Manager, the `auth-method` specified in `web.xml` files are used by Oracle WebLogic Server (not Oracle Access Manager).

Note: For the Oracle Access Manager Authenticator, Oracle recommends `auth-method BASIC` in `login-config` within `web.xml`.

To configure the application authentication method for the Authenticator

1. Locate the `web.xml` file in the application EAR file:

```
WEB-INF/web.xml
```

2. Locate the `auth-method` in `login-config` and enter BASIC. For example:

```
<security-constraint>
<web-resource-collection>
<web-resource-name>protected</web-resource-name>
<url-pattern>/servlet</url-pattern>
</web-resource-collection>
```

```

<auth-constraint>
<role-name>auth-users</role-name>
</auth-constraint>
</security-constraint>
<login-config>
<auth-method>BASIC</auth-method>
</login-config>
<security-role>
<description>Authenticated Users</description>
<role-name>auth-users</role-name>
</security-role>

```

3. Save the file.
4. Redeploy and restart the application.
5. Repeat for each web.xml file in the application EAR file.
6. Proceed with ["Mapping the Authenticated User to a Group in LDAP"](#).

16.2.5.3 Mapping the Authenticated User to a Group in LDAP

This topic describes how to map the authenticated user to a group in LDAP. To do this, you must edit the weblogic.xml file. For example, you might need to map your role-name *auth-users* to a group named *managers* in LDAP.

To map the authenticated user to a group in LDAP for the Oracle Access Manager Authenticator

1. Go to the application's weblogic.xml file.
2. Add the following information for your environment anywhere in the file:

```

<weblogic-web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.bea.com/ns/weblogic/weblogic-web-app
http://www.bea.com/ns/weblogic/weblogic-web-app/1.0/weblogic-web-app.xsd"
xmlns="http://www.bea.com/ns/weblogic/weblogic-web-app">
<security-role-assignment>
<principal-name>managers</principal-name>
<role-name>auth-users</role-name>
</security-role-assignment>
</weblogic-web-app>

```

3. Save the file.
4. Restart the WebLogic Server.
5. Configure centralized logout as described in ["Configuring Centralized Log Out for Oracle Access Manager 11g"](#) and then return here to perform ["Testing the Oracle Access Manager Authenticator Implementation"](#).

16.2.5.4 Testing the Oracle Access Manager Authenticator Implementation

After performing all tasks to implement the Authenticator, you can test it by attempting to log in to the application using valid credentials. If the configuration is incorrect, a valid user is denied access.

The following procedure describes how to test your Authenticator setup. Alternatively, you can run Access Tester in Oracle Access Manager to test your policy domain, as described in the *Oracle Fusion Middleware Administrator's Guide for Oracle Access Manager with Oracle Security Token Service*.

To validate the Oracle Access Manager Authenticator implementation

1. Enter the URL to access the protected resource in your environment. For example:
`http://yourdomain.com:port`
2. Provide appropriate credentials when the login form appears.
 - Successful: The implementation works.
 - Not Successful: See "[Troubleshooting Tips](#)" on page 16-42.

16.2.6 Configuring Identity Assertion for Oracle Web Services Manager and OAM 11g

This section describes how to set up the Oracle Access Manager Identity Asserter to enable validation of the token when you have Oracle Web Services Manager protecting Web services.

As discussed earlier, the Oracle Access Manager Identity Asserter works in two modes. The default mode of operation simply asserts the header that is set by WebGate at the perimeter, which handles most SSO situations. The alternate mode uses the custom AccessGate in `oamAuthnProvider.jar`. In this case, and with the absence of the header, the Identity Asserter contacts the OAM Server to validate the token. For more information about the token, see "[Installing the Authentication Provider with Oracle Access Manager 11g](#)" on page 16-8.

Note: The 10g custom AccessGate provided with the Authentication Provider is required for Identity Assertion for Oracle Web Services Manager.

With OAM 10g, you would need to manually create the policy domain and policies for this configuration. However, with OAM 11g, a pre-seeded application domain is delivered with policies that protect the following resources (resource type `wl_authen`):

- `/Authen/Basic`
- `/Authen/SSOToken`
- `/Authen/UsernameAssertion`

You can add policies, Responses, or Constraints for resources of type `wl_authen` only. Ideally, however, you can use this application domain with no further configuration. For more information, see "[Previewing Pre-Seeded OAM 11g Policies for Use by the 10g AccessGate](#)" on page 16-4.

When the Oracle Access Manager Identity Asserter is configured for both header and token validation modes, preference is given to the presence of the header. If the header is not present, the Identity Asserter contacts the OAM Server to validate the token. For more information on the token, see "[Oracle Access Manager Authentication Provider Parameter List](#)" on page 17-14.

Prerequisites

[Installing the Authentication Provider with Oracle Access Manager 11g](#)

[Session Token: Provisioning an OAM Agent with Oracle Access Manager 11g](#)

Task overview: Deploying the Identity Asserter with Oracle Web Services Manager includes

1. [Configuring Providers in a WebLogic Domain for Oracle Web Services Manager](#)

2. [Configuring Centralized Log Out for Oracle Access Manager 11g](#)
3. [Testing the Identity Asserter with Oracle Web Services Manager](#)

16.2.6.1 Configuring Providers in a WebLogic Domain for Oracle Web Services Manager

To use Oracle Access Manager Identity Asserter with Oracle Web Services Manager protected Web services, several Authentication providers must be configured and ordered in a WebLogic domain:

- OAM Identity Asserter: REQUIRED
- OID Authenticator: SUFFICIENT
- DefaultAuthenticator: SUFFICIENT

This procedure is nearly identical to the one for the Oracle Access Manager Identity Asserter with OAM 11g. The difference in this case is that Oracle Web Services Manager requires the custom 10g AccessGate and additional provider-specific values:

- Primary Access Server: Specify the primary OAM Server host and port. For example: *mnop:8888*
- Access Gate Name: The name of the AccessGate registration protecting the application. For example: *AG1*
- Access Gate Password: The AccessGate password as specified in the Oracle Access Manager Console.

You can add these using either the Oracle WebLogic Administration Console or Oracle WebLogic Scripting Tool (WLST) command-line tool.

See Also:

- ["About Oracle WebLogic Server Authentication and Identity Assertion Providers"](#) on page 16-19
- *Oracle Fusion Middleware Oracle WebLogic Scripting Tool*
- *Oracle Fusion Middleware WebLogic Scripting Tool Command Reference*

Note: With a Oracle Fusion Middleware application installed, you have the required provider file. Skip Step 1.

To set up providers in a WebLogic domain

1. **No Oracle Fusion Middleware Application:** Obtain the Oracle Access Manager provider if you have no Oracle Fusion Middleware application.
 - a. Log in to Oracle Technology Network at:

http://www.oracle.com/technology/software/products/middleware/html/docs/111110_fm.html
 - b. Locate the oamAuthnProvider ZIP file with Access Manager WebGates (10.1.4.3.0). For example:


```
oamAuthnProvider<version>.zip
```
 - c. Extract and copy the oamAuthnProvider.jar to the following path on the computer hosting Oracle WebLogic Server:

BEA_HOME/wlserver_10.x/server/lib/mbeantypes/oamAuthnProvider.jar

2. Log in to the Oracle WebLogic Administration Console.
3. **OAM Identity Asserter:** Perform the following steps to add this provider:
 - a. Click **Security Realms**, *Default Realm Name*, and click **Providers**.
 - b. Click **Authentication**, click **New**, and then enter a name and select a type:
 Name: *OAM Identity Asserter*
 Type: **OAMIdentityAsserter**
 OK
 - c. In the **Authentication Providers** table, click the newly added authenticator.
 - d. On the **Common** tab, set the **Control Flag** to **REQUIRED**, and click **Save**.
 - e. Click the **Common** tab, specify **ObSSOCookie** as the chosen **Active Type** for the 10g custom **AccessGate**, and click **Save**.
 - f. Click the **Provider Specific** tab and configure these parameters:
 Primary Access Server: Specify the primary OAM Server host and port. For example: *abcd:7777*
 Access Gate Name: The name of the OAM Agent registration protecting the application. For example: *AG1*
 Access Gate Password: The AccessGate password, if any, that was specified in during provisioning.
 Save.
4. **OID Authenticator:** Perform the following steps to add this provider.
 - a. Click **Security Realms**, *Default Realm Name*, and click **Providers**
 - b. Click **New**, enter a name, and select a type:
 Name: *OID Authenticator*
 Type: **OracleInternetDirectoryAuthenticator**
 Click **OK**.
 - c. In the **Authentication Providers** table, click the newly added authenticator.
 - d. On the **Settings** page, click the **Common** tab, set the **Control Flag** to **SUFFICIENT**, and then click **Save**.
 - e. Click the **Provider Specific** tab and specify the following required settings using values for your own environment:
 Host: Your LDAP host. For example: *localhost*
 Port: Your LDAP host listening port. For example: *6050*
 Principal: LDAP administrative user. For example: *cn=orcladmin*
 Credential: LDAP administrative user password.
 User Base DN: Same searchbase as in Oracle Access Manager.
 All Users Filter: For example: *(&(uid=*)(objectclass=person))*
 User Name Attribute: Set as the default attribute for username in the LDAP directory. For example: *uid*

Group Base DN: The group searchbase (same as User Base DN)

Note: Do not set the All Groups filter as the default works fine as is.

Click Save.

5. **Default Authenticator:** Perform the following steps to set up the Default Authenticator for use with the Identity Asserter:
 - a. Go to **Security Realms**, *Default Realm Name*, and click **Providers**.
 - b. Click Authentication, Click **DefaultAuthenticator** to see its configuration page.
 - c. Click the Common tab and set the Control Flag to **SUFFICIENT**.
 - d. Click Save.
6. Reorder Providers:
 - a. Click **Security Realms**, *Default Realm Name*, **Providers**.
 - b. On the Summary page where providers are listed, click the **Reorder** button
 - c. On the **Reorder Authentication Providers** page, select a provider name and use the arrows beside the list to order the providers as follows:

OAM Identity Asserter (REQUIRED)
OID Authenticator (SUFFICIENT)
Default Authenticator (SUFFICIENT)
 - d. Click OK to save your changes
7. **Activate** Changes: In the Change Center, click Activate Changes
8. Reboot Oracle WebLogic Server.
9. Proceed as follows:
 - Successful: Go to "[Configuring Centralized Log Out for Oracle Access Manager 11g](#)", and then return here to perform "[Testing the Identity Asserter with Oracle Web Services Manager](#)".
 - Not Successful: Confirm the all providers have the proper specifications for your environment, are in the proper order, and that `oamAuthnProvider.jar` is in the correct location as described in "[Installing the Authentication Provider with Oracle Access Manager 11g](#)" on page 16-8.

16.2.6.2 Testing the Identity Asserter with Oracle Web Services Manager

To validate the use of the Oracle Access Manager Identity Asserter with Oracle Web Services Manager, you can access the Web service protected by the Identity Asserter and Oracle Web Services Manager policies. If access is granted, the implementation works. If not, see "[Troubleshooting Tips](#)" on page 16-42.

16.3 Configuring Centralized Log Out for Oracle Access Manager 11g

This section introduces Centralized logout for Oracle Access Manager 11g.

With OAM 11g, centralized logout refers to the process of terminating an active user session. Guidelines include:

- Applications must not provide their own logout page for use in an SSO environment.
- Applications must make their logout links configurable with a value that points to the logout URL specified by the OAM WebGate Administrator.

Note: Oracle strongly recommends that applications use the ADF Authentication servlet, which in turn interfaces with OPSS, where a domain wide configuration parameter can be used to specify the logout URL. This way applications need not be modified or redeployed to change logout configuration.

For more information, see:

- [Logout for 11g WebGate and OAM 11g](#)
- [Logout for 10g WebGate with Oracle Access Manager 11g](#)

16.3.1 Logout for 11g WebGate and OAM 11g

Several elements in the OAM 11g Agent registration page enable centralized logout for OAM 11g WebGates. After agent registration, the ObAccessClient.xml file is populated with the information.

11g WebGate logout options that you must have in the agent registration include the following:

- Logout URL: Triggers the logout handler, which removes the cookie (ObSSOCookie for 10g WebGates; OAMAuthnCookie for 11g WebGates) and requires the user to re-authenticate the next time he accesses a resource protected by Oracle Access Manager.
- Logout Callback URL: The URL to oam_logout_success, which clears cookies during the call back. This can be a URI format without host:port (recommended), where the OAM Server calls back on the host:port of the original resource request.
- Logout Redirect URL: This parameter is automatically populated after agent registration completes. By default, this is based on the OAM Server host name with a default port of 14200.
- Logout Target URL: The value for this is name for the query parameter that the OPSS applications passes to WebGate during logout. This query parameter specifies the target URL of the landing page after logout.

For more information, see "Configuring Centralized Logout for 11g WebGate with OAM 11g Server" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Access Manager with Oracle Security Token Service*.

16.3.2 Logout for 10g WebGate with Oracle Access Manager 11g

Logout is initiated when an application causes the invocation of the logout.html file configured for the OAM Agent (in this case, a 10g WebGate). The application might also pass end_url as a query string to logout.html. The end_url parameter could either be a URI or a URL.

See Also: *Oracle Fusion Middleware Administrator's Guide for Oracle Access Manager with Oracle Security Token Service*

- About Centralized Logout with OAM 10g Agents and OAM 11g Servers
- Example 15-5: logout.html Script
- Configuring Centralized Logout for 10g Webgate with OAM

Task overview: Configuring centralized logout for 10g WebGates

1. Create a default logout page (logout.html) and make it available on the WebGate installation directory: For example, *WebGate_install_dir/oamssso/logout.html*.
2. In your logout.html, confirm that the logOutUrls parameter is configured for each resource WebGate and that <callBackUri> is the second value as part of 'logOutUrls'.
3. In your logout.html, confirm (from Step 1), confirm that the user is redirected to the central logout URI on the OAM 11g Server, "/oam/server/logout'.
4. **Optional:** Allow the application to pass the end_url parameter indicating where to redirect the user after logout.
5. Check the OHS Web server configuration file, httpd.conf, on which the 10g WebGate is configured and if the following lines exist delete them.

```
<LocationMatch "/oamssso/*">  
Satisfy any  
</LocationMatch>
```

For more information, see "Configuring Centralized Logout for 10g WebGate with OAM 11g Servers" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Access Manager with Oracle Security Token Service*.

16.4 Synchronizing the User and SSO Sessions: SSO Synchronization Filter

In Fusion Middleware 11g, a new component that synchronizes the container user session and SSO session has been introduced. SSO Sync Filter is an Oracle WebLogic system filter implementation that intercepts all requests to the container, acts on protected resource requests, and attempts to synchronize the container's user session with the user identifying header in OSSO (Proxy-Remote-User) or the user data in the Oracle Access Manager SSO session cookie (ObSSOCookie).

SSO Synchronization Filter (SSO Sync Filter) is an implementation of the Servlet Filter based on Java Servlet Specification version 2.3. SSO sync filter relieves applications from tracking the SSO user session and synchronizing it with their respective sessions. Instead, applications would only need to synchronize with container's user session.

SSO Sync Filter intercepts each request to the container and determines whether to act on it based on certain HTTP headers that are attached to the request. Filter expects SSO agent to have set those headers in the Web Tier. When access is made to unprotected areas of the application, the filter acts as a pass through. Once a protected resource is accessed, SSO agents in the Web Tier, direct user to perform authentication with SSO system such as Oracle Access Manager. After the authentication, Oracle Access Manager Identity Asserter helps establish a user identity in form of JAAS Subject to the container and a user session is created. WebLogic maintains the user session data as part of HTTP Session Cookie (JSESSIONID).

Subsequent access to the application resources provides two pieces of information to the SSO Sync Filter:

- User identifying header in OSSO (Proxy-Remote-User)
- User data in the Oracle Access Manager SSO session cookie (ObSSOCookie)

The job of SSO Sync Filter is to make sure that the user identity in the container matches with that of the SSO session. If there is a mismatch, filter invalidates the container's user session. As a result, the downstream application would only have to track container user session and react in a consistent fashion regardless of SSO environment in use.

Notes:

- **Enabled and Active by Default:** SSO Sync Filter fetches the user information from the configured tokens, gets the user from existing session (if any), invalidates the session and redirects to the requested URL in case the CurrentSessionUser does not match the incoming SSO User. Otherwise, the request is simply passed through.

If you have not configured the OSSO or Oracle Access Manager Assertion Providers in your domain, the filter disables automatically during WebLogic Server start-up.

- **Active for All URI's by Default (/*):** No changes are required in the application code.
- **Configured for the OSSO Tokens/Header:** Proxy-Remote-User, and performs a case insensitive match.
- **Configured for the Oracle Access Manager SSO Tokens/Header:** OAM_REMOTE_USER and REMOTE_USER, and does a case insensitive match.
- **Configured for the Oracle Access Manager SSO Tokens/Header:** OAM_IDENTITY_ASSERTION, a case insensitive match. For details, see "[Trusted Header Assertion: Configuring Digital Signature Verification](#)" on page 16-25.
- **Global Logout:** SSO Sync Filter is intended to provide the Single Logout Experience to the Oracle Fusion Middleware applications that use the OSSO or Oracle Access Manager Solutions. Is handled similarly to single sign-on. After global logout is performed, SSO filter reconciles the session when subsequent access to an application that has not cleaned up its session is made.

Any application that use the OSSO or Oracle Access Manager Solutions is expected to invalidate its session before making a call to OSSO logout or Oracle Access Manager logout. For more information on OSSO logout, see [Example 18-2, "SSO Logout with Dynamic Directives"](#) on page 18-11. For details about Oracle Access Manager logout, see "[Configuring Global Logout for Oracle Access Manager 10g and 10g WebGates](#)" on page 17-10.

- **Application Session Time Out:** SSO cookies typically track user inactivity/idle times and force users to login when a time out occurs. OSSO and Oracle Access Manager are no exception. Oracle Access Manager takes a sophisticated approach at this and specifically tracks Maximum Idle Session Time and Longest Idle Session Time along with SSO session creation time and time when it was last refreshed.

The general recommendation for applications that are maintaining their own sessions when integrating with SSO systems is to configure their session time outs close to that of SSO session time outs so as to make user experience remains consistent across SSO and application session time outs.

You can alter the behavior of the SSO Sync Filter for application requirements by passing various over-riding system properties to WebLogic. To do this, you change the Oracle WebLogic startup script and check for EXTRA_JAVA_PROPERTIES in setDomainEnv.sh. The properties and Sync behavior is shown in [Table 16-6](#).

Table 16-6 SSO Sync Filter Properties and Sync Behavior

Area	Overriding System Property	Default value of System property	Default Behavior of the Sync Filter
Status (Active or Inactive)	sso.filter.enable	Not configured	Enabled
Case sensitive matches	sso.filter.name.exact.match	Not configured	Case Ignore Match
Configured Tokens	sso.filter.ssotoken	Not configured	<ul style="list-style-type: none"> ■ OSSO: Look for Proxy-Remote-User ■ Oracle Access Manager: Look for OAM_REMOTE_USER and REMOTE_USER. OAM_REMOTE_USER takes precedence.
URI Mappings	Not Applicable	Not Applicable	/*

You cannot enable the filter for selected applications. The SSO Sync Filter is a system filter. As such, it is activated for all deployed applications (the URI mapping is /*).

Note: You cannot enable the filter for selected applications.

The following procedure gives some tips about modifying the SSO Sync filter properties and behavior.

To modify the SSO Sync Filter properties and behavior

1. **Disable the Filter:** Change the system property "sso.filter.enable" to "false" (pass as -D to the jvm) and restart the Oracle WebLogic Server. This toggles the filter status.
2. **User-Identifying Header Differs from Pre-Configured Sync Filter Tokens:** Over-ride the SSO token that the Sync Filter looks for using the system property "sso.filter.ssotoken".

For example, pass to the WebLogic Server jvm in the WebLogic Server startup script -Dssotoken=HEADERNAME, and restart the server.

When you contact Oracle Support you might be requested to set up debugging, as described in ["Setting Up Debugging in the WebLogic Administration Console"](#) on page 15-14.

16.5 Troubleshooting Tips

For more information, see "Troubleshooting" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Access Manager with Oracle Security Token Service*.

See Also: ["Troubleshooting Tips for OAM Provider Deployments"](#) on page 17-68

Configuring Single Sign-On Using Oracle Access Manager 10g

The chapter describes how to configure single sign-on using Oracle Access Manager 10g. It includes the following major sections:

- [Deploying SSO Solutions with Oracle Access Manager 10g](#)
- [Oracle Access Manager Authentication Provider Parameter List](#)
- [Introduction to OAMCfgTool](#)
- [Configuring OAM Identity Assertion for SSO with Oracle Access Manager 10g](#)
- [Configuring the Authenticator for Oracle Access Manager 10g](#)
- [Configuring Identity Assertion for Oracle Web Services Manager and OAM 10g](#)
- [Synchronizing the User and SSO Sessions: SSO Synchronization Filter](#)
- [Troubleshooting Tips for OAM Provider Deployments](#)

17.1 Deploying SSO Solutions with Oracle Access Manager 10g

This section provides the following topics:

- [Installing and Setting Up Authentication Providers for OAM 10g](#)
- [Oracle Access Manager Authentication Provider Parameter List](#)
- [Configuring Global Logout for Oracle Access Manager 10g and 10g WebGates](#)

17.1.1 Installing and Setting Up Authentication Providers for OAM 10g

This topic provides an overview of Oracle Access Manager installation and initial setup and additional information about installing components and files for use when you deploy the Oracle Access Manager Authentication Provider.

Unless explicitly stated, these topics describe requirements for both the Oracle Access Manager Identity Asserter and the Oracle Access Manager Authenticator:

- [About Oracle Access Manager 10g Installation and Setup](#)
- [Installing Components and Files for Authentication Providers and OAM 10g](#)
- [Converting Oracle Access Manager Certificates to Java Keystore Format](#)
- [Creating Resource Types in Oracle Access Manager 10g](#)

17.1.1.1 About Oracle Access Manager 10g Installation and Setup

This topic provides a brief installation and setup overview if you are new to Oracle Access Manager.

See Also: ["Requirements for the Provider with Oracle Access Manager"](#) on page 15-13

Access Servers: For the Oracle Access Manager Authentication Provider, you need two Access Servers for WebGates or AccessGates: one primary server and one secondary server. Currently, only one secondary Access Server is supported. Installing Access Servers includes:

- Adding an Access Server configuration profile in the Access System Console for the primary server. Ensure that the **Access Management Service is On** (also known as Policy Manager API Support Mode).
- Adding a secondary Access Server configuration profile with the **Access Management Service On**.
- Installing the primary Access Server instance.
- Installing the secondary Access Server instance.

WebGate/AccessGate: Whether you need a WebGate or an AccessGate depends on your use of the Oracle Access Manager Authentication Provider. For instance, the:

- **Identity Asserter for Single Sign-On:** Requires a separate WebGate and configuration profile for each application to define perimeter authentication. Ensure that the **Access Management Service is On**.
- **Authenticator or Oracle Web Services Manager:** Requires a separate AccessGate and configuration profile for each application. Ensure that the **Access Management Service is On**.

About OAM 10g WebGate/AccessGate Profiles and Policy Domains

This topic introduces the WebGate/AccessGate profiles, policy domains, and the methods you can use to create these.

While there are subtle differences between WebGates and AccessGates, these terms are often used interchangeably. In the Access System Console, the configuration profile for WebGates or AccessGates is known as an AccessGate profile. The Policy Manager is where an Oracle Access Manager policy domain is created.

Access System Console Method: Enables users with specific Oracle Access Manager administration rights to enter information and set parameters directly in Oracle Access Manager. This method is required if you are using the Authenticator, or if you have Oracle Web Services Manager policies protecting Web services.

OAMCfgTool Method: Application administrators who are implementing the Identity Asserter for single sign-on, can use OAMCfgTool to create a new WebGate profile for a fresh Web Tier. Required parameters are provisioned using values for your environment specified on the command line. Default values are accepted for non-required parameters; the Access Management Service is set to On. After creating a profile, values can be modified in the Access System Console.

Each AccessGate profile must include the following parameters; those marked with an asterisk, *, are provisioned with OAMCfgTool:

- ***AccessGate Name**—A unique name without spaces. With OAMCfgTool the name is derived from the app_domain value, appended with _AG.

- ***Hostname**—The name of the computer where the WebGate/AccessGate is or will be installed. With OAMCfgTool the app_domain value is used as the host name.
- ***AccessGate Password**—A unique password to verify and identify the component. This prevents unauthorized AccessGates from connecting to Access Servers and obtaining policy information. With OAMCfgTool, this is specified with the app_agent_password parameter. This should differ for each WebGate/AccessGate instance.
- **Transport Security**—The level of transport security between the Access Server and associated WebGates (these must match). The default value is Open. You can specify a different value with OAMCfgTool oam_aaa_mode value.
- ***Preferred HTTP Host**—The host name as it appears in all HTTP requests as users attempt to access the protected Web server. The host name in the HTTP request is translated into the value entered into this field, regardless of the way it was defined in a user's HTTP request. With OAMCfgTool the Preferred HTTP Host is the app_domain value.

The Preferred Host function prevents security holes that can be inadvertently created if a host's identifier is not included in the Host Identifiers list. However, it cannot be used with virtual Web hosting. For virtual hosting, you must use the Host Identifiers feature.

- ***Primary HTTP Cookie Domain:** The Web server domain on which the WebGate is deployed. The cookie domain is required to enable single sign-on among Web servers; each must have the same Primary HTTP Cookie Domain value. Use the cookie_domain parameter with the OAMCfgTool to set this value.

See Also:

- ["About Administrative Requirements for AccessGate Profiles and Policy Domains"](#) on page 17-3
- ["Introduction to OAMCfgTool"](#) on page 17-15
- "Configuring WebGates and Access Servers" in the *Oracle Access Manager Access Administration Guide*

About Administrative Requirements for AccessGate Profiles and Policy Domains

This topic introduces the administrative rights needed for the methods you can use when creating new WebGate and AccessGate profiles and policy domains for Oracle Access Manager.

An Oracle Access Manager Master Access Administrator must create the first policy domain after the policy domain root is defined. He or she can then create policy domains for URLs beneath the first one and delegate administration of those policy domains to other administrators.

Access System Console Method: You must be a Master or Delegated Access Administrator can use the Access System Console to create a new AccessGate profile, associate it with an Access Server, and create an authentication scheme. Master or Delegated Access Administrators can also use the Policy Manager to create a policy domain. The following deployments require this method:

- Authenticator
- Identity Asserter when Oracle Web Services Manager is protecting Web services

OAMCfgTool Method: You do not need specific Oracle Access Manager administration rights for OAMCfgTool, which automates creating and associating a

WebGate profile and creating a new policy domain. However, this method can be used for only Identity Assertion. In a:

- **Fresh Web Tier:** Use OAMCfgTool to streamline creating a new WebGate profile and policy domain for Identity Asserter only.

After creating the profile and policy domain with OAMCfgTool, these can be modified in the Access System Console.

See Also: ["Introduction to OAMCfgTool"](#) on page 17-15

- **Existing Web Tier:** When one or more WebGates exist in the Web Tier, no new WebGate is needed. However, you can specify an existing host identifier to make newly established policies enforceable by an existing WebGate.

See Also:

- ["Installing Components and Files for Authentication Providers and OAM 10g"](#)
- "Configuring WebGates and Access Servers" in the *Oracle Access Manager Access Administration Guide*

17.1.1.2 Installing Components and Files for Authentication Providers and OAM 10g

The following task overview outlines the components and files that must be installed and where to locate more information.

Note: If you already have components installed and set up, you do not need to install new ones. Skip any steps that do not apply to your deployment.

Unless specifically stated, all details apply whether you intend to deploy the Identity Asserter for single sign-on, or the Authenticator, or if Oracle Web Services Manager policies are protecting Web services.

Task overview: Installing required components and files for Oracle Access Manager 10g Authentication Provider

1. An Oracle Internet Directory or Oracle Sun One LDAP directory server configured to be used by the Oracle Access Manager Access Server. Ensure that the directory server is tuned for your deployment.

See Also: The following Release 11g (11.1.1.1.0) manuals

- *Oracle Fusion Middleware Installation Guide for Oracle Identity Management*
- *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*

2. Install and set up Oracle WebLogic Server 10.3.1+.

See Also: Item 3 in this list, and the *Oracle Fusion Middleware Getting Started With Installation for Oracle WebLogic Server*

3. **Optional:** Install a Fusion Middleware product (Oracle Identity Manager, Oracle SOA Suite, or Oracle Web Center for example):

Note: Without a Fusion Middleware application, you must acquire the required JAR and WAR files as described in later procedures.

- a. Confirm the location of required JAR files in the following Fusion Middleware path:

```
ORACLE_INSTANCE/modules/oracle.oamprovider_11.1.1/oamAuthnProvider.jar
ORACLE_INSTANCE/modules/oracle.oamprovider_11.1.1/oamcfgtool.jar
```

- b. Locate the console-extension WAR file in the following path:

```
ORACLE_INSTANCE/modules/oracle.oamprovider_11.1.1/oamauthenticationprovider.war
```

- c. Copy the WAR file to the following path in the WebLogic Server home:

```
WL_HOME/server/lib/console-ext/autodeploy/oamauthenticationprovider.war
```

4. Install OHS 11g for the Oracle Access Manager 10g (10.1.4.3) WebGate, if needed:

- **Authenticator or Oracle Web Services Manager:** No Web server is required for the custom AccessGate. The protected resource is accessed using its URL on the Oracle WebLogic Server.
- **Oracle Access Manager Identity Asserter:** Requires Oracle HTTP Server 11g Web server configured as a reverse proxy in front of Oracle WebLogic Server.

5. Install Oracle Access Manager 10g (10.1.4.3) components and perform initial setup as follows:

See Also: ["About Oracle Access Manager 10g Installation and Setup"](#) on page 17-2

- a. Install an Identity Server; install a WebPass; set up the Identity System.
- b. Install and set up Policy Manager. Ensure that the policy protecting the Policy Manager, /access, is created and enabled, as well as the default authentication schemes.
- c. Install Access Servers (one as a primary server and one as a secondary server for WebGate).
 - Add an Access Server configuration profile in the Access System Console for the primary server for WebGate. Ensure that the **Access Management Service** is **On** (also known as Policy Manager API Support Mode).
 - Add a secondary Access Server configuration profile with the **Access Management Service On**.
 - Install the primary Access Server instance and then install the secondary Access Server instance.

Note: Only one secondary Access Server is supported

- d. **WebGate for Identity Asserter for Single Sign-On:** In an existing Web Tier with one or more WebGates, no new WebGates or profiles are needed.

See Also: ["Introduction to OAMCfgTool"](#) on page 17-15

In a fresh Web Tier, you must create a profile to define the WebGate for perimeter authentication, as follows:

- Create an AccessGate configuration profile to define the WebGate for perimeter authentication. Ensure that the **Access Management Service is On**. You can use the OAMCfgTool or Access System Console.
 - Associate the WebGate profile with a primary and a secondary Access Server.
 - Install a WebGate for Oracle HTTP Server 11g configured as a reverse proxy for every application.
 - Repeat until you have a profile and a WebGate protecting each application.
- e. **AccessGate:** For the Authenticator, or when you have Oracle Web Services, Manager you must add a new profile for custom AccessGates in the Access System Console

See Also: ["About OAM 10g WebGate/ AccessGate Profiles and Policy Domains"](#) on page 17-2

- Add an AccessGate configuration profile in the Access System Console and ensure that the **Access Management Service is On**.
 - Associate the AccessGate profile with a primary and a secondary Access Server.
 - Deploy the custom AccessGate in oamAuthnProvider.jar.
 - Repeat until you have a profile and a AccessGate protecting each application.
6. Proceed as follows:
- **Simple or Cert Mode:** ["Converting Oracle Access Manager Certificates to Java Keystore Format"](#)
 - **Authenticator or Oracle Web Services Manager:** ["Creating Resource Types in Oracle Access Manager 10g"](#) on page 17-9 must be performed if you use the Oracle Access Manager Authenticator or if you have Oracle Web Services Manager policies protecting Web services.
 - **Identity Asserter for Single Sign-On:** Perform tasks in ["Configuring OAM Identity Assertion for SSO with Oracle Access Manager 10g"](#) on page 17-35.

17.1.1.3 Converting Oracle Access Manager Certificates to Java Keystore Format

Oracle recommends that all Java components and applications use JKS as the keystore format. This topic provides steps to convert Oracle Access Manager X.509 certificates to Java Keystore (JKS) format.

These steps, when followed properly, generate the JKS stores that can be used while the Java NAP client wants to communicate with an Oracle Access Manager Access Server in Simple or Cert (certificate) mode.

When communicating in Simple or Cert mode, the Access Server uses a key, server certificate, and CA chain files:

- `aaa_key.pem`: the random key information generated by the certificate-generating utilities while it sends a request to a Root CA. This is your private key. The certificate request for WebGate generates the certificate-request file `aaa_req.pem`. You must send this WebGate certificate request to a root CA that is trusted by the Access Server. The root CA returns the WebGate certificates, which can then be installed either during or after WebGate installation.
- `aaa_cert.pem`: the actual certificate for the Access Server, signed by the Root CA.
- `aaa_chain.pem`: the public certificate of the Root CA. This is used when peers communicating in Simple or Cert mode perform an SSL handshake and exchange their certificates for validity. In Simple Mode, the `aaa_chain.pem` is the OpenSSL certificate located in `AccessServer_install_dir/access/oblix/tools/openssl/simpleCA/cacert.pem`

Here, *aaa* is the name you specify for the file (applicable only to Cert and chain files).

You can edit an existing certificate with a text editing utility to remove all data except that which is contained within the `CERTIFICATE` blocks. You then convert the edited certificate to JKS format, and import it into the keystore. Java KeyTool does not allow you to import an existing Private Key for which you already have a certificate. You must convert the PEM format files to DER format files using the OpenSSL utility.

To convert an Oracle Access Manager certificate to JKS format and import it

1. Install and configure Java 1.6 or the latest version.
2. Copy the following files before editing to retain the originals:
 - `aaa_chain.pem`
 - `aaa_cert.pem`
 - `cacert.pem`, only if configuring for Simple mode
3. Edit `aaa_chain.pem` using TextPad to remove all data except that which is contained within the `CERTIFICATE` blocks, and save the file in a new location to retain the original.

```
-----BEGIN CERTIFICATE-----
...
CERTIFICATE
...
-----END CERTIFICATE-----
```

4. Run the following command for the edited `aaa_chain.pem`:

```
JDK_HOME\bin\keytool" -import -alias root_ca -file aaa_chain.pem -keystore
rootcerts
```

Here you are assigning an alias (short name) **root_ca** to the key. The input file `aaa_chain.pem` is the one that you manually edited in step 3. The keystore name is `rootcerts`.

You must give a password to access the keys stored in the newly created keystore.

Note: To ensure security, Oracle recommends that you allow the keytool to prompt you to enter the password. This prompt occurs automatically when the “-storepass” flag is omitted from the command line.

5. Enter the keystore password, when asked. For example:

```
Enter keystore password: <keystore_password>
Re-enter new keystore password: <keystore_password>
```

6. Enter Yes when asked if you trust this tool:

```
Trust this certificate? [no]: yes
```

7. Confirm that the certificate has been imported to the JKS format by executing the following command and then the password.

```
JDK_HOME\bin\keytool" -list -v -keystore "rootcerts"
Enter keystore password: <keystore_password>
```

8. Look for a response like the following:

```
Keystore type: JKS
Keystore provider: SUN
Your keystore contains n entries
Alias name: root_ca
Creation date: April 19, 2009
Entry type: trustedCertEntry

Owner: CN=NetPoint Simple Security CA - Not for General Use, OU=NetPoint,
O="Oblix, Inc.", L=Cupertino, ST= California , C=US

Issuer: CN=NetPoint Simple Security CA - Not for General Use, OU=NetPoint,
O="Oblix, Inc.", L=Cupertino, ST= California ,C=US

Serial number: x
Valid from: Tue Jul 25 23:33:57 GMT+05:30 2000 until: Sun Jul 25 23:33:57
GMT+05:30 2010

Certificate fingerprints
MD5: CE:45:3A:66:53:0F:FD:D6:93:AD:A7:01:F3:C6:3E:BC
SHA1: D6:86:9E:83:CF:E7:24:C6:6C:E1:1A:20:28:63:FE:FE:43:7F:68:95
Signature algorithm name: MD5withRSA
Version: 1
*****
```

9. Repeat steps 3 through 7 for the other PEM files (except aaa_chain.pem unless there is a chain).
10. Convert the aaa_key.pem file to DER format using the OpenSSL utility in the Access Server installation directory path. For example:

```
AccessServer_install_dir\access\oblix\tools\openssl>openssl pkcs8 -topk8
-nocrypt -in aaa_key.pem -inform PEM -out aaa_key.der -outform DER
```

Here the input file is aaa_key.pem and the output file is aaa_key.der. Additional options include:

Table 17–1 Options to Create DER Format Files from PEM

Option	Description
-topk8	Reads a traditional format private key and writes a PKCS#8 format key. This reverses the default situation where a PKCS#8 private key is expected on input and a traditional format private key is written.
-nocrypt	An unencrypted PrivateKeyInfo structure is expected for output.

Table 17-1 (Cont.) Options to Create DER Format Files from PEM

Option	Description
-inform	Specifies the input format. If a PKCS#8 format key is expected on input, then either a DER or PEM encoded version of a PKCS#8 key is expected. Otherwise the DER or PEM format of the traditional format private key is used.
-outform	Specifies the output format. If a PKCS#8 format key is expected on output, then either a DER or PEM encoded version of a PKCS#8 key is expected. Otherwise the DER or PEM format of the traditional format private key is used.

- 11. Simple or Cert Mode:** In the PEM file (in this case, `aaa_cert.pem`), enter the passphrase for the Oracle Access Manager Access Server if it is configured for Simple or Cert mode.

```
Passphrase for the certificate
```

- 12.** Run the following command to convert the `aaa_cert.pem` file to DER format.

```
AccessServer_install_dir\access\oblix\tools\openssl>openssl x509 -in
aaa_cert.pem -inform PEM -out aaa_cert.der -outform DER
```

- 13.** Import the DER format files into a Java keystore using the ImportKey utility. For example:

```
Java_install_dir\doc>java -Dkeystore=jks certs ImportKey aaa_key.der
aaa_cert.der
```

- 14.** Review the results in the window, which should look something like the following example:

```
Using keystore-file : jks certs
One certificate, no chain
Key and certificate stored
Alias:importkey Password:your_password
```

- 15.** Proceed as follows:

- **Identity Asserter for Single Sign-On:** Go to "[Configuring OAM Identity Assertion for SSO with Oracle Access Manager 10g](#)" on page 17-35.
- **Authenticator or Oracle Web Services Manager:** Perform steps in "[Creating Resource Types in Oracle Access Manager 10g](#)".

17.1.1.4 Creating Resource Types in Oracle Access Manager 10g

This section describes how to create resource types in Oracle Access Manager to identify the types of resources that you want the policy domain to protect. This task is required if you use the Oracle Access Manager Authenticator or if you have Oracle Web Services Manager policies protecting Web services.

You use the Oracle Access Manager Access System Console to define resource types as described here.

Note: If you are using the Oracle Access Manager Identity Asserter for single sign-on, you can skip this task. In this case, only the default http resource type is used.

Defining the `wl_authen` resource type in Oracle Access Manager is required only when you are using:

- Oracle Access Manager Authenticator
- Identity Asserter with Oracle Web Services Manager

To define resource types in Oracle Access Manager 10g

1. Go to the Access System Console and log in.
2. Select the **Access System Configuration** tab, and then click **Common Information Configuration, Resource Type Definitions**, to display the List All Resource Types page.
3. On the List All Resource Types page, click **Add**, to display the Define a new Resource Type page.
4. Define the resource type with the following details:
 - Name: `wl_authen`
 - Display name: `wl_authen`
 - Resource matching: **Case insensitive**
 - Resource operation: `LOGIN`
5. Save the resource type you just defined.
6. Proceed as follows:
 - **Authenticator:** ["Configuring the Authenticator for Oracle Access Manager 10g"](#) on page 17-49
 - **Oracle Web Services Manager:** ["Configuring Identity Assertion for Oracle Web Services Manager and OAM 10g"](#) on page 17-49

17.1.2 Configuring Global Logout for Oracle Access Manager 10g and 10g WebGates

This section discusses configuring logout for applications protected by a 10g WebGate with Oracle Access Manager 10g. In Oracle Access Manager 10g, global logout (also known as single log out (SLO) can be handled in various ways. This section describes the recommended method.

Note: Oracle Access Manager SSO user session tracking is performed using DOMAIN cookies, specifically the `ObSSOCookie`. WebGates look for the `ObSSOCookie`. Global or SLO for Oracle Access Manager simply means killing the `ObSSOCookie`. Without the `ObSSOCookie`, WebGates enforce a re-authentication workflow.

For more information on killing the `ObSSOCookie`, see:

- [Recommended Process for Configuring Logout](#)
- [Alternative Process for Configuring Logout](#)

17.1.2.1 Recommended Process for Configuring Logout

There are two steps in the Oracle-recommended approach to configuring logout:

- [Configuring WebGate for Logout using the Sample Logout File](#)
- [Configuring Applications for Logout](#)

17.1.2.1.1 Configuring WebGate for Logout using the Sample Logout File

WebGate configuration consist of:

- `logout.html`: A logout page must be available on the Web server in the WebGate installation directory: `WebGate_install_dir/oamssso/logout.html`.

If the file is located elsewhere on the Web server, ensure that the logout link is correctly specified to load `logout.html`. See the `logout.html` in [Example 17-1](#) on page 17-11, which you can customize further depending on your needs.

- `logOutUrls` (optional): If this parameter has already been configured for the WebGate, the value `"/oamssso/logout.html"` must be added to the existing list.
- **Web Server Configuration**: Check the Oracle HTTP Server Web server configuration file, `httpd.conf`, on which the 10g WebGate is configured and delete the following lines if they are present

```
<LocationMatch "/oamssso/*">
Satisfy any
</LocationMatch>
```

Use [Example 17-1](#) when you begin constructing a `logout.html` for logout configuration for an application protected by 10g WebGate in an OAM 10g deployment.

Example 17-1 *logout.html Script*

```
<html>
<head>
<script language="javascript" type="text/javascript">

function handleLogout() {

    //get protocol used at the server (http/https)
    var webServerProtocol = window.location.protocol;
    //get server host:port
    var webServerHostPort = window.location.host;
    //get query string present in this URL
    var origQueryString = window.location.search.substring(1);

    //vars to parse the querystring
    var params = new Array();
    var par = new Array();
    var val;

    if (origQueryString != null && origQueryString != "") {

        params = origQueryString.split("&");

        //search for end_url and redirect the user to this
        for (var i=0; i<params.length; i++) {

            par = params[i].split("=");

            if ("end_url" == par[0]) {

                endUrlVal = par[1];

                //check if val (value of end_url) begins with "/" or "%2F" (is it an URI?)
                if (endUrlVal.substring(0,1) == "/" || endUrlVal.substring(0,1) == "%") {

                    if (endUrlVal.substring(0,1) == "%")
```

```
        endUrlVal = "/" + endUrlVal.substring(3);

        //modify the end_url value now
        endUrlVal = webServerProtocol + "://" + webServerHostPort + endUrlVal;
    }
    //redirect the user to this URL
    window.location.href = endUrlVal;
    }
}
}
}
</script>
</head>

<body onLoad="handleLogout();" >

<h3>You have been logged out</h3>

</body>
</html>
```

17.1.2.1.2 Configuring Applications for Logout Application configuration for logout depends on whether it is an ADF application integrated with OPSS or if it is not integrated with OPSS.

Note: The logout configuration assumes that the applications are present in a single DNS domain. If you would like SLO (single log out) across applications deployed in different DNS domains, you must customize the logout script to ensure processing for each WebGate. If you have a multi DNS domain deployment, see the *Oracle Access Manager Access Administration Guide*.

One of the following must be done to configure the application for logout:

- A non-ADF application must be coded to invoke the logout link: `"/oamssso/logout.html?end_url=<target uri>"`.
- An ADF application that has been integrated with OPSS requires configuring OPSS for the OAM SSO provider and the application must send the 'end_url' parameter.

Non-ADF Application

A non-ADF application must be coded to invoke the link for logout: `"/oamssso/logout.html?end_url=<target uri>"`.

The application can pass a parameter (named `end_url`) indicating the location where the user should eventually be redirected to after logout. The value that is part of `end_url` could either be a URL or a URI. For example, the logout link for the application might be specified as

```
/oamssso/logout.html?end_url=<someUri>
```

or

```
/oamssso/logout.html?end_url=<someUrl>
```

If the `end_url` querystring is a URI, then the `logout.html` must construct the URL by determining the `host:port` of the server where `logout.html` is hosted.

ADF-Coded Applications

If the Application is an ADF application that has been integrated with OPSS, then you can use the following procedure to configure logout.

To configure centralized logout for ADF-coded applications

1. Check with the OAM Administrator to confirm the location of the `logout.html` script configured with the Agent, which you need in following steps.
2. Configure OPSS for OAM as the SSO provider to update `jps-config.xml` for the WebLogic administration domain, as follows:
 - a. Run `wlst.sh` (Linux) or `wlst.cmd` (Windows) from the following directory path:

```
WLST_install_dir/middleware/oracle_common/common/bin
```

- b. At the WLS prompt, enter the OAM administrator ID and password and the WebGate host and port:

```
wls:/> /connect("admin_ID", "admin_pw", "hostname:port"
```

The last parameter is optional if the server is running on localhost at the default port (7001).

- c. Enter the login URI for ADF authentication and the logout URI (location of the `logout.html` script configured with the agent); the host and port are not needed.

```
wls:/>addOAMSSOProvider(loginuri="/${loginuri}",
logouturi="<logouturl>," autologinuri=None")
```

Here, `logouturl` is the URI of the logout script `/oamssso/logout.html`. The `logouturl` could either begin with "logout" (exceptions are `logout.gif` and `logout.jpg`) or could be any other value configured by the OAM Administrator.

3. **Required:** The ADF application must pass the `end_url` parameter indicating where to redirect the user after logout. For ADF applications, logout is initiated when the application causes the following URI to be invoked:

```
</app context root>/adfAuthentication?logout=true&end_url=<any uri>
```

17.1.2.2 Alternative Process for Configuring Logout

Oracle does not recommend this method unless your application already has a custom logout page that you do not wish to change for any reason.

WebGate logs out of any request for a URL that has the string "logout." in it.

Exception: Image files such as `logout.gif` and `logout.jpg`. This is the simplest way to integrate an application with OAM SLO. If your logout page begins with "logout." (for example, `logout.jsp`) then you do not need to do any thing.

Note: If your logout page begins with "logout." (for example, `logout.jsp`) then you do not need to do any thing.

If your logout page does not begin with "logout.", then you must add your logout URL to the WebGate `LogOutUrls` parameter. For instance: `LogOutUrls = "/myapplication/customscript.jsp"`.

17.2 Oracle Access Manager Authentication Provider Parameter List

This section enumerates the common and provider-specific parameters relevant to the Oracle Access Manager Authentication Provider. These are specified in the Oracle WebLogic Administration Console. For more information, see:

- [Table 17–2, "Oracle Access Manager Authentication Provider Common Parameters"](#)
- [Table 17–3, "Provider-Specific Parameters"](#)
- [Table 17–4, "Provider-Specific Parameters: Oracle Access Manager Authenticator"](#)

Table 17–2 Oracle Access Manager Authentication Provider Common Parameters

Parameter Name	Parameter Description
Name	The name of the provider. Read-only.
Description	The description of the provider. Read-only.
Version	The version of the provider. Read-only.
Control Flag	The provider JAAS control flag. Set one of the following: REQUIRED, REQUISITE, OPTIONAL, or SUFFICIENT. When configuring multiple Authentication Providers, use this flag to control how they are use in the login sequence. See JAAS Control Flag .
Active Types	This parameter is relevant to only Oracle Access Manager Identity Asserter. This parameter determines the token types that the Identity Asserter Provider processes. Set as follows for OAM 10g and 10g WebGate: <ul style="list-style-type: none"> ■ OAM 10g and 10g WebGate: ObSSOCookie ■ OAM_REMOTE_USER
Base64 Decoding Required	False is Read-only (the default).

The WebLogic Server Administration Console sets the JAAS Control Flag to OPTIONAL when you create a new security provider. The default value for out-of-the-box security providers is REQUIRED. For more details about the control flag, see the online help.

[Table 17–3](#) lists the provider-specific parameters for Oracle Access Manager the Authenticator or the Identity Asserter for Oracle Web Services Manager.

Note: With OAM 11g, the Access Server is known as the OAM Server.

Table 17–3 Provider-Specific Parameters

Parameter Name	Parameter Description
Transport Security	The mode of communication between AccessGate and Access Server.
Minimum Access Server Connections In Pool	The minimum number of connections allowed. Default is 5.
Access Gate Password	The password of the AccessGate used by the provider.
Key Store Pass Phrase	The password to access the key store.
Access Gate Name	The name of the AccessGate used by the provider. Required.
Primary Access Server	The name of the primary access server. It must conform to the format <i>host:port</i> . Required. See "Installing and Setting Up Authentication Providers for OAM 10g" on page 17-1.

Table 17-3 (Cont.) Provider-Specific Parameters

Parameter Name	Parameter Description
Maximum Access Server Connections In Pool	The maximum number of connections allowed. Default is 10. Set to 1.
Simple Mode PassPhrase	The password shared by AccessGate and Access Server for Simple or Cert communication modes.
Trust Store	The absolute path of JKS trust store used for SSL communication between the provider and the Oracle Access Manager Access Server.
SSOHeader Name	OAM_REMOTE_USER
Secondary Access Server	The name of the secondary access server. It must conform to the format <i>host:port</i> . See "Installing and Setting Up Authentication Providers for OAM 10g" on page 17-1.
Key Store	The absolute path of JKS key store used for SSL communication between the provider and the Oracle Access Manager Access Server.

[Table 17-4](#) lists provider-specific parameters for the Oracle Access Manager Authenticator.

Table 17-4 Provider-Specific Parameters: Oracle Access Manager Authenticator

Parameter Name	Parameter Description
Transport Security	The mode of communication between AccessGate and Access Server.
Maximum Access Server Connections In Pool	The maximum number of connections allowed. Default is 10. Set to 1.
Simple Mode Pass Phrase	The password shared by AccessGate and Access Server for simple or cert communication modes.
Minimum Access Server Connections In Pool	The minimum number of connections allowed. Default is 5.
Trust Store	The absolute path of JKS trust store used for SSL communication between the provider and the Oracle Access Manager Access Server.
Use Retrieved username As Principal	Specifies whether to use the user name retrieved from Oracle Access Manager as the Principal in the Subject.
Access Gate Password	The password of the AccessGate used by the provider.
Key Store Pass Phrase	The password to access the key store.
Access Gate Name	The name of the AccessGate used by the provider. Required.
Secondary Access Server	The name of the secondary access server. It must conform to the format <i>host:port</i> . See: <ul style="list-style-type: none"> ▪ "Installing and Setting Up Authentication Providers for OAM 10g" on page 17-1
Key Store	The absolute path of JKS key store used for SSL communication between the provider and the Oracle Access Manager Access Server.
Primary Access Server	The name of the primary access server. It must conform to the format <i>host:port</i> . Required. See: <ul style="list-style-type: none"> ▪ "Installing and Setting Up Authentication Providers for OAM 10g" on page 17-1

17.3 Introduction to OAMCfGTool

This topic introduces OAMCfGTool, which can be used only if you are deploying the Oracle Access Manager 10g Identity Asserter for single sign-on.

OAMCfGTool launches a series of scripts to request information and set up the required profiles and policies in Oracle Access Manager 10g. OAMCfGTool runs in the following modes:

- CREATE mode

```
java -jar oamcfgtool.jar mode=CREATE param=value
```

- VALIDATE mode

```
java -jar oamcfgtool.jar mode=VALIDATE param=value
```

- DELETE mode

```
java -jar oamcfgtool.jar mode=DELETE param=value
```

Unless you specify an LDIF output file, configuration changes are written directly in the LDAP directory server that is configured with Oracle Access Manager. In addition, without an LDIF file, OAM Access Server's cache is updated with the configuration changes.

Note: When configuration changes are written to an LDIF file, it can be loaded into the directory server for Oracle Access Manager at a later time.

You can also specify a log level and an output file for logging details. If errors occur when running OAMCfGTool, these are reported on the command line.

Passwords

OAMCfGTool expects four passwords: LDAP user, Application agent, OAM mode, and Test user.

Without the `-noprompt` parameter, OAMCfGTool attempts to fetch passwords first from the command line. If no password is found, then OAMCfGTool pauses and prompts for a password to be entered on the command line.

However if you specify the `-noprompt` parameter, OAMCfGTool checks for passwords passed from the command line:

- If the password was not passed from command line, then OAMCfGTool checks for passwords passed from System.in.
- If no password is passed from System.in, OAMCfGTool stops execution with an exception indicating that the required password was not provided.

Passwords can be passed from a shell using an echo command and a semi-colon as a separator. For instance:

- To specify all four passwords:

```
$ (echo ldapUserPwd; echo appAgentPwd; echo OAMModePwd; echo TestUserPwd) | java -jar oamcfgtool.jar <args> -noprompt
```

- To specify only ldapUserPwd and appAgentPwd:

```
$ (echo ldapUserPwd; echo appAgentPwd) | java -jar oamcfgtool.jar <args> -noprompt
```

- To specify only appAgentPwd:

```
$ (echo; echo appAgentPwd) | java -jar oamcfgtool.jar <args> -noprompt
```

For more information, see ["OAMCfgTool Parameters and Values"](#) on page 17-17.

17.3.1 OAMCfgTool Process Overview

This topic describes the processing that occurs when you use OAMCfgTool with various parameters and values for your environment.

This topic focuses on using OAMCfgTool for OAM 10g. If you are using OAM 11g, skip this topic and instead refer to the chapter [Chapter 16, "Configuring Single Sign-On with Oracle Access Manager 11g"](#).

See Also:

- ["OAMCfgTool Parameters and Values"](#) on page 17-17
- ["Sample Policy Domain and AccessGate Profile Created with OAMCfgTool"](#) on page 17-29

Process overview: OAMCfgTool creates the authentication scheme, policy domain, and WebGate profile

1. The `app_domain` parameter creates a policy domain in the Policy Manager to enable authentication for the application.
2. The `web_domain` parameter creates a host identifier that connects the WebGate host sending requests to your application and links the policy to the existing WebGate.

Note: See the `web_domain` parameter in [Table 17-5](#).

3. The `protected_uris` parameter defines application-specific URL's to protect HTTP resources using the host identifier.
4. The `public_uris` parameter creates a policy to protect certain URIs with the Anonymous Authentication scheme for http resources (GET and POST operations) in the `app_domain` *name*.

Note: See the `uris_file` parameter in [Table 17-5](#) for details about specifying protected and public URIs in a file.

5. The LDAP parameters specify the directory server used by Oracle Access Manager to identify the searchbase from which all LDAP queries are performed. For more information, see [Table 17-5](#)
6. The log file and level parameters specify an output file and logging level for OAMCfgTool.
7. The `output_ldif_file` parameter defines the name of the LDIF file that is created to be loaded later in the directory server, if specified. Otherwise, configuration changes are written to the directory server

17.3.2 OAMCfgTool Parameters and Values

Find the following topics here:

- [Create Mode Parameters and Values](#)
- [Validate Mode Parameters and Values](#)

- Delete Mode Parameters and Values

17.3.2.1 Create Mode Parameters and Values

Table 17–5 provides both required and optional OAMCfGTool parameters and values for CREATE mode. You can specify multiple parameters at one time.

Table 17–5 OAMCfGTool CREATE Mode Parameters and Values

Parameters	CREATE Mode Values
Required Parameters	Required Parameter Values
app_domain	Name of the Oracle Access Manager policy domain to protect the application. Within the Policy Manager this is known as the policy domain name.
protected_uris	URIs for the protected application in a comma separated list (with or without spaces): /myapp/login, for example. See Also: The uris_file parameter in this table.
uris_file	The full path to a file containing any number of protected or public URIs and eliminates the need to use the protected_uris or public_uris parameters. Ensure that the file uses the following syntax and format: --At least one protected URI is required. --Only one product family is allowed per file. --Comments begin with '#' --Keyword: public_uris. List public URIs on separate lines after this key word. --Key word: protected_uris. List URIs to be protected on separate lines after this key word. For example: ##### #Finance ##### . ##### protected_uris ##### /finance/protected/test1 /finance/protected/test2 ##### public_uris ##### /finance/public /finance/protected/test1/public
app_agent_password	Password to be provisioned for the WebGate. In the AccessGate Profile within the 10g Access System Console, this parameter is known as the AccessGate Password. Your entry appears in clear text on the command line but is not captured in a log file. Note: This parameter is not required if you will not create a WebGate profile. See Also: -noprompt later in this table and the discussion "Passwords" on page 17-16.
ldap_host	DNS name of the computer hosting the LDAP directory server for Oracle Access Manager. This is the directory server containing the OAM policy data. Note: SSL-enabled communication with the directory server is not supported.
ldap_port	Port of the LDAP directory server.
ldap_userdn	The valid DN of the LDAP administrative user, entered as a quoted string. In Oracle Access Manager this is known as the Root DN or Bind DN.

Table 17-5 (Cont.) OAMCfGTool CREATE Mode Parameters and Values

Parameters	CREATE Mode Values
ldap_userpassword	<p>Password of LDAP administrative user. Passwords appear in clear text but are not captured in a log file. See Also: -noprompt later in this table.</p> <p>See Also: -noprompt later in this table and the discussion "Passwords" on page 17-16.</p>
oam_aaa_host	<p>DNS name of the computer hosting an accessible Access Server.</p> <p>After making appropriate changes to the Directory Server, a Cache flush request would be sent to this Access Server so that Access Servers refresh their appropriate caches.</p> <p>If the 'primary_oam_servers' parameter is not specified, then the WebGate profile being created would be configured to use the Access Server, specified as part of oam_aaa_host, as the Primary Access Server. Number of connections would default to 1.</p> <p>See Also: primary_oam_servers and secondary_oam_servers, later in this table.</p>
oam_aaa_port	Listening port on the accessible Access Server
Optional Parameters	Optional Parameter Values
help	Provides a list of parameters and descriptions.
version	Lists the version of the OAMCfGTool.
web_domain	<p>Primarily used to specify the host identifier.</p> <p>Note: OAMCfGTool either creates a host identifier and Webgate profile together or does not create either of them, as described in the following two scenarios:</p> <p>Creation of a Fresh Web Tier: If the host identifier specified by the parameter "web_domain" (or "app_domain" if "web_domain" is not specified) does not exist in OAM, then the following would be created in OAM:</p> <ol style="list-style-type: none"> 1. A new host identifier is created with the value specified by "web_domain" (or "app_domain" if "web_domain" is not specified). 2. A new WebGate profile, the name of which is derived using the following rules: <ol style="list-style-type: none"> a. If "webgate_id" is specified, then the WebGate profile is created with the value specified in "webgate_id" b. If "webgate_id" is not specified, then the WebGate profile is created with the value specified in "web_domain" with "_AG" appended to it. For example: <web_domain>_AG. c. If "webgate_id" and "web_domain" are not specified, then the WebGate profile is created with the value specified in "app_domain" with "_AG" appended to it. For example: <app_domain>_AG. 3. The value of the "Preferred Http Host" field of the WebGate profile and the "hostname variations" as part of the Host Identifier created in step 1 above are automatically populated with a same value. <p>See Also: The hostname_variations parameter in this table for configuring virtual hosts.</p> <p>Using an existing Web Tier (Join a web domain): If the host identifier specified as part of "web_domain" (or "app_domain", if "web_domain" is not specified) exists in OAM, then:</p> <ul style="list-style-type: none"> ■ A host identifier is not created ■ A WebGate profile is not created <p>Note: The host identifier created in a fresh Web Tier is used in the policy domain being used.</p> <p>If virtual Web hosting is supported, supply a reserved name in the Preferred HTTP Host field instead of a host name variation.</p> <p>See Also: The hostname_variations parameter in this table and the <i>Oracle Access Manager Access Administration Guide</i>.</p>

Table 17-5 (Cont.) OAMCfGTool CREATE Mode Parameters and Values

Parameters	CREATE Mode Values
cookie_domain	Name of the domain to use for the ObSSOCookie. Within the AccessGate Profile in the Access System Console, this is known as the Primary HTTP Cookie Domain. Use this parameter when you create a new WebGate profile in a fresh Web Tier.
public_uris	URIs that must be unprotected using the Anonymous authentication scheme. You can identify public URIs by providing a comma separated list: "uri1,uri2,uri3", for example. See Also: The uris_file parameter in this table.
ldap_base	Base from which all LDAP searches are performed.
oam_aaa_mode	Transport security mode of the accessible Access Server: OPEN, SIMPLE, or CERT. Default presumes OPEN.
oam_aaa_passphrase	Passphrase required for SIMPLE mode transport security mode only. The passphrase appears in clear text but is not captured in a log file. See Also: The discussion "Passwords" on page 17-16.
log_file	Name of the OAMCfGTool log file. Output to the screen is the default.
log_level	Level for OAMCfGTool logging: ALL, SEVERE, WARNING, INFO, CONFIG, FINE, FINER, FINEST, OFF. Default = WARNING
output_ldif_file	Name of the LDIF file in which to store details from OAMCfGTool operations to load into the LDAP directory server later. If none is specified, changes are written immediately to the LDAP directory server and caches in Oracle Access Manager are flushed to make new information available.
noprompt	Disables password prompts from OAMCfGTool and enables password checks as follows: <ul style="list-style-type: none"> ▪ If no password was passed from the command line, then OAMCfGTool checks for passwords passed from System.in. See Also: "Passwords" on page 17-16 for more information. ▪ If no password is passed from System.in, OAMCfGTool stops execution with an exception indicating that the required password was not provided.
authenticating_wg_url	URI containing the host and port of the authenticating WebGate (when you have both an authenticating and a resource WebGate). For example: <code>authenticating_wg_uri=http://host:port</code> This parameter configures the "Challenge Redirect Parameter" of both the following authentication schemes: <ul style="list-style-type: none"> ▪ OraDefaultFormAuthenNScheme ▪ OraDefaultI18NformAuthenNScheme Note: The 'Challenge Redirect' parameter is added when the authentication scheme is created. The 'Challenge Redirect' parameter of an existing authentication scheme is not updated.
configOIMPwdPolicy	Creates the Oracle Identity Manager (OIM) password policy to automate integration with Oracle Access Manager. Also, the corresponding authentication scheme used by the policy is enabled to check password policies. See Also: "OIM Integration-Related Parameters and Values" on page 17-25.

Table 17-5 (Cont.) OAMCfgTool CREATE Mode Parameters and Values

Parameters	CREATE Mode Values
OimOhsHostPort	<p>Required when integrating Oracle Identity Manager (OIM) with Oracle Access Manager and an authentication WebGate and resource WebGate.</p> <p>See Also: "OIM Integration-Related Parameters and Values" on page 17-25.</p> <p>Not required without an authenticating WebGate. In this case, Oracle Identity Manager (OIM) password policy (OraOIMDefPasswdPolicy) automates integration with Oracle Access Manager and the corresponding authentication scheme used by the policy is enabled to check password policies. Default values are used for the password policy-related parameters with the value in OimOhsHostPort prepended to these. For example:</p> <pre data-bbox="537 541 1354 709"> -OimLostPwdRedirectUrl (Lost Password Redirect URL): <OimOHSHostPort>/admin/faces/pages/forgotpwd.jspx -OimPwdRedirectUrl (Password Change Redirect URL): <OimOHSHostPort>/admin/faces/pages/pwdmgmt.jspx?backUrl=%RESOURCE% -OimLockoutRedirectUrl (Account Lockout Redirect URL): <OimOHSHostPort>/ApplicationLockoutURI </pre> <p>OimOhsHostPort parameter is applicable only if the -configOimPwdPolicy flag is present.</p> <p>See Also: "OIM Integration-Related Parameters and Values" on page 17-25.</p>
logouturi	<p>Facilitates configuration of LogoutRedirectUrl on the Resource WebGate by pointing to the URL location on the Authenticating WebGate where the perl script for logout is configured.</p> <p>The value of logouturi parameter must be a URI. The WebGate LogoutRedirectUrl parameter is configured using the authenticating_wg_url and logouturi parameters:</p> <pre data-bbox="537 1024 1062 1050">http://<awghost>:<awgport>/cgi-bin/logout.pl</pre> <p>LogoutRedirectUrl http://myhost.us.myco.com:7777/cgi-bin/logout.pl.</p> <p>Note: Do not configure the LogoutRedirectUrl parameter on the authenticating WebGate itself. Instead, leave the LogoutRedirectUrl blank on the authenticating WebGate.</p> <p>To configure the logout URI when you create an application domain and provision a fresh WebGate:</p> <pre data-bbox="537 1262 1435 1461"> \$ (echo ldapUserPwd; echo appAgentPwd; echo OAMModePwd; echo TestUserPwd) java -jar oamcfgtool.jar app_domain=app_domain protected_uris="/protUri" ldap_host=<ldap-host> ldap_port=3899 ldap_userdn="cn=Directory Manager" oam_aaa_host=<aaa_host> oam_aaa_port=7054 oam_aaa_mode=simple ldap_ base="o=company,c=us" oam_aaa_passphrase=welcome1 authenticating_wg_ url=http://myhost.us.myco.com:7777 -logouturi=/cgi-bin/logout.pl -noprompt </pre> <p>Note: To use an existing WebGate, use the webgate_id parameter as described next.</p>

Table 17–5 (Cont.) OAMCfGTool CREATE Mode Parameters and Values

Parameters	CREATE Mode Values
webgate_id	<p>Specifies the name of the existing WebGate for which "LogoutRedirectUrl" is not yet configured.</p> <p>Notes: The WebGate profile is created only if the corresponding host identifier does not already exist in Oracle Access Manager. Further:</p> <ul style="list-style-type: none"> ■ If you do not provide <code>webgate_id</code>, the profile is created with the name specified with the value of <code>web_domain</code>, appended with <code>_AG</code> (<code>web_domain_AG</code>). ■ If <code>web_domain</code> is not specified, then <code>app_domain</code> is used to create the name of the profile, appended with <code>_AG</code> (<code>app_domain_AG</code>). <p>Following is a sample command using <code>webgate_id</code>:</p> <pre>\$ (echo ldapUserPwd; echo appAgentPwd; echo OAModePwd; echo TestUserPwd) java -jar oamcfgtool.jar app_domain=myapp webgate_id=MyWebgate protected_uris="/protUri" ldap_host=<ldap-host> ldap_port=3899 ldap_userdn="cn=Directory Manager" oam_aaa_host=<aaa_host> oam_aaa_port=7054 oam_aaa_mode=simple ldap_ base="o=company,c=us" oam_aaa_passphrase=welcome1 authenticating_wg_ url=http://myhost.us.myc.com:7777 -logouturi=/cgi-bin/logout.pl -noprompt</pre>
hostname_variations	<p>Enables you to add values to the Hostname Variations section of the Host Identifier in Oracle Access Manager.</p> <p>To configure virtual hosts for Apache-based Web servers (including OHS), include this parameter as follows:</p> <pre>java -jar oamcfgtool.jar app_domain=<app domain> web_domain=<hostid1> ... hostname_variations=vhost1,vhost2</pre> <p>Note:</p> <ul style="list-style-type: none"> ■ If a host identifier specified with <code>web_domain</code> parameter does not exist in Oracle Access Manager, it is created and values of <code>hostname_variations</code> are added to the hostname variations section of this host identifier. ■ If the specified host identifier already exists in Oracle Access Manager, values of <code>hostname_variations</code> are appended to the existing set of hostname variations for the host identifier. ■ If the WebGate profile identified by the <code>webgate_id</code> parameter (or the <code>web-domain</code> parameter or the <code>app_domain</code> parameter) does not exist, it is created and its "preferred http host" field is set to 'SERVER_NAME: the <code>preferred_http_host</code> parameter is not required in this case. <p>To configure virtual hosts for non-Apache-based Web servers, include the parameter, <code>preferred_http_host</code> as described next.</p>

Table 17-5 (Cont.) OAMCfGTool CREATE Mode Parameters and Values

Parameters	CREATE Mode Values
preferred_http_host	<p>Makes configurable the Preferred Http Host field of the WebGate profile.</p> <p>To configure virtual hosts for non-Apache-based Web servers, include this parameter, with a value of HOST_HTTP_HEADER, as follows:</p> <pre>java -jar oamcfgtool.jar app_domain=<app domain> web_domain=<hostid1> ... hostname_variations=vhost1,vhost2 preferred_http_host=HOST_HTTP_HEADER</pre> <p>You can simply add multiple hostname variations to a host identifier using the <code>hostname_variations</code> and <code>preferred_http_host</code> parameters as follows:</p> <pre>java -jar oamcfgtool.jar app_domain=<app domain> web_domain=<hostid1> ... hostname_variations=hostname1,hostname2 preferred_http_host=SOME_ HOSTNAME_VARIATION_VALUE</pre> <p>The virtual environment notes apply. Additionally, if the WebGate profile is being created, then you can set the preferred http host field of the profile to any value from the hostname variations</p> <p>Generally, you do not need additional hostname variations when creating a host identifier in a non-virtual host environment. OAMCfGTool adds a default value to the preferred http host field of the WebGate profile and to the hostname variation section of the host identifier being created.</p>

Table 17-5 (Cont.) OAMCfGTool CREATE Mode Parameters and Values

Parameters	CREATE Mode Values
default_authn_scheme	<p>Configures the default authentication scheme for a policy domain. You must pass the authentication scheme name as displayed in the Access System Console.</p> <p>OAMCfGTool always provisions the following authentication schemes:</p> <ul style="list-style-type: none"> ■ OraDefaultBasicAuthNScheme: The default Basic authentication scheme ■ OraDefaultFormAuthNScheme: The default Form authentication scheme ■ OraDefaultI18NFormAuthNScheme: The default i18n authentication scheme ■ OraDefaultAnonAuthNScheme: The default Anonymous Authentication scheme <p>The first time you run the tool in a new deployment, the schemes in the previous list are created.</p> <p>The authentication scheme specified as part of the "default_authn_scheme" parameter is used to configure the Default Authentication Rule section of the Policy Domain being configured.</p> <p>With the OAM URIs file, you can configure the authentication scheme for a protected policy (policies that are specified after the key word "protected_uris" for the Policy Domain. You must pass the Authentication Scheme name in the URIs file in the following format (the policy name and authentication scheme name must be separated by a tab character):</p> <p><i><Policy Name> 'tab' <Authentication Scheme Name>.</i></p> <p>Following is an example of entries in a URIs file (for more information, see the uris_file parameter earlier in this table):</p> <pre>#----- protected_uris protected policy1 Basic Over LDAP /protected1 public1/mystuff.html protected policy2 OraDefaultFormAuthNScheme /protected2/public2/prot2 ../../{*.js,*.png,*.gif} protected policy3 Client Certificate /protected2/public2/prot2/../../{*.js,*.png,*.gif} #-----</pre> <p>The previous entries in a URIs file produce the following named policies:</p> <ul style="list-style-type: none"> ■ protected_policy1 is configured to use the Basic Over LDAP scheme ■ protected_policy2 is configured to use the OraDefaultFormAuthNScheme scheme ■ protected_policy3 is configured to use the Client Certificate scheme

Table 17–5 (Cont.) OAMCfGTool CREATE Mode Parameters and Values

Parameters	CREATE Mode Values
max_oam_connections	Supports high availability and multiple Access Servers by specifying the maximum number of connections ('Maximum Connections') for the WebGate profile being created.
primary_oam_servers	<p>Supports high availability and multiple Access Servers by configuring the WebGate profile with more than one primary Access Server. The format of this parameter is:</p> <ul style="list-style-type: none"> ■ Colons join each Access Server name with the number of connections to the WebGate. For example: primary_oam_servers="aaaid1:3". If no numeric value is specified, the default is 1. ■ Comma-separated list of Access Server names and the number of connections to the WebGate. For example: primary_oam_servers="aaaid1:3,aaaid2:1,aaaid3,aaaid4:2" <p>Notes:</p> <ul style="list-style-type: none"> ■ Access Server IDs must exist within OAM and must be unique (no duplicates and not present in both primary and secondary values). ■ Transport Security mode of WebGate and Access Servers must match. ■ The Access Management Service mode of WebGate and Access Server must match.
secondary_oam_servers	<p>Supports high availability and multiple Access Servers by configuring the WebGate profile with more than one secondary Access Server. The format of this parameter is:</p> <ul style="list-style-type: none"> ■ Colons join each Access Server name with the number of connections to the WebGate. For example: secondary_oam_servers="aaaid1:3". If no numeric value is specified, the default is 1. ■ Comma-separated list of Access Server names and the number of connections to the WebGate. For example: secondary_oam_servers="aaaid1:3,aaaid2:1,aaaid3,aaaid4:2" <p>Notes:</p> <ul style="list-style-type: none"> ■ Access Server IDs must exist within OAM and must be unique (no duplicates and not present in both primary and secondary values). ■ Transport Security mode of WebGate and Access Servers must match. ■ The Access Management Service mode of WebGate and Access Server must match.

17.3.2.1.1 OIM Integration-Related Parameters and Values Table 17–6 identifies OIM integration-related parameters and values for OAMCfGTool.

See Also: The section on integrating Oracle Access Manager 10g with Oracle Identity Manager 11g in the *Oracle Fusion Middleware Enterprise Deployment Guide for Oracle Identity Management*

Table 17–6 Additional OIM Integration-Related Parameters and Values

Parameter	Description
configOIMPwdPolicy	<p>Creates the Oracle Identity Manager (OIM) password policy (OraOIMDefPasswdPolicy) to automate integration with Oracle Access Manager. Additionally, the corresponding authentication scheme used by the policy is enabled to check password policies.</p> <p>For example, if the policy is used with the default authentication scheme (OraDefaultFormAuthnScheme), then the scheme's "Validate_Password" plug-in is updated to include 'obReadPasswdMode="LDAP",obWritePasswdMode="LDAP"'. </p> <p>Note: Use default values for password-related parameters in Identity System Console, prepended with the value specified with OimOhsHostPort.</p> <p>When configOIMPwdPolicy is used, ensure that you do not have the default OIM password policy created using the tool previously and do not pass any of the following parameters:</p> <p>When configOIMPwdPolicy is used, ensure that you do not have the default OIM password policy created using the tool previously and do not pass any of the following parameters:</p>
OimOhsHostPort	<p>Required when integrating Oracle Identity Manager (OIM) with Oracle Access Manager and an authentication WebGate and resource WebGate.</p> <p>Not required without an authenticating WebGate. In this case, Oracle Identity Manager (OIM) password policy (OraOIMDefPasswdPolicy) automates integration with Oracle Access Manager and the corresponding authentication scheme used by the policy is enabled to check password policies. Default values are used for the password policy-related parameters with the value in OimOhsHostPort prepended to these. For example:</p> <pre>-OimLostPwdRedirectUrl (Lost Password Redirect URL): <OimOHSHostPort>/admin/faces/pages/forgotpwd.jspx -OimPwdRedirectUrl (Password Change Redirect URL): <OimOHSHostPort>/admin/faces/pages/pwdmgmt.jspx?backUrl =%RESOURCE% -OimLockoutRedirectUrl (Account Lockout Redirect URL): <OimOHSHostPort>/ApplicationLockoutURI</pre> <p>OimOhsHostPort parameter is applicable only if the -configOimPwdPolicy flag is present.</p>
OimPwdRedirectUrl	Required for configOIMPwdPolicy. Configures the Password Change Redirect URL parameter in Oracle Access Manager.
OimLockoutRedirectUrl	Required for configOIMPwdPolicy. Configures the Custom Account Lockout Redirect URL parameter in Oracle Access Manager.
OimLostPwdRedirectUrl	Required for configOIMPwdPolicy. Configures the Lost Password Redirect URL parameter in Oracle Access Manager.

Note: This is a one time setup requirement. If the OraOIMDefPasswdPolicy policy already exists, it is not created anew. You must restart the Identity and Access Servers after this operation. See [Example 17–2](#).

Example 17–2 OIM Integration-Related Parameter Usage

```
$ (echo ldapUserPwd; echo appAgentPwd; echo OAMModePwd; echo TestUserPwd)
java -jar oamcfgtool.jar app_domain=app_domain protected_uris="/protUri"
ldap_host=<ldap-host> ldap_port=3899 ldap_userdn="cn=Directory Manager"
oam_aaa_host=<aaa_host> oam_aaa_port=7054 oam_aaa_mode=simple ldap_
base="o=company,c=us" oam_aaa_passphrase=welcome1 authenticating_wg_
url=http://myhost.us.myco.com:7777 -configOIMPwdPolicy
OimPwdRedirectUrl="http://oimredirectutl.com
OimLockoutRedirectUrl="http://oimlockouturl.com"
OimLostPwdRedirectUrl="http://oimLostpwdurl.com"
-noprompt
```

17.3.2.2 Validate Mode Parameters and Values

Master or Delegated Access Administrators can check Oracle Access Manager directly to validate policy domain and WebGate profile setup.

Note: You cannot use OAMCfGTool mode to validate AccessGate profile creation.

Using OAMCfGTool in VALIDATE mode, you can ensure that the policy domain for single sign-on configuration is correct. In this case, a set of requests are sent automatically to protected resources.

Table 17–7 provides both required and optional OAMCfGTool parameters and values for VALIDATE mode.

Table 17–7 OAMCfGTool VALIDATE Mode Parameters and Values

VALIDATE Mode Parameters	VALIDATE Mode Values for Required Parameters
Required Parameters	Values
app_domain	Name of the Oracle Access Manager policy domain that was created to protect the Application.
ldap_host	DNS name of the computer hosting the LDAP directory server for Oracle Access Manager.
ldap_port	Port of the LDAP directory server.
ldap_userdn	The valid DN of the LDAP administrative user, entered as a quoted string. In Oracle Access Manager this is known as the Root DN or Bind DN.
ldap_userpassword	Password of the LDAP administrative user. Passwords appear in clear text but are not captured in a log file. See Also: noprompt in this table.
ldap_base	Base from which all LDAP searches are done. In Oracle Access Manager this is known as the search base or configuration base. For example: dc=company,c=us.
oam_aaa_host	DNS name of the computer hosting the Access Server.
oam_aaa_port	Listening port on the Access Server host.
test_username	User name to be used for policy validation.
test_userpassword	User password to be used for policy validation. Passwords appear in clear text but are not captured in a log file. See Also: noprompt in this table.

Table 17-7 (Cont.) OAMCfGTool VALIDATE Mode Parameters and Values

VALIDATE Mode Parameters	VALIDATE Mode Values for Required Parameters
noprompt	Enables OAMCfGTool to read passwords from System.in to ensure safe passage. Passwords can be passed from a shell using an echo command and a semi-colon as a separator. ConfigTool expects four passwords: Ldap user, App agent, OAM mode and Test user: See Also: noprompt in Table 17-5 .
Optional Parameters	Values
web_domain	Host identifier
ldap_base	Base from which all LDAP searches are done. In Oracle Access Manager this is known as the search base or configuration base. For example: dc=company,c=us.
oam_aaa_mode	Transport security mode of the accessible Access Server: OPEN, SIMPLE, or CERT. Default presumes OPEN.
oam_aaa_passphrase	Passphrase required for SIMPLE mode transport security mode only. Your entry appears in clear text. However, it is not captured in a log file.
log_file	Name of the OAMCfGTool log file. Output to the screen is the default.
log_level	Level for OAMCfGTool logging: ALL, SEVERE, WARNING, INFO, CONFIG, FINE, FINER, FINEST, OFF (the default).
noprompt	Enables OAMCfGTool to read passwords from System.in to ensure safe passage. Passwords can be passed from a shell using an echo command and a semi-colon as a separator. OAMCfGTool expects four passwords: LDAP user, Application agent, OAM mode and Test user. See Also Table 17-5 .

17.3.2.3 Delete Mode Parameters and Values

Using OAMCfGTool in DELETE mode, you can remove the provisioned policies, the web domain, WebGate registration, and authentication scheme.

[Table 17-8](#) provides both required and optional OAMCfGTool parameters and values for DELETE mode.

Table 17-8 OAMCfGTool DELETE Mode Parameters

DELETE Mode Parameters	DELETE Mode Values for Required Parameters
ldap_host	DNS name of the computer hosting the LDAP directory server for Oracle Access Manager.
ldap_port	Port of the LDAP directory server.
ldap_userdn	The valid DN of the LDAP administrative user, entered as a quoted string. In Oracle Access Manager this is known as the Root DN or Bind DN.
ldap_userpassword	Password of the LDAP administrative user. Passwords appear in clear text but are not captured in a log file. See Also: -noprompt in Table 17-5 .
oam_aaa_host	DNS name of the computer hosting the Access Server.
oam_aaa_port	Listening port on the Access Server host.
Optional Parameters	Values
app_domain	To delete the entire application domain, specify only app_domain with no URI-related parameters.
web_domain	web_domain= <i>existing_host_Identifier</i> To delete the host identifier identified by this parameter and the WebGate registration. See Also: Table 17-5 .

Table 17–8 (Cont.) OAMCfTool DELETE Mode Parameters

DELETE Mode Parameters	DELETE Mode Values for Required Parameters
protected_uris	URIs for the protected application in a comma separated list (with or without spaces): /myapp/login, for example. Deletes one or more protected URIs from an application domain. See Also: The uris_file parameter in this table.
public_uris	Deletes one or more public URIs from an application domain. See Also: The uris_file parameter in this table.
uris_file	The full path to a file containing any number of protected or public URIs and eliminates the need to use the protected_uris or public_uris parameters. Ensure that the file uses the following syntax and format. See Also: Table 17–5 .
authn_scheme	The name of the authentication scheme to delete: OraDefAuthSchemes, OraDefaultAWGFormAuthNScheme, OraDefault18NFormAuthNScheme. To delete all three, specify OraDefAuthSchemes: You can include the following options: -noconfirm With this parameter there is no prompt for confirmation before deleting.
noprompt	Enables OAMCfTool to read passwords from System.in to ensure safe passage. Passwords can be passed from a shell using an echo command and a semi-colon as a separator. OAMCfTool expects four passwords: LDAP user, Application agent, OAM mode and Test user. See Also Table 17–5 .

17.3.3 Sample Policy Domain and AccessGate Profile Created with OAMCfTool

This topic describes and illustrates the results of running OAMCfTool when viewed in Oracle Access Manager:

- [My Policy Domains](#)
- [Policy Domain, General Tab](#)
- [Policy Domain, Resources Tab](#)
- [Policy Domain, Authorization Rules Tab](#)
- [Policy Domain, Default Rules Tab](#)
- [Policy Domain, Policies Tab](#)
- [Policy Domain, Delegated Access Admins Tab](#)
- [Host Identifiers](#)
- [AccessGate Profile](#)

My Policy Domains

Name: *app_domain* value specified with OAMCfTool.

Policy Domain, General Tab

[Figure 17–1](#) illustrates the General tab in a sample policy domain created with OAMCfTool. The Description is provided automatically.

Name: *app_domain* value specified with OAMCfTool

Description: includes the *app_domain* value created by *user@hostname* ...

Note: For descriptions only, the Java API retrieves the current *user* from the operative platform and the name of the computer host: *user@hostname*.

Figure 17-1 Sample OAMCfGTool Policy Domain General Tab



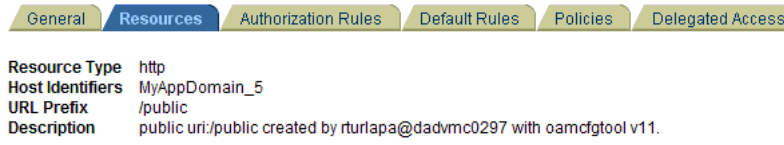
Surrounding text describes this graphic.

Policy Domain, Resources Tab

Figure 17-2 illustrates the Resources tab in a sample policy domain created with OAMCfGTool. The http resource type is the default. The host identifier and URL prefixes are derived from OAMCfGTool parameters and the values you enter. The Description is provided automatically.

Host Identifier: *app_domain* value
 URL Prefix: *protected_uris* values

Figure 17-2 Sample OAMCfGTool Policy Domain Resources Tab

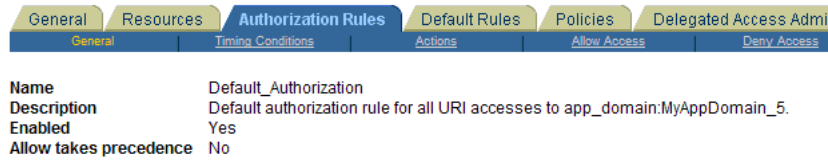


Surrounding text describes this graphic.

Policy Domain, Authorization Rules Tab

Figure 17-3 illustrates the Authorization Rules tab in a sample policy domain created with OAMCfGTool. Details found on sub tabs follow the figure. Authorization rules are automatically configured for the policy domain when you use OAMCfGTool.

Figure 17-3 Sample OAMCfGTool Policy Domain Authorization Rules Tab



Surrounding text describes this graphic.

Timing Conditions: There are no timing conditions defined. This rule is always valid.
Actions: There are no actions defined.

Allow Access: Role: Anyone
Deny Access: No one is denied access.

Policy Domain, Default Rules Tab

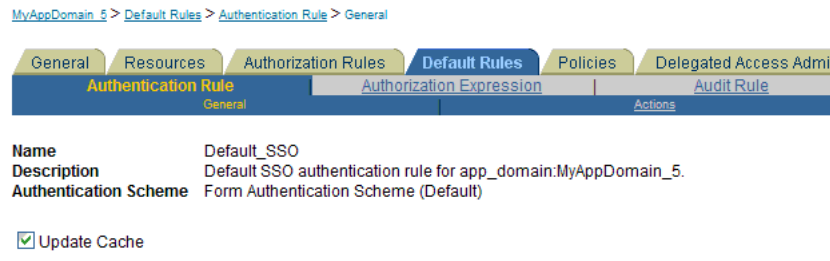
Figure 17-4 illustrates the Default Rules tab in a sample policy domain created with OAMCfGTool. All values are configured automatically for the policy domain; details on sub tabs follow the figure.

Authentication Rule

General, see Figure 17-4.

Actions: There are no actions defined.

Figure 17-4 Sample OAMCfGTool Policy Domain Default Rules Tab



Surrounding text describes this graphic.

Authorization Expression

Authorization Expression: Default_Authorization

Duplicate Actions: No policy defined for this Authorization Expression. The Access System level default policy for dealing with duplicate action headers are employed.

Actions

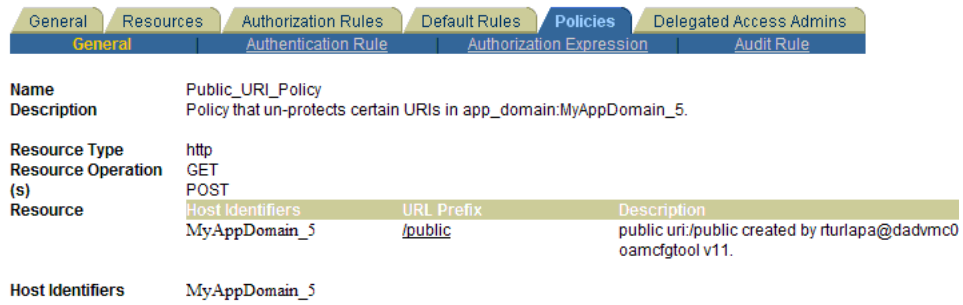
Authorization Success

Return	Type	Name	Attribute
	HeaderVar	REMOTE_USER	uid
	HeaderVar	OAM_REMOTE_USER	uid

Policy Domain, Policies Tab

Figure 17-5 illustrates the Policies tab, General sub tab, in a sample policy domain created using parameters and values that you specify with OAMCfGTool. The host identifiers are based on your app_domain value. Details on other sub tabs follow the figure.

Figure 17–5 Sample OAMCfGTool Policy Domain Policies Tab



Surrounding text describes this graphic.

Authentication Rule

General

Name: Anonymous

Description: Authentication scheme allows un-authenticated access to some URIs

Authentication Scheme: Anonymous Authentication (Default)

Actions: There are no actions defined.

Authorization Expression

There is no Authorization Expression defined.

Audit Rule

There is no Master Audit Rule defined.

If you would like to add an auditing rule to this Policy, please contact your Access System Administrator.

Policy Domain, Delegated Access Admins Tab

Figure 17–6 illustrates the Delegated Access Admins tab in a sample policy domain created using OAMCfGTool. No parameters are specified with the tool to set up delegated rights for Master Web resource Admins.

Figure 17–6 OAMCfGTool Policy Domain Delegated Access Admins Tab



Surrounding text describes this graphic.

See Also: "Protecting Resources with Policy Domains" in the *Oracle Access Manager Access Administration Guide*.

Host Identifiers

You can find the Host Identifiers created with OAMCfGTool in the Access System Console, under the Access System Configuration tab.

Figure 17–7 illustrates a sample host identifiers created using OAMCfGTool. As described here, required parameters are derived from the value entered with OAMCfGTool *app_domain* parameter. A Description is provided by OAMCfGTool.

Figure 17–7 Sample OAMCfGTool Host Identifiers

Host identifier details

Name	MyAppDomain_5
Description	Host identifier for domain MyAppDomain_5 created by rturlapa@dadvmc0297 with oamcfgtool v11.
Hostname variations	MyAppDomain_5

Surrounding text describes this graphic.

AccessGate Profile

Figure 17–8 illustrates a sample AccessGate profile created using OAMCfGTool when the *web_domain* parameter is omitted. The profile is in the Access System Console. As described here, required profile parameters are derived from values entered with OAMCfGTool. Other profile parameters use default values. A Description is provided by OAMCfGTool.

Name: *app_domain* value *_AG*
 Hostname: *app_domain* value
 Access Gate Password: *app_agent_password* value

ASDK Client

Access Management Service: On

Web Server Client

Primary HTTP Cookie Domain: *cookie_domain* value
 Preferred HTTP Host: *app_domain* value

Figure 17–8 Sample OAMCfGTool AccessGate Profile**Details for AccessGate**

AccessGate Name	MyAppDomain_5_AG
Description	AccessGate for hostid:MyAppDomain_5 created by rturlapa@dadvmc0297 with oamcfgtool
State	Enabled
Hostname	MyAppDomain_5
Port	<No Port Specified>
Access Gate Password	<Not Displayed>
Debug	Off
Maximum user session time (seconds)	3600
Idle Session Time (seconds)	3600
Maximum Connections	1
Transport Security	Open
IPValidation	On
IPValidationException	
Maximum Client Session Time (hours)	24
Failover threshold	1
Access server timeout threshold	
Sleep For (seconds)	60
Maximum elements in cache	100000
Cache timeout (seconds)	1800
Impersonation username	
Impersonation password	<Not Displayed>
ASDK Client	
Access Management Service	On
Web Server Client	
Primary HTTP Cookie Domain	.us.oracle.com
Preferred HTTP Host	MyAppDomain_5
Deny On Not Protected	Off
CachePragmaHeader	no-cache
CacheControlHeader	no-cache
LogOutURLs	
User Defined Parameters	
Parameters	Values

Surrounding text describes this graphic.

17.3.4 Known Issues: JAR Files and OAMCfGTool

Table 17–9 identifies known issues with this release. For more information about the tool, parameters, and values, see "Introduction to OAMCfGTool" on page 17-15.

Table 17–9 OAMCfGTool Known Issues

Bug Number	Description
n/a	The location where you obtain Oracle Access Manager Authentication Provider and OAMCfGTool JAR files when you do not have an Oracle Fusion Middleware application installed could change. If the location is different than the one stated in this chapter, see the Release Notes for the latest information.
8362080	OAMCfGTool provides Create, Validate, and Delete modes. It does not provide an Overwrite option.

Table 17–9 (Cont.) OAMCfgTool Known Issues

Bug Number	Description
8362039	<p>OAMCfgTool does not provide explicit options to specify the Web Tier host and port. Instead, without <code>web_domain</code> specified the <code>app_domain</code> value specifies the WebGate name, host, and Preferred HTTP Host. For example:</p> <ul style="list-style-type: none"> ▪ <code>app_domain=ABC</code> (without <code>web_domain</code> specified) ▪ AccessGate Name: <code>ABC_AG</code> ▪ Hostname: <code>ABC</code> ▪ Port: Not specified ▪ Preferred HTTP Host: <code>ABC</code>
n/a	<p>With OAMCfgTool, if <code>web_domain</code> parameter is included in the command line, you must provide a WebGate password. Otherwise, the command can fail.</p> <p>The <code>app_agent_password</code> parameter accepts as the password whatever follows the equal sign, <code>=</code>. For instance, if you enter <code>app_agent_password=</code> and then enter a space character and <code>web_domain=value</code>, the <code>app_agent_password</code> is presumed to be a space character followed by <code>web_domain</code>.</p>
n/a	<p>SSL-enabled communication with the directory server is not supported by OAMCfgTool.</p>

17.4 Configuring OAM Identity Assertion for SSO with Oracle Access Manager 10g

This section describes the unique steps needed to configure Oracle Access Manager Identity Assertion for Single Sign-On.

Prerequisites

Unless explicitly noted for the Authenticator or Oracle Web Services Manager, all tasks described in "[Installing and Setting Up Authentication Providers for OAM 10g](#)" on page 17-1 should be performed, including:

- [Installing Components and Files for Authentication Providers and OAM 10g](#)

Note: If you are implementing:

- OAM 11g: Provision WebGates and security policies using the remote registration tool as described in "[Session Token: Provisioning an OAM Agent with Oracle Access Manager 11g](#)".
 - OAM 10g: Add WebGate profiles and policies with OAMCfgTool as described in the following Task 3.
-
-

To configure Oracle Access Manager Identity Asserter for single sign-on with your application, perform the tasks as described in the following task overview.

Task overview: Deploying and configuring the Oracle Access Manager Identity Asserter for single sign-on includes

1. Ensuring that all prerequisite tasks have been performed
2. [Establishing Trust with Oracle WebLogic Server](#)
3. [Configuring the Authentication Scheme for the Identity Asserter](#)
4. [Configuring Providers in the WebLogic Domain](#)
5. [Setting Up the Login Form for the Identity Asserter and OAM 10g](#)

6. [Testing Identity Assertion for SSO with OAM 10g](#)
7. [Configuring Global Logout for Oracle Access Manager 10g and 10g WebGates](#)

17.4.1 Establishing Trust with Oracle WebLogic Server

The following topics explain the tasks you must perform to set up the application for single sign-on with the Oracle Access Manager Identity Asserter:

Note: This task is the same for both OAM 11g and OAM 10g.

- [Setting Up the Application Authentication Method for SSO](#)
- [Confirming mod_weblogic for Oracle Access Manager Identity Asserter](#)
- [Establishing Trust between Oracle WebLogic Server and Other Entities](#)

17.4.1.1 Setting Up the Application Authentication Method for SSO

This topic describes how to create the application authentication method for Oracle Access Manager Identity Assertion.

See Also: *Oracle Fusion Middleware Deploying Applications to Oracle WebLogic Server*

When you use the Oracle Access Manager Identity Asserter, all `web.xml` files in the application EAR file must specify `CLIENT-CERT` in the element `auth-method` for the appropriate realm.

The `auth-method` can use `BASIC`, `FORM`, or `CLIENT-CERT` values. While these look like similar values in Oracle Access Manager, the `auth-method` specified in `web.xml` files are used by Oracle WebLogic Server (not Oracle Access Manager).

Note: You can specify `CLIENT-CERT`, `FORM` if you are also planning to access the applications directly over WebLogic and want the WebLogic authentication scheme to be invoked.

To specify authentication in `web.xml` for the Identity Asserter and OAM 10g

1. Locate the `web.xml` file in the application EAR file:

```
your_app/WEB-INF/web.xml
```

2. Locate the `auth-method` in `login-config` and enter `CLIENT-CERT`.

```
<login-config>
  <auth-method>CLIENT-CERT</auth-method>
</login-config>
```

3. Save the file.
4. Redeploy and restart the application.
5. Repeat for each `web.xml` file in the application EAR file.
6. Proceed to "[Confirming mod_weblogic for Oracle Access Manager Identity Asserter](#)".

17.4.1.2 Confirming mod_weblogic for Oracle Access Manager Identity Asserter

Oracle HTTP Server includes the mod_weblogic plug-in module (mod_wl_ohs.so in 11g) which is already enabled. You can perform the following procedure to confirm this or skip this procedure.

With Oracle HTTP Server 11g, the mod_weblogic configuration is present in mod_wl_ohs.conf by default, and the path of this file is included in httpd.conf. If the mod_weblogic configuration is not present then you must edit httpd.conf.

To configure mod_weblogic for the Identity Asserter and OAM 10g

1. Locate httpd.conf. For example:

```
ORACLE_INSTANCE/config/OHS/<ohs_name>/httpd.conf
```

2. Confirm that the following statement is in the file with appropriate values for your deployment (add or uncomment this, if needed):

```
<IfModule mod_weblogic.c>
  WebLogicHost yourHost.yourDomain.com
  WebLogicPort yourWlsPortNumber
</IfModule>

<Location http://request-uri-pattern>
  SetHandler weblogic-handler
</Location>
```

3. Save the file.
4. Proceed to ["Establishing Trust between Oracle WebLogic Server and Other Entities"](#).

17.4.1.3 Establishing Trust between Oracle WebLogic Server and Other Entities

The Oracle WebLogic Connection Filtering mechanism must be configured for creating access control lists and for accepting requests from only the hosts where Oracle HTTP Server and the front-end Web server are running.

Note: This topic is the same whether you are using OSSO or Oracle Access Manager.

A *network connection* filter is a component that controls the access to network level resources. It can be used to protect resources of individual servers, server clusters, or an entire internal network. For example, a filter can deny non-SSL connections originating outside of a corporate network. A network connection filter functions like a firewall since it can be configured to filter protocols, IP addresses, or DNS node names. It is typically used to establish trust between Oracle WebLogic Server and foreign entities.

To configure a connection filter to allow requests from only mod_weblogic and the host where OHS 11g is running, perform the procedure here.

Note: This chapter uses the generic name of the WebLogic Server plug-in for Apache: mod_weblogic. For Oracle HTTP Server 11g, the name of this plug-in is mod_wl_ohs; the actual binary name is mod_wl_ohs.so. Examples show exact syntax for implementation.

WebLogic Server provides a default connection filter: `weblogic.security.net.ConnectionFilterImpl`. This filter accepts all incoming connections and also provides static factory methods that allow the server to obtain the current connection filter. To configure this connection filter to deny access, simply enter the connection filters rules in the WebLogic Server Administration Console.

You can also use a custom connection filter by implementing the classes in the `weblogic.security.net` package. Like the default connection filter, custom connection filters are configured in the WebLogic Server Administration Console.

Connection Filter Rules: The format of filter rules differ depending on whether you are using a filter file to enter the filter rules or you enter the filter rules in the Administration Console. When entering the filter rules on the Administration Console, enter them in the following format:

```
targetAddress localAddress localPort action protocols
```

Table 17–10 provides a description of each parameter in a connection filter.

Table 17–10 Connection Filter Rules

Parameter	Description
target	Specifies one or more systems to filter
localAddress	Defines the host address of the WebLogic Server instance. (If you specify an asterisk (*), the match returns all local IP addresses.)
localPort	Defines the port on which the WebLogic Server instance is listening. (If you specify an asterisk, the match returns all available ports on the server.)
action	Specifies the action to perform. This value must be allow or deny
protocols	Is the list of protocol names to match. The following protocols may be specified: http, https, t3, t3s, giop, giops, dcom, ftp, ldap. If no protocol is defined, all protocols match a rule.

The Connection Logger Enabled attribute logs successful connections and connection data in the server. This information can be used to debug problems relating to server connections.

See Also: "Configuring Security in a WebLogic Domain" in *Oracle Fusion Middleware Securing Oracle WebLogic Server*

To configure a connection filter to allow requests from the host of the 11g Oracle HTTP Server

1. Log in to the Oracle WebLogic Administration Console.
2. Click Domain under Domain Configurations.
3. Click the Security tab, click the Filter tab.
4. Click the Connection Logger Enabled attribute to enable the logging of accepted messages for use when debugging problems relating to server connections.
5. Specify the connection filter to be used in the domain:
 - Default Connection Filter: In the Connection Filter attribute field, specify `weblogic.security.net.ConnectionFilterImpl`.
 - Custom Connection Filter: In the Connection Filter attribute field, specify the class that implements the network connection filter, which should also be specified in the CLASSPATH for Oracle WebLogic Server.

6. Enter the appropriate syntax for the connection filter rules.
7. Click Save.
8. Restart the Oracle WebLogic Server.
9. Proceed to "[Configuring the Authentication Scheme for the Identity Asserter](#)".

17.4.2 Configuring the Authentication Scheme for the Identity Asserter

This topic focuses on using OAMCfgTool for OAM 10g. If you are using OAM 11g, skip this topic and instead perform tasks in "[Session Token: Provisioning an OAM Agent with Oracle Access Manager 11g](#)".

After setting up your application, you must protect it with Oracle Access Manager. To help automate this task, Oracle provides the command-line tool: OAMCfgTool in the Fusion Middleware application-provided oamcfgtool.jar file for OAM 10g.

While you can perform steps manually in the Access System Console and Policy Manager, you can optionally use OAMCfgTool to setup and validate a form-based authentication scheme, a policy domain for the application, and Oracle Access Manager access policies required for Identity Assertion for single sign-on. Additionally, you can create a new WebGate profile in a fresh Web Tier or modify a WebGate profile in an existing Web Tier.

See Also: "[Introduction to OAMCfgTool](#)" on page 17-15

For more information, see "[Creating an Authentication Scheme, Policy Domain, and a WebGate Profile](#)".

17.4.2.1 Creating an Authentication Scheme, Policy Domain, and a WebGate Profile

This topic provides a procedure that you can use as a model when you run OAMCfgTool.

This example presumes a fresh Web Tier that requires a new WebGate profile. Therefore, the `web_domain=` parameter is omitted. A new profile is created and named with the `app_domain` value (appended with `_AG`).

The following procedure is only an example to illustrate how to use the tool. Values for your environment will be different.

Note: If you have an Oracle Fusion Middleware application installed you already have the OAMCfgTool. In this case, skip Step 1.

To create a form authentication scheme, policy domain, and access polices with OAMCfgTool

1. **No Oracle Fusion Middleware Application:** Obtain the OAMCfgTool if you have no Oracle Fusion Middleware application.

- a. Log in to Oracle Technology Network at:

http://www.oracle.com/technology/software/products/middleware/html/docs/111110_fm.html

- b. Locate the OAMCfgTool ZIP file with Access Manager Core Components (10.1.4.3.0):

`oamcfgtool<version>.zip`

- c. Extract and copy oamcfgtool.jar to the computer hosting WebGate.
2. Confirm that JDK 1.6 (or the latest version) is installed and configured.
3. Log in to the computer that is hosting the application to protect, change to the file system directory containing OAMCfGTool.

Note:

- Fresh Web Tier: Omit web_domain parameter to create and associate a new a profile. Include the cookie_domain parameter.
 - Existing Web Tier: Include web_domain parameter with the value of an existing host identifier.
-
-

4. Create a WebGate Profile, Authentication Scheme, and Policy Domain: Run the following command using values for your environment as described in [Table 17-5](#). For example:

```
(echo ldappwd | java -jar oamcfgtool.jar
mode=CREATE app_domain=IASSO_App1
protected_uris=/myapp/login
cookie_domain=<preferred_http_cookie_domain>
ldap_host=wxyz
ldap_port=6633
ldap_userdn=orcladmin
oam_aaa_host=abcd
oam_aaa_port=7789
oam_aaa_mode=cert
log_file=OAMCfG_date.log
log_level=INFO
output_ldif_file=<LDIF_filename>
-noprompt
```

5. Review the information provided by the tool. For example, the parameters and values in Step 3 would provide the following information:

```
Processed input parameters
Initialized Global Configuration
Successfully completed the Create operation.
Operation Summary:
  Policy Domain : IASSO_App1
  Host Identifier: IASSO_App1
  Access Gate ID : IASSO_App1_AG
```

6. **Output LDIF Created:** Import the LDIF to write information to the directory server. Otherwise, skip this step.
7. **Validate:** Run OAMCfGTool to validate the policy domain that was created (see [Table 17-7](#)). For example:

```
(echo ldappwd | java -jar oamcfgtool.jar mode=VALIDATE app_domain=IASSO_App1
protected_uris=/myapp/login
ldap_host=wxyz
ldap_port=6633
ldap_userdn=orcladmin
oam_aaa_host=abcd
oam_aaa_port=7789
log_file=OAMCfG_date.log
log_level=INFO
test_username=gcf
```

```
test_userpassword=<test_userpassword>
-noprompt
```

8. **Fresh WebGate Profile/WebGate Not Installed:** Specify the same values when you install the WebGate as you specified when creating the profile (plus additional values to properly finish the installation).
9. **Fresh WebGate Profile with Installed WebGate:** Using output from the OAMCfgTool Create command, run the Oracle Access Manager configureWebgate tool to set up the installed WebGate. For example:

- a. Go to:

```
WebGate_install_dir\access\oblix\tools\configureWebGate
```

where *WebGate_install_dir* is the directory where WebGate is installed.

- b. Run the following command to configure the WebGate using values specified with OAMCfgTool and other values needed to finish the profile. For example:

```
configureWebGate -i WebGate_install_dir -t WebGate WebGate_Name -P
WebGate_password
-m <open|simple|cert>
-h Access_Server_Host_Name
-p Access_Server_Port
-a Access_Server_ID
-r Access_Server_Pass_Phrase (must be the same as the WebGate_password)
-Z Access_Server_Retry count
```

See Also: "Configuring AccessGates and WebGates" in the *Oracle Access Manager Access Administration Guide*

10. **Confirm Profile in the Access System Console:** Perform the following steps to view or modify the WebGate profile.

- a. Log in to the Access System Console as a Master or Delegated Access Administrator. For example:

```
http://hostname:port/access/oblix
```

hostname refers to computer that hosts the WebPass Web server; *port* refers to the HTTP port number of the WebPass Web server instance; */access/oblix* connects to the Access System Console.

- b. Click **Access System Configuration**, and then click **AccessGate Configuration**.
 - c. Click the All button to find all profiles (or select the search attribute and condition from the lists) and then click Go.
 - d. Click a WebGate's name to view its details.
 - e. Click Cancel to dismiss the page without changes, or click Modify to change values as described in the *Oracle Access Manager Access Administration Guide*.
11. Repeat Steps 3 through 8 for each application that you are protecting.
 12. Proceed to "[Configuring Providers in the WebLogic Domain](#)".

17.4.3 Configuring Providers in the WebLogic Domain

This topic is divided as follows:

- [About Oracle WebLogic Server Authentication and Identity Assertion Providers](#)

- [About the Oracle WebLogic Scripting Tool \(WLST\)](#)
- [Configuring Oracle WebLogic Server for a Web Application Using ADF Security, OAM SSO, and OPSS SSO](#)
- [Setting Up Providers for Oracle Access Manager Identity Assertion](#)

17.4.3.1 About Oracle WebLogic Server Authentication and Identity Assertion Providers

This topic introduces only a few types of Authentication Providers for a WebLogic security realm, if you are new to them.

Each WebLogic security realm must have one at least one Authentication Provider configured. The WebLogic Security Framework is designed to support multiple Authentication Providers (and thus multiple LoginModules) for multipart authentication. As a result, you can use multiple Authentication Providers as well as multiple types of Authentication Providers in a security realm. The Control Flag attribute determines how the LoginModule for each Authentication Provider is used in the authentication process.

Oracle WebLogic Server offers several types of Authentication and Identity Assertion providers including, among others:

- The default WebLogic Authentication Provider (Default Authenticator) allows you to manage users and groups in one place, the embedded WebLogic Server LDAP server. This Authenticator is used by the Oracle WebLogic Server to login administrative users.
- Identity Assertion providers use token-based authentication; the Oracle Access Manager Identity Asserter is one example.
- LDAP Authentication Providers store user and group information in an external LDAP server. They differ primarily in how they are configured by default to match typical directory schemas for their corresponding LDAP server.

Oracle WebLogic Server 10.3.1+ provides the OracleInternetDirectoryAuthenticator.

When you configure multiple Authentication Providers, use the JAAS Control Flag for each provider to control how the Authentication Providers are used in the login sequence. You can choose the following the JAAS Control Flag settings, among others:

- **REQUIRED**—The Authentication Provider is always called, and the user must always pass its authentication test. Regardless of whether authentication succeeds or fails, authentication still continues down the list of providers.
- **SUFFICIENT**—The user is not required to pass the authentication test of the Authentication Provider. If authentication succeeds, no subsequent Authentication Providers are executed. If authentication fails, authentication continues down the list of providers.
- **OPTIONAL**—The user is allowed to pass or fail the authentication test of this Authentication Provider. However, if all Authentication Providers configured in a security realm have the JAAS Control Flag set to **OPTIONAL**, the user must pass the authentication test of one of the configured providers.

When additional Authentication Providers are added to an existing security realm, the Control Flag is set to **OPTIONAL** by default. You might need to change the setting of the Control Flag and the order of providers so that each Authentication Provider works properly in the authentication sequence.

See Also: "Configuring Authentication Providers" in *Oracle Fusion Middleware Securing Oracle WebLogic Server* for a complete list of Authentication Providers and details about configuring the Oracle Internet Directory provider to match the LDAP schema for user and group attributes

17.4.3.2 About the Oracle WebLogic Scripting Tool (WLST)

This topic introduces WLST, if you are new to it.

You can add providers to a WebLogic domain using either the Oracle WebLogic Administration Console or Oracle WebLogic Scripting Tool (WLST) command-line tool.

WLST is a Jython-based command-line scripting environment that you can use to manage and monitor WebLogic Server domains. Generally, you can use this tool online or offline. You can use this tool interactively on the command line in batches supplied in a file (Script Mode, where scripts invoke a sequence of WLST commands without requiring your input), or embedded in Java code.

When adding Authentication Providers to a WebLogic domain, you can use WLST online to interact with an Authentication Provider and add, remove, or modify users, groups, and roles.

When you use WLST offline to create a domain template, WLST packages the Authentication Provider's data store along with the rest of the domain documents. If you create a domain from the domain template, the new domain has an exact copy of the Authentication Provider's data store from the domain template. However, you cannot use WLST offline to modify the data in an Authentication Provider's data store.

Note: You cannot use WLST offline to modify the data in an Authentication Provider's data store.

See Also:

- ["Configuring Oracle WebLogic Server for a Web Application Using ADF Security, OAM SSO, and OPSS SSO"](#)
- *Oracle Fusion Middleware Oracle WebLogic Scripting Tool*
- *Oracle Fusion Middleware WebLogic Scripting Tool Command Reference* "Infrastructure Security Commands" chapter

17.4.3.2.1 Configuring Oracle WebLogic Server for a Web Application Using ADF Security, OAM SSO, and OPSS SSO

On the Oracle WebLogic Server, you can run a Web application that uses Oracles Application Development Framework (Oracle ADF) security, integrates with Oracle Access Manager Single Sign On (SSO), and uses Oracle Platform Security Services (OPSS) SSO for user authentication. However before the Web application can be run, you must configure the domain-level `jps-config.xml` file on the application's target Oracle WebLogic Server for the Oracle Access Manager security provider.

The domain-level `jps-config.xml` file is in the following path and should not be confused with the deployed application's `jps-config.xml` file:

```
domain_home/config/fmwconfig/jps-config.xml
```

You can use an Oracle Access Manager-specific WLST script to configure the domain-level `jps-config.xml` file, either before or after the Web application is deployed. This Oracle JRF WLST script is named as follows:

Linux: `wlst.sh`

Windows: `wlst.cmd`

The Oracle JRF WLST script is available in the following path if you are running through JDev:

```
$JDEV_HOME/oracle_common/common/bin/
```

In a standalone JRF WebLogic installation, the path is:

```
$Middleware_home/oracle_common/wlst
```

Note: The Oracle JRF WLST script is required. When running WLST for Oracle Java Required Files (JRF), do not use the WLST script under `$JDEV_HOME/wlserver_10.3/common/bin`.

Command Syntax

```
addOAMSSOProvider(loginuri, logouturi, autologinuri)
```

[Table 17–11](#) defines the expected value for each argument in the `addOAMSSOProvider` command line.

Table 17–11 *addOAMSSOProvider Command-line Arguments*

Argument	Definition
<code>loginuri</code>	Specifies the URI of the login page
<code>logouturi</code>	Specifies the URI of the logout page
<code>autologinuri</code>	Specifies the URI of the autologin page

See Also:

- *Oracle Fusion Middleware Oracle WebLogic Scripting Tool*
- *Oracle Fusion Middleware WebLogic Scripting Tool Command Reference* "Infrastructure Security Commands" chapter

Prerequisites

Before starting this task, ensure that all previous tasks have been performed as described in:

- [Establishing Trust with Oracle WebLogic Server](#)
- [Configuring the Authentication Scheme for the Identity Asserter](#)

To modify domain-level `jps-config.xml` for a Fusion Web application with Oracle ADF Security enabled

1. On the computer hosting the Oracle WebLogic Server and the Web application using Oracle ADF security, locate the Oracle JRF WLST script. For example:

```
cd $ORACLE_HOME/oracle_common/common/bin
```

2. Connect to the computer hosting the Oracle WebLogic Server:

```
connect login_id password hostname:port
```

For example, the Oracle WebLogic Administration Server host could be localhost using port 7001. However, your environment might be different.

3. Enter the following command-line arguments with values for the application with ADF security enabled:

```
addOAMSSOProvider(loginuri="/${app.context}/adfAuthentication",
logouturi="/oamssso/logout.html", autologinuri="/obrar.cgi")
```

4. Stop and start the Oracle WebLogic Server.
5. Perform the following tasks as described in this chapter:
 - [Setting Up the Login Form for the Identity Asserter and OAM 10g](#)
 - [Testing Identity Assertion for SSO with OAM 10g](#)
 - [Configuring Global Logout for Oracle Access Manager 10g and 10g WebGates](#)
6. Run the application.

17.4.3.3 Setting Up Providers for Oracle Access Manager Identity Assertion

This topic describes how to configure providers in the WebLogic security domain to perform single sign-on with the Oracle Access Manager Identity Asserter. Several Authentication Provider types must be configured and ordered:

- OAM Identity Asserter: REQUIRED
- OID Authenticator: SUFFICIENT
- DefaultAuthenticator: SUFFICIENT

See Also: ["About Oracle WebLogic Server Authentication and Identity Assertion Providers"](#) on page 17-42

The following procedure uses the WebLogic Administration Console.

Note: With an Oracle Fusion Middleware application installed, you have the required provider JAR file. Skip Step 1.

To set up Providers for Oracle Access Manager single sign-on in a WebLogic domain

1. **No Oracle Fusion Middleware Application:** Obtain the Oracle Access Manager provider:

- a. Log in to Oracle Technology Network at:

http://www.oracle.com/technology/software/products/middleware/docs/111110_fmww.html

- b. Locate the oamAuthnProvider ZIP file with Access Manager WebGates (10.1.4.3.0):

```
oamAuthnProvider<version number>.zip
```

- c. Extract and copy oamAuthnProvider.jar to the following path on the computer hosting Oracle WebLogic Server:

```
BEA_HOME/wlserver_10.x/server/lib/mbeantypes/oamAuthnProvider.jar
```

2. With Oracle Fusion Middleware Application Installed:

- a.** Locate `oamauthenticationprovider.war` in the following path:

`ORACLE_INSTANCE/modules/oracle.oamprovider_11.1.1/oamauthenticationprovider.war`

- b.** Copy `oamauthenticationprovider.war` to the following location:

`BEA_HOME/wlserver_10.x/server/lib/console-ext/autodeploy/oamauthenticationprovider.war`

- 3.** Log in to the WebLogic Administration Console.

- 4.** Click **Security Realms**, *Default Realm Name*, and click **Providers**.

- 5. OAM Identity Asserter:** Perform the following steps to add this provider:

- a.** Click **Authentication**, click **New**, and then enter a name and select a type:

Name: *OAM Identity Asserter*

Type: **OAMIdentityAsserter**

OK

- b.** In the **Authentication Providers** table, click the newly added authenticator.

- c.** Click the **Common** tab, set the **Control Flag** to **REQUIRED**, and click **Save**

- 6. OID Authenticator:** Perform the following steps to add this provider.

- a.** Click **Security Realms**, *Default Realm Name*, and click **Providers**

- b.** Click **New**, enter a name, and select a type:

Name: *OID Authenticator*

Type: **OracleInternetDirectoryAuthenticator**

OK

- c.** In the **Authentication Providers** table, click the newly added authenticator.

- d.** On the **Settings** page, click the **Common** tab, set the **Control Flag** to **SUFFICIENT**, and then click **Save**.

- e.** Click the **Provider Specific** tab and specify the following required settings using values for your own environment:

Host: Your LDAP host. For example: *localhost*

Port: Your LDAP host listening port. For example: *6050*

Principal: LDAP administrative user. For example: *cn=orcladmin*

Credential: LDAP administrative user password.

User Base DN: Same searchbase as in Oracle Access Manager.

All Users Filter: For example: `(&(uid=*)(objectclass=person))`

User Name Attribute: Set as the default attribute for username in the LDAP directory. For example: *uid*

Group Base DN: The group searchbase (same as User Base DN)

Do not set the All Groups filter as the default works fine as is.

Save.

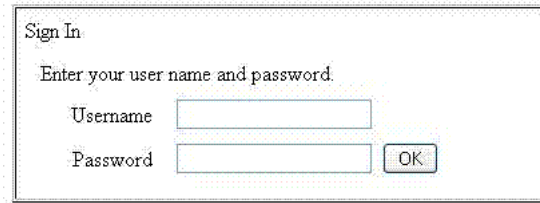
7. **Default Authenticator:** Perform the following steps to set up the Default Authenticator for use with the Identity Asserter:
 - a. Go to **Security Realms**, *Default Realm Name*, and click **Providers**.
 - b. Click Authentication, Click **DefaultAuthenticator** to see its configuration page.
 - c. Click the Common tab and set the Control Flag to **SUFFICIENT**.
 - d. Save.
8. Reorder Providers:
 - a. Click **Security Realms**, *Default Realm Name*, **Providers**.
 - b. On the Summary page where providers are listed, click the **Reorder** button
 - c. On the **Reorder Authentication Providers** page, select a provider name and use the arrows beside the list to order the providers as follows:
 - OAM Identity Asserter (REQUIRED)
 - OID Authenticator (SUFFICIENT)
 - Default Authenticator (SUFFICIENT)
 - d. Click OK to save your changes
9. **Activate** Changes: In the Change Center, click Activate Changes
10. Reboot Oracle WebLogic Server.
11. Proceed as follows:
 - Successful: Go to "[Setting Up the Login Form for the Identity Asserter and OAM 10g](#)".
 - Not Successful: Confirm that all providers have the proper specifications for your environment, are in the proper order, and that `oamAuthnProvider.jar` is in the correct location as described in "[Installing Components and Files for Authentication Providers and OAM 10g](#)" on page 17-4.

17.4.4 Setting Up the Login Form for the Identity Asserter and OAM 10g

This topic introduces the login form provided for the Oracle Access Manager Identity Asserter for single sign-on and provides a procedure that you can use to deploy the form.

The form shown in [Figure 17-9](#) is provided with the WebGate installation for Oracle HTTP Server 11g Web server. The form contains two fields (UserID and Password) and a Login button. The variables in this form are required by the Form Login authentication scheme that was generated by the OAMCfgTool and used in the policy domain protecting resources for Identity Assertion.

Figure 17–9 Default Login Form for Single Sign-On with 10g WebGates



Surrounding text describes this graphic.

Note: Do not alter any variables in this login form. Variables are required for use with Oracle Access Manager Identity Asserter.

The following information is added to the Oracle HTTP Server 11g Web server httpd.conf file during WebGate installation and configuration. It ensures that WebGate for Oracle HTTP Server 11g can find the default login form.

```
Alias /oamssso "/oam/webgate/access/oamssso"
```

Delete the following three lines if they exist:

```
<LocationMatch "/oamssso/*">
Satisfy any
</LocationMatch>
```

The following procedure guides as you set up the login form for your environment.

Note: The Login form is for only 10g WebGates with OAM 10g.

To set up the login form for Identity Assertion and OAM 10g

1. Verify that the login form is located in the following Oracle HTTP Server11g WebGate path on the computer hosting the application:
WebGate_install_dir/access/oamssso/login.html
2. From your browser, go to the following URL:
http://WebGatehost:port/oamssso/login.html
3. Confirm that the /access policy was created and enabled to protect Policy Manager resources to ensure that the login process can redirect authenticated users to the originally requested application URL.
4. Proceed to:
 - [Configuring Global Logout for Oracle Access Manager 10g and 10g WebGates](#)
 - [Testing Identity Assertion for SSO with OAM 10g](#)

17.4.5 Testing Identity Assertion for SSO with OAM 10g

The following procedure describes how to test your Oracle Access Manager Identity Assertion setup.

Alternatively, you can run Access Tester within Oracle Access Manager 10g to test your policy domain, as described in the *10g Oracle Access Manager Access Administration Guide*.

To validate Identity Assertion for SSO with OAM 10g

1. Enter the URL to access the protected resource in your environment. For example:

```
http://ohs_server:port/<protected url>
```
2. Provide appropriate credentials when the login form appears.
 - Successful: The implementation works.
 - Not Successful: See "[Troubleshooting Tips for OAM Provider Deployments](#)" on page 17-68.

17.5 Configuring the Authenticator for Oracle Access Manager 10g

To configure the Oracle Access Manager Authentication Provider as the Authenticator, you must perform the tasks in this section.

Prerequisites

Unless explicitly labeled Identity Assertion, all tasks described in "[Installing and Setting Up Authentication Providers for OAM 10g](#)" on page 17-1 must be completed:

- [Installing Components and Files for Authentication Providers and OAM 10g](#) which includes manually creating AccessGate profiles in the Access System Console and accepting defaults during Policy Manager setup

See Also:

- "[About Oracle Access Manager 10g Installation and Setup](#)" on page 17-2
- "[About OAM 10g WebGate/ AccessGate Profiles and Policy Domains](#)" on page 17-2
- [Converting Oracle Access Manager Certificates to Java Keystore Format](#), if you are using Simple or Cert transport security mode.
- [Creating Resource Types in Oracle Access Manager 10g](#)

Remaining tasks to configure the Oracle Access Manager Authenticator are described in the following task overview.

Note: You must be either a Master or Delegated Access Administrator in Oracle Access Manager to perform tasks here. There is no tool available to automate tasks outside Oracle Access Manager.

Task overview: Configuring the Oracle Access Manager Authenticator includes

1. Ensuring that all prerequisite tasks have been performed
2. [Creating an Authentication Scheme for the Authenticator](#)
3. [Configuring a Policy Domain for the Oracle Access Manager Authenticator](#)
4. [Configuring Providers for the Authenticator in a WebLogic Domain](#)
5. [Configuring the Application Authentication Method for the Authenticator](#)

6. [Mapping the Authenticated User to a Group in LDAP](#)
7. [Testing the Oracle Access Manager Authenticator Implementation](#)

17.5.1 Creating an Authentication Scheme for the Authenticator

This topic describes how to create an authentication scheme for the policy domain you will define for the Authenticator later. The Oracle Access Manager authentication scheme must be available before you create the policy domain.

With the Authenticator, the user is challenged for credentials based on the authentication method that is configured within the application web.xml. However, an Oracle Access Manager authentication scheme is required for the policy domain.

17.5.2 Configuring a Policy Domain for the Oracle Access Manager Authenticator

After creating an authentication scheme for the Authenticator, you must create a policy domain in Oracle Access Manager to use the scheme.

A policy domain in Oracle Access Manager includes several types of information. Individual tabs are provided where you can enter specific details, as shown in [Figure 17-10](#).

Figure 17-10 Create Policy Domain Page in the Oracle Access Manager Policy Manager



Surrounding text describes this screen.

For more information, see the following topics:

- [About Creating a Policy Domain](#)
- [Creating a Policy Domain and Access Policies for the Authenticator](#)

17.5.2.1 About Creating a Policy Domain

This topic describes the tabs in the Policy Manager that you use to enter details for your policy domain and access policies. While you might not use every tab in your policy domain, the following general information is provided:

- **General Tab:** Enter a short alphanumeric string to name this policy domain. You can use spaces in the Name field. A description is optional. Do not enable this policy domain until all details are saved and you are ready to use the domain.

- **Resources Tab:** Add resources to be protected by this policy domain. You use URL prefixes to define the policy domain content. A description is optional.
- **Authorization Rules Tab:** specify an authorization rule that consists of general information, Allow Access and Deny Access conditions, and actions for the rule, if any, to be used in an Authorization Expression later. You must specify an authorization scheme for every authorization rule you define.
- **Default Rules Tab:** Create default rules that apply to the resources protected by the policy domain, unless the resource is protected by a specific policy. From this tab you add the authentication rule, authorization expression, and audit rule for this policy domain.

Authentication Rule: A policy domain must have at least one authentication rule, which specifies one authentication scheme and authentication actions.

Authorization Expression: These include authorization rules and the operators used to combine them. The Authenticator function requires an Authorization rule that allows access by anyone.

Audit Rule: If there is no Master Audit Rule defined, you are instructed to contact your Access System Administrator.

- **Policies Tab:** If no rules are defined, the default rules for the policy domain remain in effect. For each policy you create, you can assign a specific authentication rule, authorization expression, and auditing rule. You can create policies with granular URL patterns. Before setting up a policy, decide the level of access control needed for the URL you to be protected.
- **Delegated Administrators Tab:** When adding URL prefixes to a policy domain, the Delegated Access Administrator must specify a server hosting the URL prefix.

See Also: ["Creating a Policy Domain and Access Policies for the Authenticator"](#) and the following topics in the *Oracle Access Manager Access Administration Guide*:

- "Creating an Authentication Rule for a Policy Domain"
- "Creating an Audit Rule for a Policy Domain"

17.5.2.2 Creating a Policy Domain and Access Policies for the Authenticator

The Authenticator implementation requires several default and some unique values in the policy domain. You must be a Master or Delegated Access Administrator in Oracle Access Manager to create, view, or modify a policy domain.

In the following procedure, you create a policy domain for the Authenticator to:

- Use the default Basic Authentication scheme (set up with Policy Manager) internally to authenticate users and to protect URL resources prefixed with `/Authen/Basic`.
- Protect resources of type `wl_authen`, which was defined earlier. See also, ["Creating Resource Types in Oracle Access Manager 10g"](#) on page 17-9
- Request user credentials using the Oracle Access Manager authentication scheme created earlier.

Note: The Authenticator requires the BASIC authentication method defined in the application `web.xml` file, which you will set up later as described in "[Configuring the Application Authentication Method for the Authenticator](#)" on page 17-58.

- Require a default authentication rule and actions, which you configure in the following procedure to return users and groups on authentication success.
- Require a default authorization rule with no actions, which you configure in the following procedure to allow access by anyone.

Note: The Authenticator does not perform authorization. However, you must create the authorization rule to allow access by anyone (but no authorization expression is required).

Examples in the following procedure are for illustration only. Be sure to enter appropriate values for your environment.

To create a policy domain for the Oracle Access Manager Authenticator

1. Go to the Policy Manager and log in. For example:

`http://Webserver:port/access/oblis`

where *Webserver* refers to computer that hosts the Policy Manager Web server; *port* refers to the HTTP port number of the Web server instance; `/access/oblis` connects to the Access System.

2. Click Policy Manager.
3. Click Create Policy Domain in the left navigation pane to display the Create Policy Domain page.
4. **General Tab:** Fill in the name and optional description that appear in pages showing lists of policy domains, and then click Save. For example:

Name: *Default OAM Authenticator*

Description: *For Username Resolution*

Note: Do not enable this policy domain until you finish all specifications.

5. **Resources Tab:** Click the Resources tab, click the Add button, select resource types, enter URL prefixes, and save as follows:

Resource Type: `wl_authen`

Host Identifier (optional): Select the Preferred HTTP host for the AccessGate.

URL prefix: `/Authen/Basic`

Description: *OAM Authenticator validates user name, password*

Click Add.

Resource Type: `wl_authen`

URL prefix: `/Authen/UsernameAssertion`

Description: *Authenticator Resource to validate user name*

Click Save.

- 6. Default Rules Tab:** From this tab you add the authentication rule, authorization expression, and audit rule for this policy domain. The policy domain's default rules apply to the resources it contains, unless the resource is protected by a specific policy.

- a. Click **Default Rules**, and then click Add to create the rule for the Basic Authentication scheme.
- b. **Authentication Rule:** A policy domain must have at least one authentication rule, which specifies one authentication scheme and authentication actions. Enter a Name, optional description, and choose an Authentication Scheme.

Click **Authentication Rule** and fill in the General tab as follows.

Name: Basic Authentication Scheme

Description: User name and password based authentication

Authentication Scheme: **Basic over LDAP**

Click Save.

Note: For the Authenticator you need only an Authentication Success Return Action in the rule for the ObMyGroups attribute. This Access Server-specific attribute returns all the groups to which the user belongs. Two other implementations require this action, as described in Step C.

- c. **Authentication Rule, Actions:** For the Authenticator (or to boot Oracle WebLogic with Administrator users who exist in Oracle Access Manager, or if you are using Oracle Web Services Manager).

Click the **Actions** tab, click **Add**.

Enter the following for Authentication Success:

Redirection URL: Leave blank

Return

Type: **WL_REALM**

Name: obmygroups

Return Attribute: obmygroups

This return attribute directs the Access Server to return all groups to which the user belongs.

Next, enter the name of the login parameter for user name to help in identifying the user uniquely in the LDAP directory server

Type: **WL_REALM**

Name: uid

Return Attribute: uid

This return attribute should be the name of the login parameter for the user name. This helps in identifying the user uniquely in the LDAP directory server used by Oracle Access Manager.

7. **Authorization Rule:** Click the Authorization Rules tab, click Add and:
 - a. Specify a rule name and, optionally, a brief description. For example:

Name: *Default rule for Authenticator.*

Description: *Default rule enables Authenticator function for anyone.*
 - b. Select Yes from the Enabled list and then click Save.
 - c. Click the rule, click the Allow Access tab, click Add, Under Role, select Anyone to allow anyone access to the protected resources.
 - d. Click Save.
8. **Policies Tab:** Click the Policies tab, click Add.

Fill in and save **General** details:

Name: *Default Username Resolution Policy*

Description: *Default Username Policy for Authenticator*

Resource Type: **wl_authen**

Resource operation(s): **LOGIN**

Resource: */Authen/UsernameAssertion*

Leave other items as they are.

Click Save.

Click the **Authentication Rule** sub tab, click Add, and fill in General details (Name, optional Description, Authentication Scheme).

Name: *Username Resolution Authentication Rule*

Authentication Scheme: **UsernameAssertion Authentication Scheme**

See "[Creating an Authentication Scheme for the Authenticator](#)".

Click Save.

Click the **Actions** sub tab and add the following details for Authentication Success:

- Return Type: *wl_REALM*
- Return Name: *uid*
- Return Attribute: *uid*

Note: Be sure to enter Return Attribute. *uid* is the name of the login attribute in the LDAP ObjectClass that helps to identity the user uniquely in the directory server used by Oracle Access Manager.

Click the **Actions** sub tab and add the following details for Authentication Success:

- Return Type: *wl_REALM*
- Return Name: *obmygroups*
- Return Attribute: *obmygroups*

Note: `obmygroups` returns all groups to which a member belongs.

9. **Delegated Access Admins:** When adding URL prefixes to a policy domain, the Delegated Access Administrator must specify a server hosting the URL prefix.

See Also: *Oracle Access Manager Access Administration Guide*, "Delegating Policy Domain Administration"

10. Proceed with "[Configuring Providers for the Authenticator in a WebLogic Domain](#)".

17.5.3 Configuring Providers for the Authenticator in a WebLogic Domain

This topic includes a procedure that you can use to add and configure the appropriate Authentication Providers in a WebLogic domain.

The Oracle Access Manager Authenticator must be configured along with the Default Authentication Provider in a WebLogic domain.

- DefaultAuthenticator: SUFFICIENT
- OAM Authenticator: OPTIONAL

The following procedure describes this task using the WebLogic Administration Console. You can also add these using the Oracle WebLogic Scripting Tool (WLST).

See Also:

- "[About Oracle WebLogic Server Authentication and Identity Assertion Providers](#)" on page 17-42
- *Oracle Fusion Middleware Oracle WebLogic Scripting Tool*
- *Oracle Fusion Middleware WebLogic Scripting Tool Command Reference*

Note: When a Oracle Fusion Middleware application is installed, you have the required files and can skip Step 1.

To configure providers for the Oracle Access Manager Authenticator in a WebLogic domain

1. **No Oracle Fusion Middleware Application:** Obtain the Oracle Access Manager provider if you have no Oracle Fusion Middleware application.

- a. Log in to Oracle Technology Network at:

http://www.oracle.com/technology/software/products/middleware/htdocs/111110_fmweb.html

- b. Locate the `oamAuthnProvider` ZIP file with Access Manager WebGates (10.1.4.3.0). For example:

`oamAuthnProvider<version>.zip`

- c. Extract and copy the `oamAuthnProvider.jar` to the following path on the computer hosting Oracle WebLogic Server:

`BEA_HOME/wlserver_10.x/server/lib/mbeans/types/oamAuthnProvider.jar`

2. Go to the Oracle WebLogic Administration Console.
3. **With Oracle Fusion Middleware Application Installed:**
 - a. Locate `oamauthenticationprovider.war` in the following path:
`ORACLE_INSTANCE/modules/oracle.oamprovider_11.1.1/oamauthenticationprovider.war`
 - b. Copy `oamauthenticationprovider.war` to the following location:
`BEA_HOME/wlserver_10.x/server/lib/console-ext/autodeploy/oamauthenticationprovider.war`
4. Go to the Oracle WebLogic Administration Console.
5. Click **Lock & Edit**, if desired.
6. **OAM Authenticator:**
 - a. Click **Security Realms** and select the realm you want to configure.
 - b. Select **Providers, Authentication**, and click **New** to display the Create a New Authentication Provider page
 - c. Enter a name and select a type:
Name `OAMAuthN`
Type: **OAMAuthenticator**
OK
 - d. Click the name of the Authentication Provider you have just created to display the Provider Configuration page.
 - e. In the Provider Configuration page, set the required values as follows:
Access Gate Name: The name of the AccessGate profile used by the provider. This must match exactly the name in the AccessGate configuration profile in the Access System Console.

Note: You might have only one AccessGate configuration profile for the Authenticator.

Access Gate Password: The same password, if any, that is as defined for the AccessGate configuration profile in the Access System Console.

Primary Access Server: The *host:port* of the primary Access Server that is associated with this AccessGate in the Access System Console.

Advanced Configuration: Following are several advanced configuration values.

Transport Security: The communication mode between Access Server and AccessGate: open, simple, or cert.

If transport security is Simple or Cert, include the following parameters and values:

Trust Store: The absolute path of JKS trust store used for SSL communication between the provider and the Oracle Access Server.

Key Store: The absolute path of JKS key store used for SSL communication between the provider and the Oracle Access Server.

Key Store Pass Phrase: The password to access the key store.

Simple mode pass phrase: The password shared by AccessGate and Access Server for simple communication modes.

Secondary Access Server: The *host:port* of the secondary Access Server that is associated with this AccessGate in the Access System Console.

Maximum Access Server Connections in Pool: The maximum number of connections that the AccessGate opens to the Access Server. The default value is 10.

Note: The Maximum Access Server Connections in Pool (or Minimum Access Server Connections in Pool) settings in the WebLogic Administration Console are different from the Maximum (or Minimum) Connections specified in profiles within the Access System Console.

Minimum Access Server Connections in Pool: The minimum number of connections that the Authentication Provider uses to send authentication requests to the Access Server. The default value is 5.

See Also: "[Oracle Access Manager Authentication Provider Parameter List](#)" on page 17-14 for descriptions and values of the common and provider-specific parameters

- f. Ensure that the parameter **Control Flag** is set to OPTIONAL initially.

Note: Do not set the parameter **Control Flag** to REQUIRED until you have verified that the Authentication Provider is operational and configured correctly.

7. In the Change Center, click **Activate Changes**.
8. **DefaultAuthenticator:** Under the Providers tab, select **DefaultAuthenticator**, which changes its control flag to SUFFICIENT.
9. **Reorder:** Under the Providers tab, reorder the providers so that DefaultAuthenticator is first (**OAMAuthenticator** follows **DefaultAuthenticator**).

Note: If the Oracle Access Manager Authenticator flag is set to REQUIRED, or if Oracle Access Manager Authenticator is the only Authentication Provider, perform the next step to ensure that the LDAP user who boots Oracle WebLogic Server is included in the administrator group that can perform this task. By default the Oracle WebLogic Server Admin Role includes the Administrators group.

10. **Oracle Access Manager Authenticator REQUIRED or the Only Authenticator:** Perform the following steps to set user rights for booting Oracle WebLogic Server.
 - a. Create an Administrators group in the directory server, if one does not already exist (or any other group for which you want boot access).

Note: To provide access to any other group, you must create that group in the directory server and add the user who boots WebLogic Server in that group.

- b. Confirm that the LDAP user who boots Oracle WebLogic Server is included in the Administrators (or other) group.
 - c. From the WebLogic Administration Console, go to **Security Realms**, *myrealm*, Roles and Policies, Global Roles.
 - d. Select **View Conditions** for the Admin Role.
 - e. Add the group and click Save.
11. Reboot the WebLogic Server.
 12. Once the server has started, reset the Authentication Provider parameter **Control Flag** to the appropriate value (REQUIRED, OPTIONAL, or SUFFICIENT).

Note: The recommended value is REQUIRED. To prevent a known issue, see "[JAAS Control Flag](#)" on page 17-74.

13. Proceed with "[Configuring the Application Authentication Method for the Authenticator](#)".

17.5.4 Configuring the Application Authentication Method for the Authenticator

This topic describes how to create the application authentication method for Oracle Access Manager Authenticator.

See Also: *Oracle Fusion Middleware Deploying Applications to Oracle WebLogic Server*

When you use the Oracle Access Manager Authenticator, all `web.xml` files in the application EAR file must specify BASIC in the element `auth-method` for the appropriate realm.

The `auth-method` can use BASIC or FORM values. While these look like similar values in Oracle Access Manager, the `auth-method` specified in `web.xml` files are used by Oracle WebLogic Server (not Oracle Access Manager).

Note: For the Oracle Access Manager Authenticator, Oracle recommends `auth-method BASIC` in `login-config` within `web.xml`.

To configure the application authentication method for the Authenticator

1. Locate the `web.xml` file in the application EAR file:

```
WEB-INF/web.xml
```

2. Locate the `auth-method` in `login-config` and enter BASIC. For example:

```
<security-constraint>
<web-resource-collection>
<web-resource-name>protected</web-resource-name>
<url-pattern>/servlet</url-pattern>
</web-resource-collection>
```

```

<auth-constraint>
<role-name>auth-users</role-name>
</auth-constraint>
</security-constraint>
<login-config>
<auth-method>BASIC</auth-method>
</login-config>
<security-role>
<description>Authenticated Users</description>
<role-name>auth-users</role-name>
</security-role>

```

3. Save the file.
4. Redeploy and restart the application.
5. Repeat for each web.xml file in the application EAR file.
6. Proceed with ["Mapping the Authenticated User to a Group in LDAP"](#).

17.5.5 Mapping the Authenticated User to a Group in LDAP

This topic describes how to map the authenticated user to a group in LDAP. To do this, you must edit the weblogic.xml file. For example, you might need to map your role-name *auth-users* to a group named *managers* in LDAP.

To map the authenticated user to a group in LDAP for the Oracle Access Manager Authenticator

1. Go to the application's weblogic.xml file.
2. Add the following information for your environment anywhere in the file:

```

<weblogic-web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.bea.com/ns/weblogic/weblogic-web-app
http://www.bea.com/ns/weblogic/weblogic-web-app/1.0/weblogic-web-app.xsd"
xmlns="http://www.bea.com/ns/weblogic/weblogic-web-app">
<security-role-assignment>
<principal-name>managers</principal-name>
<role-name>auth-users</role-name>
</security-role-assignment>
</weblogic-web-app>

```

3. Save the file.
4. Restart the WebLogic Server.
5. Proceed to:
 - [Configuring Global Logout for Oracle Access Manager 10g and 10g WebGates](#)
 - [Testing the Oracle Access Manager Authenticator Implementation](#)

17.5.6 Testing the Oracle Access Manager Authenticator Implementation

After performing all tasks to implement the Authenticator, you can test it by attempting to log in to the application using valid credentials. If the configuration is incorrect, a valid user is denied access.

The following procedure describes how to test your Authenticator setup. Alternatively, you can run Access Tester in Oracle Access Manager to test your policy domain, as described in the *Oracle Access Manager Access Administration Guide*.

To validate the Oracle Access Manager Authenticator implementation

1. Enter the URL to access the protected resource in your environment. For example:
`http://yourdomain.com:port`
2. Provide appropriate credentials when the login form appears.
 - Successful: The implementation works.
 - Not Successful: See "[Troubleshooting Tips for OAM Provider Deployments](#)" on page 17-68

17.6 Configuring Identity Assertion for Oracle Web Services Manager and OAM 10g

This section describes how to set up the Oracle Access Manager Identity Asserter to enable validation of ObSSOCookie token when you have Oracle Web Services Manager protecting Web services.

When the Oracle Access Manager Identity Asserter is configured for both header and ObSSOCookie token validation modes, preference is given to the presence of the header. If the header is not present, the Identity Asserter contacts the Access Server to validate the ObSSOCookie token.

Oracle Access Manager Identity Asserter works in two modes:

- The default mode of operation simply asserts the header that is set by WebGate at the perimeter.
- The alternate mode uses the custom AccessGate in oamAuthnProvider.jar. In this case, and with the absence of the header, the Identity Asserter contacts with the Access Server to validate the ObSSOCookie token.

Note: The AccessGate is required for Oracle Web Services Manager.

Prerequisites

- [Installing Components and Files for Authentication Providers and OAM 10g](#) which includes manually creating AccessGate profiles in the Access System Console for the custom AccessGate and accepting defaults during Policy Manager setup

See Also:

- ["About Oracle Access Manager 10g Installation and Setup"](#) on page 17-2
- ["About OAM 10g WebGate/ AccessGate Profiles and Policy Domains"](#) on page 17-2
- [Creating Resource Types in Oracle Access Manager 10g](#)

Task overview: Deploying the Identity Asserter with Oracle Web Services Manager includes

1. Ensuring that all prerequisite tasks have been performed
2. [Creating an Policy Domain for Use with Oracle Web Services Manager](#)
3. [Configuring Providers in a WebLogic Domain for Oracle Web Services Manager](#)

4. Testing the Identity Asserter with Oracle Web Services Manager

17.6.1 Creating an Policy Domain for Use with Oracle Web Services Manager

This topic describes how to set up a policy domain for use by the Oracle Access Manager Identity Asserter when you have Oracle Web Services Manager protecting Web services. You must be a Master or Delegated Access Administrator in Oracle Access Manager to create, view, or modify a policy domain.

The following unique values are required in this policy domain:

- Requires the default Basic over LDAP Authentication scheme (set up with Policy Manager) internally to authenticate users and to protect URL resources prefixed with `/Authen/SSOToken`.
- Protects resources of type `w1_authen`, which were defined in "Creating Resource Types in Oracle Access Manager 10g" on page 17-9
- Requires a default authentication rule with no actions, which you set up in the following procedure
- Requires a default authorization rule with actions, which you set up in the following procedure.

The following procedure walks you through creating a policy domain for use with Oracle Web Services Manager and the Oracle Access Manager Identity Asserter.

To create a policy domain for the Identity Asserter with Oracle Web Services Manager

1. Go to the Policy Manager and log in. For example:

```
http://Webserver:port/access/oblix
```

where *Webserver* refers to computer that hosts the Policy Manager Web server; *port* refers to the HTTP port number of the Web server instance; `/access/oblix` connects to the Access System Console.

2. Click Policy Manager.
3. Click Create Policy Domain in the left navigation pane to display the Create Policy Domain page.
4. **General Tab:** Fill in a name and optional description that appears in pages showing lists of policy domains, and then click Save. For example:

Name: *OAM IA OWSM*

Description: *Used by Identity Asserter with Oracle Web Services Manager*

Note: Do not enable this policy domain until you finish all details.

5. **Resources Tab:** Click the Resources tab, click the Add button, select resource types, enter URL prefixes, and save as follows:

Resource Type: `w1_authen`

URL prefix: `/Authen/SSOToken`

Description: *Used by IA OWS to validate SSO token*

Save.

6. **Authorization Rules Tab:** Add an authorization rule to use in an Authorization Expression later.

Click the **Authorization Rules** tab, then click the Add button

- a. **General Tab:** For Authorization Rules, enter a rule name and, optionally, a brief description.

Name: *Default_OAM_IA_OWS_AuthZ_Rule*

Description: *For use with OWS and Identity Asserter.*

Enabled: **Yes**

Allow takes precedence: **No**

Update Cache: Yes (updates all Access Server caches immediately)

- b. **Timing Conditions:** None required for this scenario.
- c. **Actions:** None required on this tab. Instead, you set these up under the Default Rules tab.
- d. **Allow Access:** Add details that define to whom the Allow Access part of the rule applies.
Role: **Any one**
- e. **Deny Access:** Not Needed for this scenario.
- f. Return to the General tab for Authorization Rules and enable the rule so that you can add it to an authorization expression later.

See Also: Chapter 6 in *Oracle Access Manager Access Administration Guide* for details about configuring authorization schemes and rules.

7. **Default Rules Tab:** From here you can add the authentication rule, authorization expression, and audit rule for this policy domain. These default rules apply to the resources it contains, unless the resource is protected by a specific policy.

Click **Default Rules**, and then click Add.

- a. **Authentication Rule:** A policy domain must have at least one authentication rule, which specifies one authentication scheme and optional authentication actions. Enter a Name, optional description, and choose an Authentication Scheme.

General tab: Fill in the as follows:

Name: *Default AuthN Rule*

Description: *Default Rule for OAM IA OSW*

Authentication Scheme: **Basic over LDAP**

Click Save.

Actions tab: No authentication actions are needed in the default rule for Oracle Web Services Manager.

Note: With Oracle Web Services Manager you need an Authorization rule.

- b. Authorization Expression:** The authorization expression in the default rules for a policy domain applies to all resources of the domain unless those resources are protected by a policy containing an expression.

Click the **Authorization Expression** tab, and then click Add.

Expression tab: Select the authorization rule you created in Step 6:

Select Authorization Rule: *Default_OAM_IA_OWS_AuthZ_Rule*

Click Add.

Click Save.

Actions tab: In Step 6 you defined to whom the Allow Access part of a rule applies. Here, you specify actions for Authorization success for both rules and expressions.

Click **Actions**, click **Add**, and then create a return action on Authorization Success with the following to specify what actions should be invoked when authorization succeeds.

Authorization Success: Applies to Allow Access conditions.

Return Type: *WL_REALM*

Return Name: *uid*

Return Attribute: *uid*

Click Save.

Note: Return Attribute *uid* should match the value of the login parameter for the user name to help identify the user uniquely in the Oracle Access Manager LDAP repository. Here, *uid* is the canonical name of the login attribute. If your LDAP directory uses a different attribute as the login attribute, the Name should still be "uid". However, the Return Attribute would be whatever your login attribute is configured as (mail, for example). Be careful to put these values under Return Attribute (not Return Value).

- 8. Policies Tab:** No policies are needed. Default Rules apply.
- 9. Delegated Access Admins:** When adding URL prefixes to a policy domain, the Delegated Access Administrator must specify a server hosting the URL prefix.

See Also: *Oracle Access Manager Access Administration Guide*, "Delegating Policy Domain Administration"

- 10. Validate Policy Domain:** Click My Policy Domains, click the new policy domain you created, then click View As a Page to see all specifications at once.
- 11. Proceed with** ["Configuring Providers in a WebLogic Domain for Oracle Web Services Manager"](#).

17.6.2 Configuring Providers in a WebLogic Domain for Oracle Web Services Manager

To use Oracle Access Manager Identity Asserter with Oracle Web Services Manager protected Web services, several Authentication Providers must be configured and ordered in a WebLogic domain:

- OAM Identity Asserter: REQUIRED
- OID Authenticator: SUFFICIENT
- DefaultAuthenticator: SUFFICIENT

This procedure is nearly identical to the one for the Oracle Access Manager Identity Asserter. The difference in this case is that Oracle Web Services Manager requires a custom AccessGate and additional provider-specific values are required:

- Primary Access Server: Specify the host and part. For example: *abcd:7777*
- Access Gate Name: The name of the AccessGate protecting the application. For example: *mmmm*
- Access Gate Password: The AccessGate password as specified in the Access System Console.

You can add these using either the Oracle WebLogic Administration Console or Oracle WebLogic Scripting Tool (WLST) command-line tool.

See Also:

- ["About Oracle WebLogic Server Authentication and Identity Assertion Providers"](#) on page 17-42
- *Oracle Fusion Middleware Oracle WebLogic Scripting Tool*
- *Oracle Fusion Middleware WebLogic Scripting Tool Command Reference*

Note: With a Oracle Fusion Middleware application installed, you have the required provider file. Skip Step 1.

To set up providers in a WebLogic domain

1. **No Oracle Fusion Middleware Application:** Obtain the Oracle Access Manager provider if you have no Oracle Fusion Middleware application.

- a. Log in to Oracle Technology Network at:

http://www.oracle.com/technology/software/products/middleware/htdocs/111110_fm.w.html

- b. Locate the oamAuthnProvider ZIP file with Access Manager WebGates (10.1.4.3.0). For example:

`oamAuthnProvider<version>.zip`

- c. Extract and copy the oamAuthnProvider.jar to the following path on the computer hosting Oracle WebLogic Server:

`BEA_HOME/wlserver_10.x/server/lib/mbeantypes/oamAuthnProvider.jar`

2. Log in to the Oracle WebLogic Administration Console.

3. **OAM Identity Asserter:** Perform the following steps to add this provider:

- a. Click **Security Realms**, *Default Realm Name*, and click **Providers**.

- b. Click **Authentication**, click **New**, and then enter a name and select a type:

Name: *OAM Identity Asserter*

Type: **OAMIdentityAsserter**

OK

- c. In the Authentication Providers table, click the newly added authenticator.
- d. On the Common tab, set the Control Flag to **REQUIRED**, and click Save.
- e. Click Platform-Specific tab and configure these parameters:

Primary Access Server: Specify the host and part. For example: *abcd:7777*

Access Gate Name: The name of the AccessGate protecting the application.
For example: *mmm*

Access Gate Password: The AccessGate password as specified in the Access System Console.

Save

4. OID Authenticator: Perform the following steps to add this provider.

- a. Click **Security Realms, Default Realm Name**, and click **Providers**
- b. Click New, enter a name, and select a type:

Name: *OID Authenticator*

Type: *OracleInternetDirectoryAuthenticator*

Click OK.

- c. In the Authentication Providers table, click the newly added authenticator.
- d. On the Settings page, click the **Common** tab, set the Control Flag to **SUFFICIENT**, and then click Save.
- e. Click the **Provider Specific** tab and specify the following required settings using values for your own environment:

Host: Your LDAP host. For example: *localhost*

Port: Your LDAP host listening port. For example: *6050*

Principal: LDAP administrative user. For example: *cn=orcladmin*

Credential: LDAP administrative user password.

User Base DN: Same searchbase as in Oracle Access Manager.

All Users Filter: For example: *(&(uid=*)(objectclass=person))*

User Name Attribute: Set as the default attribute for username in the LDAP directory. For example: *uid*

Group Base DN: The group searchbase (same as User Base DN)

Note: Do not set the All Groups filter as the default works fine as is.

Click Save.

5. Default Authenticator: Perform the following steps to set up the Default Authenticator for use with the Identity Asserter:

- a. Go to **Security Realms, Default Realm Name**, and click **Providers**.
- b. Click Authentication, Click **DefaultAuthenticator** to see its configuration page.
- c. Click the Common tab and set the Control Flag to **SUFFICIENT**.

- d. Click Save.
6. Reorder Providers:
 - a. Click **Security Realms, Default Realm Name, Providers**.
 - b. On the Summary page where providers are listed, click the **Reorder** button
 - c. On the **Reorder Authentication Providers** page, select a provider name and use the arrows beside the list to order the providers as follows:
 - OAM Identity Asserter (REQUIRED)
 - OID Authenticator (SUFFICIENT)
 - Default Authenticator (SUFFICIENT)
 - d. Click OK to save your changes
7. **Activate** Changes: In the Change Center, click Activate Changes
8. Reboot Oracle WebLogic Server.
9. Proceed as follows:
 - Successful: Go to "[Testing the Identity Asserter with Oracle Web Services Manager](#)".
 - Not Successful: Confirm the all providers have the proper specifications for your environment, are in the proper order, and that `oamAuthnProvider.jar` is in the correct location as described in "[Installing Components and Files for Authentication Providers and OAM 10g](#)" on page 17-4.

17.6.3 Testing the Identity Asserter with Oracle Web Services Manager

To validate the use of the Oracle Access Manager Identity Asserter with Oracle Web Services Manager, you can access the Web service protected by the Identity Asserter and Oracle Web Services Manager policies. If access is granted, the implementation works. If not, see "[Troubleshooting Tips for OAM Provider Deployments](#)" on page 17-68.

17.7 Synchronizing the User and SSO Sessions: SSO Synchronization Filter

In Fusion Middleware 11g, a new component that synchronizes the container user session and SSO session has been introduced. SSO Sync Filter is an Oracle WebLogic system filter implementation that intercepts all requests to the container, acts on protected resource requests, and attempts to synchronize the container's user session with the user identifying header in OSSO (Proxy-Remote-User) or the user data in the Oracle Access Manager SSO session cookie (ObSSOCookie).

SSO Synchronization Filter (SSO Sync Filter) is an implementation of the Servlet Filter based on Java Servlet Specification version 2.3. SSO sync filter relieves applications from tracking the SSO user session and synchronizing it with their respective sessions. Instead, applications would only need to synchronize with container's user session.

SSO Sync Filter intercepts each request to the container and determines whether to act on it based on certain HTTP headers that are attached to the request. Filter expects SSO agent to have set those headers in the Web Tier. When access is made to unprotected areas of the application, the filter acts as a pass through. Once a protected resource is accessed, SSO agents in the Web Tier, direct user to perform authentication with SSO

system such as Oracle Access Manager. After the authentication, Oracle Access Manager Identity Asserter helps establish a user identity in form of JAAS Subject to the container and a user session is created. WebLogic maintains the user session data as part of HTTP Session Cookie (JSESSIONID).

Subsequent access to the application resources provides two pieces of information to the SSO Sync Filter:

- User identifying header in OSSO (Proxy-Remote-User)
- User data in the Oracle Access Manager SSO session cookie (ObSSOCookie)

The job of SSO Sync Filter is to make sure that the user identity in the container matches with that of the SSO session. If there is a mismatch, filter invalidates the container's user session. As a result, the downstream application would only have to track container user session and react in a consistent fashion regardless of SSO environment in use.

Notes:

- **Enabled and Active by Default:** SSO Sync Filter fetches the user information from the configured tokens, gets the user from existing session (if any), invalidates the session and redirects to the requested URL in case the `CurrentSessionUser` does not match the incoming SSO User. Otherwise, the request is simply passed through.

If you have not configured the OSSO or Oracle Access Manager Assertion Providers in your domain, the filter disables automatically during WebLogic Server start-up.

- **Active for All URI's by Default (/*):** No changes are required in the application code.
- **Configured for the OSSO Tokens/Header:** Proxy-Remote-User, and performs a case insensitive match.
- **Configured for the Oracle Access Manager SSO Tokens/Header:** OAM_REMOTE_USER and REMOTE_USER, and does a case insensitive match.
- **Global Logout:** SSO Sync Filter is intended to provide the Single Logout Experience to the Oracle Fusion Middleware applications that use the OSSO or Oracle Access Manager Solutions. Is handled similarly to single sign-on. After global logout is performed, SSO filter reconciles the session when subsequent access to an application that has not cleaned up its session is made.

Any application that use the OSSO or Oracle Access Manager Solutions is expected to invalidate its session before making a call to OSSO logout or Oracle Access Manager logout. For more information on OSSO logout, see [Example 18-2, "SSO Logout with Dynamic Directives"](#) on page 18-11. For details about Oracle Access Manager logout, see ["Configuring Global Logout for Oracle Access Manager 10g and 10g WebGates"](#) on page 17-10.

- **Application Session Time Out:** SSO cookies typically track user inactivity/idle times and force users to login when a time out occurs. OSSO and Oracle Access Manager are no exception. Oracle Access Manager takes a sophisticated approach at this and specifically tracks Maximum Idle Session Time and Longest Idle Session Time along with SSO session creation time and time when it was last refreshed.

The general recommendation for applications that are maintaining their own sessions when integrating with SSO systems is to configure their session time outs

close to that of SSO session time outs so as to make user experience remains consistent across SSO and application session time outs.

You can alter the behavior of the SSO Sync Filter for application requirements by passing various over-riding system properties to WebLogic. To do this, you change the Oracle WebLogic startup script and check for EXTRA_JAVA_PROPERTIES in setDomainEnv.sh. The properties and Sync behavior is shown in [Table 17-12](#).

Table 17-12 SSO Sync Filter Properties and Sync Behavior

Area	Overriding System Property	Default value of System property	Default Behavior of the Sync Filter
Status (Active or Inactive)	sso.filter.enable	Not configured	Enabled
Case sensitive matches	sso.filter.name.exact.match	Not configured	Case Ignore Match
Configured Tokens	sso.filter.ssotoken	Not configured	<ul style="list-style-type: none"> ■ OSSO: Look for Proxy-Remote-User ■ Oracle Access Manager: Look for OAM_REMOTE_USER and REMOTE_USER. OAM_REMOTE_USER takes precedence.
URI Mappings	Not Applicable	Not Applicable	/*

You cannot enable the filter for selected applications. The SSO Sync Filter is a system filter. As such, it is activated for all deployed applications (the URI mapping is /*).

Note: You cannot enable the filter for selected applications.

The following procedure gives some tips about modifying the SSO Sync filter properties and behavior.

To modify the SSO Sync Filter properties and behavior

1. **Disable the Filter:** Change the system property "sso.filter.enable" to "false" (pass as -D to the jvm) and restart the Oracle WebLogic Server. This toggles the filter status.
2. **User-Identifying Header Differs from Pre-Configured Sync Filter Tokens:** Over-ride the SSO token that the Sync Filter looks for using the system property "sso.filter.ssotoken".

For example, pass to the WebLogic Server jvm in the WebLogic Server startup script -Dso.filter.ssotoken=HEADERNAME, and restart the server.

When you contact Oracle Support you might be requested to set up debugging, as described in "[Setting Up Debugging in the WebLogic Administration Console](#)" on page 15-14.

17.8 Troubleshooting Tips for OAM Provider Deployments

This section contains the following topics:

- [About Using IPv6](#)

- [Apache Bridge Failure: Timed Out](#)
- [Authenticated User with Access Denied](#)
- [Browser Back Button Results in Error](#)
- [Cannot Reboot After Adding OAM and OID Authenticators](#)
- [Client in Cluster with Load-Balanced WebGates](#)
- [Error 401: Unable to Access the Application](#)
- [Error 403: Unable to Access the Application](#)
- [Error 404: Not Found ... Anything Matching the Request URI](#)
- [Error Issued with the Action URL in Form Login Page](#)
- [Error or Failure on Oracle WebLogic Server Startup](#)
- [JAAS Control Flag](#)
- [Login Form is Shown Repeatedly Upon Credential Submission: No Error](#)
- [Logout and Session Time Out Issues](#)
- [Not Found: The requested URL or Resource Was Not Found](#)
- [Oracle WebLogic Server Fails to Start](#)
- [Oracle ADF Integration and Cert Mode](#)

See Also: ["Setting Up Debugging in the WebLogic Administration Console"](#) on page 15-14

17.8.1 About Using IPv6

Oracle Fusion Middleware and Oracle Access Manager support Internet Protocol Version 4 (IPv4) and Internet Protocol Version 6 (IPv6.) Among other features, IPv6 supports a larger address space (128 bits) than IPv4 (32 bits), providing an exponential increase in the number of computers that can be addressable on the Web.

See Also: *Oracle Fusion Middleware Administrator's Guide* for details about using IPv6.

17.8.2 Apache Bridge Failure: Timed Out

If you experience a failure of the Apache bridge, you might see a message stating that there is no back-end server available for connection. In this case, the connection times out.

The Oracle WebLogic Server might be down or there might be incorrect values set in `mod_weblogic`.

To recover from an Apache Bridge Failure

1. Check the Oracle WebLogic Server to ensure that it is available.
2. Confirm that host and port information is specified correctly in the WebGate's Web server `httpd.conf`. For example:

```
ORACLE_INSTANCE/config/OHS/<ohs_name>/httpd.conf
```

```
<IfModule mod_weblogic.c>
    WebLogicHost yourHost.yourDomain.com
    WebLogicPort yourWlsPortNumber
```

</IfModule>

17.8.3 Authenticated User with Access Denied

It is possible that an authenticated user does not have access rights to the requested resource.

If a user login is inconclusive or invalid, the user can be authenticated but not recognized as authorized for the requested resource. In this case, no explicit error message states the issue. Instead, the user is prompted to log in again.

17.8.4 Browser Back Button Results in Error

After successful authentication, if you click the Back button in the browser window, you might get an error for access/oblix/apps/webgate/bin/webgate.so.

When form-based authentication is used, Oracle Access Manager creates a form login cookie that holds information about the requested resource. On successful authentication, the state of the cookie changes. When the user clicks the Back button, the login form appears. When re-posted, the form login cookie no longer holds redirection details.

The ObSSOCookie is also sent with the form login cookie. The ObSSOCookie is correctly checked. As the form login cookie state changes, the form-based authentication does not occur and the form action is considered as a request for the resource.

Solution

Retry the request using the original URL.

17.8.5 Cannot Reboot After Adding OAM and OID Authenticators

If the Oracle Access Manager Authenticator flag is set to REQUIRED, or if Oracle Access Manager Authenticator is the only Authentication Provider, perform the next step to ensure that the LDAP user who boots Oracle WebLogic Server is included in the administrator group that can perform this task. By default the Oracle WebLogic Server Admin Role includes the Administrators group.

To provide access to any other group, you must create that group in the directory server and add the user who boots WebLogic Server in that group.

To ensure you can restart the WebLogic Server

1. Create an Administrators group in the directory server, if one does not already exist (or any other group for which you want boot access).
2. Confirm that the LDAP user who boots Oracle WebLogic Server is included in the Administrators (or other) group.
3. From the WebLogic Administration Console, go to **Security Realms, myrealm, Roles and Policies, Global Roles**.
4. Select View Conditions for the Admins Role.
5. Add the group and click Save.

17.8.6 Client in Cluster with Load-Balanced WebGates

Out of the box, Oracle Access Manager does not support load balanced AccessGates; you must use a third-party load balancer.

Suppose you have two WebGates: WebGateA and WebGateB. You can use the OAMCfgTool to create the profile to be shared by the two WebGates.

See Also: ["Introduction to OAMCfgTool"](#) on page 17-15

If you have an Oracle Fusion Middleware Application installed you already have the OAMCfgTool. In this case, skip Step 1.

Solution:

1. **No Oracle Fusion Middleware Application:** Obtain the OAMCfgTool if you have no Oracle Fusion Middleware application installed.

- a. Log in to Oracle Technology Network at:

http://www.oracle.com/technology/software/products/middleware/html/docs/111110_fm.html

- b. Locate the OAMCfgTool ZIP file with Access Manager Core Components (10.1.4.3.0):

```
oamcfgtool<version>.zip
```

- c. Extract and copy oamcfgtool.jar to the computer hosting WebGate:

2. Log in to the computer for *WebGateA* (even if WebGate is not yet installed).

3. Change to the file system directory containing OAMCfgTool and run a command like the following one to create one AccessGate Profile to be shared by the two WebGates. For example:

```
java -jar oamcfgtool.jar mode=CREATE app_domain=SharedA_B
app_agent_password=<WebGate_password>
cookie_domain=<preferred_http_cookie_domain>
ldap_host=wxyz
ldap_port=6633
ldap_userdn=orcladmin
ldap_userpassword=<ldap_userpassword>
oam_aaa_host=abcd
oam_aaa_port=7789
oam_aaa_mode=cert
log_file=OAMCfg_date.log
log_level=INFO
output_ldif_file=<LDIF_filename>
```

4. Review the information provided by the tool. For example, the parameters and values in Step 3 would provide the following information:

```
Processed input parameters
Initialized Global Configuration
Successfully completed the Create operation.
Operation Summary:
  Policy Domain : SharedA_B
  Host Identifier: SharedA_B_WD
  Access Gate ID : SharedA_B_AG
```

Note:

- Perform Step 5 if you have WebGate installed.
 - Perform Step 6 if WebGate is not yet installed.
-
-

5. **Output LDIF Created:** Import the LDIF to write information to the directory server. Otherwise, skip this step.
6. **WebGates Not Installed:** Install *WebGateA* and *WebGateB* and specify the same values as you did when creating the profile (plus additional values to properly finish the installation).
7. **Installed WebGates:** Using output from the OAMCfgTool Create command, run the Oracle Access Manager `configureWebGate` tool to set up the WebGate. For example:
 - a. Go to:


```
WebGate_install_dir\access\oblix\tools\configureWebGate
```

 where *WebGate_install_dir* is the directory where WebGate is installed.
 - b. Run the following command to configure the WebGate using values specified with OAMCfgTool and other values needed to finish the installation. For example:


```
configureWebGate -i WebGate_install_dir -t WebGate SharedA_B_AG
-P WebGate_password
-m <open|simple|cert>
-h Access_Server_Host_Name
-p Access_Server_Port
-a Access_Server_ID
-r Access_Server_Pass_Phrase (must be the same as the WebGate_password)
-Z Access_Server_Retry count
```

See Also: "Configuring AccessGates and WebGates" in the *Oracle Access Manager Access Administration Guide*
 - c. Repeat these steps to configure *WebGateB*.
8. **Confirm Profile in the Access System Console:** Perform the following steps to view or modify the WebGate profile.
 - a. Log in to the Access System Console as a Master or Delegated Access Administrator. For example:


```
http://hostname:port/access/oblix
```

hostname refers to computer that hosts the Web server; *port* refers to the HTTP port number of the Web server instance; `/access/oblix` connects to the Access System Console.
 - b. Click **Access System Configuration**, and then click **AccessGate Configuration**.
 - c. Click the All button to find all profiles (or select the search attribute and condition from the lists) and then click Go.
 - d. Click a WebGate's name to view its details.
 - e. Click Cancel to dismiss the page without changes, or click Modify to change values as described in the *Oracle Access Manager Access Administration Guide*.
9. In the load balancer host identifiers, add host name variations for both WebGates: *WebGateA* and *WebGateB*.

17.8.7 Error 401: Unable to Access the Application

An error message like the following:

401 Authorization Required

This typically means that the Oracle Access Manager Authentication Provider is incorrectly configured. For a listing of correct configurations, see "[Oracle Access Manager Authentication Provider Parameter List](#)" on page 17-14.

17.8.8 Error 403: Unable to Access the Application

An error message like the following:

403 Forbidden

This typically means that the post-authenticate actions are incorrectly configured in the policy domain.

Under the policy domain's authentication success actions, ensure that you have set `obmygroups` and `uid` in the Return Attribute field (not in the Return Value field).

For more information, see "[Configuring a Policy Domain for the Oracle Access Manager Authenticator](#)" on page 17-50.

17.8.9 Error 404: Not Found ... Anything Matching the Request URI

Generally, this error indicates that the server has not found anything matching the Request-URI. This message informs that the Oracle WebLogic Server is not able to find a resource.

There is no indication of whether the condition is temporary or permanent:

- If the server cannot make temporary or permanent information available to the client, the status code 403 (Forbidden) can be used.
- If, through some internally configurable mechanism, the server could state that an old resource is permanently unavailable and has no forwarding address, the 410 (Gone) status code should be used.

To recover from Error 404

Confirm that the resource is deployed on the Oracle WebLogic Server. For example, if the pattern is `/private1/Hello`, confirm that `Hello` is accessible on the server with `private1` as the root.

17.8.10 Error Issued with the Action URL in Form Login Page

This issue occurs if Form Authentication scheme is not properly configured in Oracle Access Manager. However, this cannot occur if you use the `OAMCfgTool` to set up a policy domain. For example:

Symptoms include:

- The user name and password fields in the login form must match the details in the Form authentication scheme
- The `credential_mapping` filter must be specified correctly in the Form authentication scheme
- The login form action URL must be protected with a policy
- The login form action URL must match the Action value specified in the authentication scheme's challenge parameter

17.8.11 Error or Failure on Oracle WebLogic Server Startup

If the WebLogic Server user is not part of the administrator's group in Oracle Access Manager, Oracle WebLogic Server restart and Authentication Provider initialization can fail. In this case, one of the following messages might appear in the AdminServer.log in \$DOMAIN_HOME/servers/AdminServer/logs/AdminServer.log:

```
)<Failed ---- FatalError:InvalidSchemeMapping
...
Authentication Failed.
...
Login failed.
...
```

Solution

1. Confirm that the implementation is using the Oracle-provided default login form.
2. Create a group named "Administrators" in the Oracle Access Manager Identity System, and include the Oracle WebLogic Server user.

See Also: *Oracle Access Manager Identity and Common Administration Guide*

3. Login to Oracle WebLogic Server using the credentials of the user in the Administrators group defined within the Oracle Access Manager Identity System.
4. Restart the Oracle WebLogic Server.

17.8.12 JAAS Control Flag

If this flag is set to REQUIRED and any other parameter is set to an incorrect value, the server does not start.

To prevent this issue, ensure that the Oracle Access Manager Authentication Provider is properly configured while this parameter value is set to OPTIONAL. Only after you have validated proper behavior in this way, should you reset the control flag to REQUIRED.

For more information, see ["Configuring Providers for the Authenticator in a WebLogic Domain"](#) on page 17-55.

17.8.13 Login Form is Shown Repeatedly Upon Credential Submission: No Error

This issue typically points to an incorrect user name or password. No error is shown.

Ensure that you are supplying the correct user name and password. The user login name must be the value of the attribute that is configured in the Form Login authentication scheme. For example, Challenge Parameter creds: userid.

17.8.14 Logout and Session Time Out Issues

When a user logs out, or a user session times out, the user should be challenged for reauthentication. However, the following might occur instead:

- Logout: After logging out, if the user attempts to access the application in the same browser window the application is still accessible without reauthenticating.

- **Session Time Out:** After a user session time out, the user is challenged to reauthenticate. However, if the user gives a different user ID he is granted the same privileges as the previous user.

The ObsSOCookie is still present. Some configuration must be done at the application level to kill the ObsSOCookie. For proper behavior, WebLogic application session time out values should be the same as WebGate session time out values.

If setting up an Identity Asserter in the WebLogic Application Console, the Web application using the Identity Asserter must have its `auth-method` set to `CLIENT-CERT`. For more information, see "[Configuring OAM Identity Assertion for SSO with Oracle Access Manager 10g](#)" on page 17-35.

17.8.15 Not Found: The requested URL or Resource Was Not Found

If you receive a message stating that the requested URL or resource was not found on this server, the reverse proxy Web server might not be forwarding requests to the Oracle WebLogic Server.

To ensure that the reverse proxy is forwarding requests to Oracle WebLogic Server

1. Locate the `httpd.conf` file on the reverse proxy WebGate Web server. For example:

```
ORACLE_INSTANCE/config/OHS/<ohs_name>/httpd.conf
```

2. Confirm the correct settings to forward requests to the correct host and port of the Oracle WebLogic Server:

```
#httpd.conf
<IfModule mod_weblogic.c>
    WebLogicHost <host>
    WebLogicPort yourWlsPortNumber
</IfModule>

<Location /request-uri-pattern>
    SetHandler weblogic-handler
</Location>
```

17.8.16 Oracle WebLogic Server Fails to Start

If the Oracle WebLogic Server fails to start, you can take the following actions.

1. Determine whether the Oracle Access Manager Authentication Provider is the only provider configured in the Oracle WebLogic Server realm. If it is, continue with Step 2.
2. Confirm whether the Oracle Access Manager Authentication Provider is configured correctly and make any changes needed.
3. Determine whether the Oracle Access Manager Authentication Provider control flag is set to `REQUIRED`. In this case, perform the following steps:
 - a. Create an Administrators group in the directory server, if one does not already exist (or any other group for which you want boot access).

Note: To provide access to any other group, you must create that group in the directory server and add the user who boots WebLogic Server in that group.

- b. Confirm that the LDAP user who boots Oracle WebLogic Server is included in the Administrators (or other) group.
- c. From the WebLogic Administration Console, go to Security Realms, *Your Realm*, Roles and Policies, Global Roles.
- d. Select View Conditions for the Administrators (or other) role.
- e. Add the group and click Save.

17.8.17 Oracle ADF Integration and Cert Mode

Problem

WebGate configuration of cache directives might not be compatible with certain browser versions (specifically Internet Explorer v7) when accessing certain URLs that allow you to download Microsoft Office documents (.xls, .doc, and so on).

For example, suppose that you have an Excel workbook deployed along with an Oracle ADF application in an Oracle Access Manager Cert-based environment.

If the ADFDi component is trying to access two URLs, and trying the second URL first, a failure occurs regardless of the ADFDi client side code. It is not able to handle the redirect from Oracle Access Manager WebGate to the SSL enabled endpoint and fails with the following stack trace:

```
WebException: The request was aborted: Could not create SSL/TLS secure channel
```

If you attempt to access the workbook, and the following message appears:

```
Microsoft Office Excel cannot access the file
```

The cause could be any of the following:

- The file name or path does not exist.
- The file is being used by another program.
- The workbook you are trying to save has the same name as a currently open workbook.

However, if the message appears when the URL to workbook is explicitly pasted to Internet Explorer v7 address bar it might be due to WebGate default Cache Directives.

WebGates have default Cache Directives (Pragma=no-cache and CacheControl=no-cache) that might cause a problem with Internet Explorer v7 when a URL to an .xls workbook is directly pasted into the browser's address bar.

Solution

If the message appears when the URL to workbook is explicitly pasted to Internet Explorer v7 address bar, Oracle recommends removing the cache directives from respective WebGate configuration pages in the Access System Console.

To remove cache directives from respective WebGate configurations

1. From the Access System Console, click the Access System Configuration tab.
2. Click AccessGate Configuration, click Go on the search page, and then click the link to the desired AccessGate configuration page.
3. On the Details for AccessGate page, click Modify.

4. On the Modify AccessGate page, locate Web Server Client label and clear the following fields:
 - CachePragmaHeader
 - CacheControlHeader
5. Click Save.

17.8.18 About Protected_JSessionId_Policy

OAM Policies are evaluated based on the URIs passed to it. With earlier releases, there was no policy for protecting `*;jsessionid*`. When an application resource URL was accessed and the JSESSIONID cookie was not found, WebLogic Server wrote the URL by including the JSESSIONID as part of the URL. If the URL in question was protected, Oracle Access Manager and OSSO Web agents could have issues matching the re-written URL.

In this release, a new policy is available that uses a pattern `"*;jessionid=*"` for all URIs under the context-root. Therefore, any URI under the context-root, with `"jsessionid=string"` appended to it, is considered protected.

The `/context-root` itself must be listed as a resource. The URL pattern is `*;jsessionid=*`. The Default authentication rule is a protected authenticating scheme. The Default authorization expression is also used. When ordering policies, this policy must be first.

Suppose you have one protected resource named `/test/protectedUri` and a public resource named `/test`. When you create a public policy with the pattern `*;jessionid;*` and apply this policy to both the above resources the public policy should have precedence over the public resource.

- When `/test;jessionid=blah` is requested, OAM first checks for a default rule for `"/test;jessionid=blah"`. Without such a rule, OAM then checks for a rule for `"/"`. Without this rule, the URL, `"/test;jessionid=blah"` is considered to be unprotected.
- When `"/test/protectedUri;jessionid=blah"` is requested, OAM checks for a default rule to protect this. Without such a rule, OAM then checks for a rule for `"/test"`. With `"/test"` in the Resources list, OAM further determines which policy to apply. In this case, the `jessionid` policy is applied and the request deemed to be protected.

Configuring Single Sign-On using OracleAS SSO 10g

The chapter describes how to implement SSO using OracleAS SSO (OSSO) 10g. It includes the following major sections:

- [Deploying the OracleAS 10g Single Sign-On \(OSSO\) Solution](#)
- [Synchronizing the User and SSO Sessions: SSO Synchronization Filter](#)
- [Troubleshooting for an OSSO Identity Asserter Deployment](#)

18.1 Deploying the OracleAS 10g Single Sign-On (OSSO) Solution

The OracleAS Single Sign-On solution provides single sign-on access to Web Applications. Oracle Internet Directory is the LDAP-based repository.

This solution is intended for applications that have been deployed on Oracle WebLogic Server but do not yet have single sign-on implemented. Requirements and steps to configure the OSSO solution are explained in "[New Users of the OSSO Identity Asserter](#)" on page 18-4.

Note: Oracle recommends using Oracle Access Manager 11g, as described in "[Introduction to Oracle Access Manager 11g SSO](#)" on page 16-1.

Applications that are already using the OracleAS Single Sign-On solution with the JPS login module and dynamically re-directing requests to OSSO are unaffected by the new OSSO solution. In this case, there is no need to configure the new OSSO Authentication Provider described in this section.

This section is divided as follows:

- [Using the OSSO Identity Asserter](#)
- [New Users of the OSSO Identity Asserter](#)
- [Troubleshooting for an OSSO Identity Asserter Deployment](#)

18.1.1 Using the OSSO Identity Asserter

This section describes the expected behavior when you implement the OracleAS Single Sign-On Identity Asserter. This section is divided as follows:

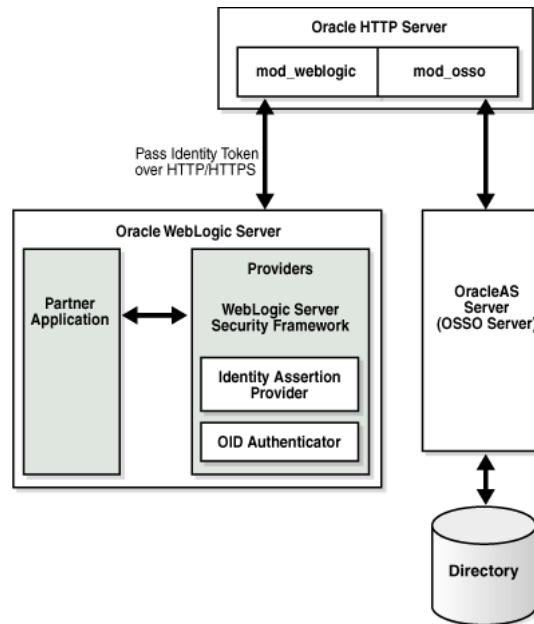
- [Oracle WebLogic Security Framework](#)

- [OSSO Identity Asserter Processing](#)
- [Consumption of Headers with OSSO Identity Asserter](#)

18.1.1.1 Oracle WebLogic Security Framework

Figure 18–1 illustrates the location of components in the Oracle WebLogic Security Framework, including the OSSO Identity Asserter. Additional details follow.

Figure 18–1 Location of OSSO Components in the Oracle WebLogic Security Framework



The following text describes this figure.

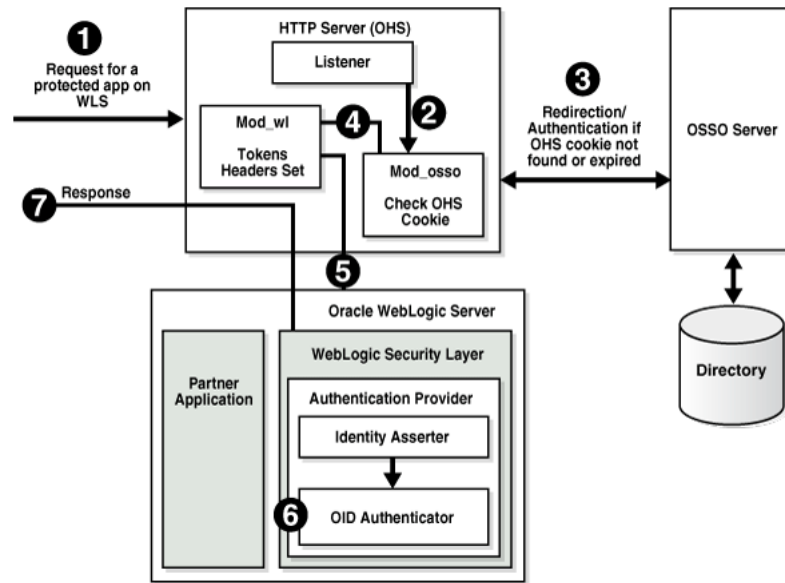
At the top of the figure, Oracle HTTP Server is installed. This installation includes mod_weblogic and mod_osso, which are required to pass the identity token to the Providers and Oracle WebLogic Server. The Oracle WebLogic Server includes the partner application and the Identity Asserter (also known as the Identity Assertion Provider). The 10g OracleAS Single Sign-On server (OSSO Server), on the right side of the figure, communicates directly with the directory server and Oracle HTTP Server.

Note: For simplicity in text, this chapter uses the generic name of the WebLogic Server plug-in for Apache: mod_weblogic. For Oracle HTTP Server, the name of this plug-in differs from release 10g to 11g:

- Oracle HTTP Server 10g: mod_wl (actual binary name is mod_wl_20.so)
 - Oracle HTTP Server 11g: mod_wl_ohs (actual binary name is mod_wl_ohs.so)
-

18.1.1.2 OSSO Identity Asserter Processing

Figure 18–2 illustrates the processing that occurs when you have OSSO implemented with the Identity Asserter. Additional details follow the figure.

Figure 18–2 OSSO Identity Asserter Processing

This diagram is described in following text.

The first time a request for a protected resource arrives at the mid-tier Web server, the request is redirected to the 10g OracleAS Single Sign-On server, which requires user credentials. For a certificate-based authentication, no login page is displayed. After the user has been successfully authenticated, all further requests from that user require only that the user identity be asserted by the OSSO Identity Asserter before the population of a JAAS Subject takes place. The Subject is consumed by the downstream applications.

For example, suppose you have an application residing on an Oracle WebLogic Server that is front-ended with the Oracle HTTP Server. The application is protected using resource mappings in the mod_osso configuration. This case is described in the following process overview.

Process overview: OSSO Identity Asserter

1. The user requests a protected application.
2. The Oracle HTTP Server intercepts the request and processes it using mod_osso to check for an existing, valid Oracle HTTP Server cookie.
3. If there is no valid Oracle HTTP Server cookie, mod_osso redirects to the OracleAS SSO Server, which contacts the directory during authentication.
4. After successful authentication mod_osso decrypts the encrypted user identity populated by the OSSO server and sets the headers with user attributes.
5. mod_weblogic completes further processing and redirects the request to the Oracle WebLogic Server.
6. The WebLogic security layer invokes providers depending on their settings and the order specified. For example: the security layer invokes the:
 - Identity Asserter, which makes the identity assertion based on retrieved tokens

- Oracle Internet Directory Authenticator (OID Authenticator), which populates the Subject with necessary Principals

See Also: ["Consumption of Headers with OSSO Identity Asserter"](#)

7. A response is sent to the user through the Oracle HTTP Server, and access to the application is granted.

18.1.1.3 Consumption of Headers with OSSO Identity Asserter

This topic describes the headers sent by Oracle HTTP Server and the tokens set in the header and the headers consumed by the OSSO Identity Asserter. If the application needs to use the JAAS subject, configure OSSO Identity Asserter.

[Table 18–1](#) provides the list of headers set by Oracle HTTP Server (mod_osso and mod_weblogic). An application whose logic consumes the JAAS subject for identifying user information, should be configured to use the OSSO Identity Asserter, which uses the OracleAS SSO token type set in bold in the table (**Proxy-Remote-User**). The OSSO Identity Asserter looks for the **Proxy-Remote-User** header and asserts the user's identity. The follow up OID Authenticator populates the JAAS subject.

Table 18–1 Headers Sent by Oracle HTTP Server

Attribute	Sample Value	Description
Cookie	OHS-Stads42.us.oracle.com:7777=.....	Cookies
Osso-User-Guid	4F4E3D2BF4BFE250E040548CE9816D7E	GUID of the authenticated user
Osso-User-Dn	cn=orcladmin,cn=users,dc=us,dc=oracle,dc=com	DN of the authenticated user
Osso-Subscriber	DEFAULT COMPANY	Subscriber name
Osso-Subscriber-Dn	dc=us,dc=oracle,dc=com	Base DN of the subscriber
Osso-Subscriber-Guid	4F4E3D2BF410E250E040548CE9816D7E	GUID of the subscriber
Proxy-Remote-User	ORCLADMIN	The authenticated user
Proxy-Auth-Type	Basic SSO	Authentication type

Applications that do not require the JAAS subject for identifying user information, can read the headers directly using the request.getHeader() API. Such applications are free to read any header they need. Headers with user info are Osso-User-Dn, Osso-User-Guid, and Proxy-Remote-User.

18.1.2 New Users of the OSSO Identity Asserter

The new OracleAS Single Sign-On solution includes the OSSO Identity Asserter, one of the two new Authentication Providers for the Oracle WebLogic Server.

To have your application use the OSSO solution, you need the components described in the following task.

Note: If you already have components installed and set up, you do not need more. You can skip any steps that do not apply to your deployment.

Task overview: Deploying and configuring the OSSO Identity Asserter

1. Install the following components:

- a. OracleAS Single Sign-On Server 10g (10g OSSO server)

See Also: *Oracle Application Server Installation Guide* on Oracle Technology Network at:

<http://www.oracle.com/technology/documentation/oim1014.html>

- b. An Oracle Internet Directory repository configured to be used by the 10g OSSO server. Ensure that the directory server is tuned for your deployment.

See Also: The following manuals for Release 11g (11.1.1.1.0)

- *Oracle Fusion Middleware Installation Guide for Oracle Identity Management*
- *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*

- c. One of the following Web servers (based on Apache 2):

- Oracle HTTP Server 11g as a front end to the Oracle WebLogic Server. This installation includes mod_osso and mod_weblogic.
- OHS 10g, available in the companion CD release Oracle HTTP Server 10.1.3. This includes mod_osso. However, mod_weblogic must be added.

See Also: The following manuals for Release 11g (11.1.1.1.0)

- *Oracle Fusion Middleware Installation Guide for Web Tier*
- *Oracle Fusion Middleware Administrator's Guide for Oracle HTTP Server*

- d. Oracle WebLogic Server 10.3.1+

See Also: *Oracle Fusion Middleware Getting Started With Installation for Oracle WebLogic Server*

- e. An Oracle Fusion Middleware product such as Oracle Identity Management, Oracle SOA Suite, or Oracle WebCenter is required; it includes the provider required for OSSO by Oracle WebLogic Server in the following path:

ORACLE_INSTANCE/modules/oracle.ossoiap_11.1.1/ossoiap.jar

See Also:

- *Oracle Fusion Middleware Installation Guide for Oracle Identity Management*
- *Oracle Fusion Middleware Installation Guide for Oracle SOA Suite*
- *Oracle Fusion Middleware Installation Guide for Oracle WebCenter*

2. Configure mod_weblogic so that it forwards requests to Oracle WebLogic Server, as explained in section "Configuring mod_weblogic" on page 18-6.
3. Register the module mod_osso with the 10g SSO Server as a partner application, as described in "Registering Oracle HTTP Server mod_osso with OSSO Server 10.1.4" on page 18-7.

4. Configure `mod_osso`, as described in ["Configuring mod_osso to Protect Web Resources"](#) on page 18-8.
5. Add the OSSO Identity Asserter to the appropriate domain, as explained in section ["Adding Providers to a WebLogic Domain for OSSO"](#) on page 18-12.
6. Configure a connection filter, as explained in section ["Establishing Trust Between Oracle WebLogic Server and Other Entities"](#) on page 18-14.
7. Configure the use of the solution by the application, as explained in section ["Configuring the Application for the OSSO Identity Asserter"](#) on page 18-15.
8. Identify and resolve issues with your OSSO Identity Asserter implementation, see ["Troubleshooting for an OSSO Identity Asserter Deployment"](#) on page 18-18.

18.1.2.1 Configuring mod_weblogic

You can either edit the Oracle HTTP Server `httpd.conf` file directly or add `mod_weblogic` configuration in a separate file and include that file in `httpd.conf`.

The following procedure includes steps for two different Web server releases. Perform steps as needed for your deployment:

- OHS 11g ships with `mod_wl_ohs.so`. In this case, skip Step 1.
- OHS 10g does not ship with `mod_weblogic (mod_wl.so)`. If Oracle HTTP Server 10g is installed, start with Step 1 to copy `mod_wl_20.so` before configuration.

Note: For Oracle HTTP Server, the name of this plug-in differs from release 10g to 11g:

- Oracle HTTP Server 10g: `mod_wl` (actual binary name is `mod_wl_20.so`)
 - Oracle HTTP Server 11g: `mod_wl_ohs` (actual binary name is `mod_wl_ohs.so`)
-
-

To install and configure mod_weblogic

1. **Oracle HTTP Server 10.1.3:** Copy `mod_wl_20.so` to the Oracle HTTP Server modules directory: For example:

From: `WL_HOME/wlserver_10.0/server/plugin/linux/i686`

To: `ORACLE_HOME/ohs/modules`

2. Locate the Oracle HTTP Server `httpd.conf` file. For example:

Oracle HTTP Server 10.1.3:

`ORACLE_HOME/ohs/conf/httpd.conf`

Oracle HTTP Server 11g:

`ORACLE_INSTANCE/config/OHS/<ohs_name>/httpd.conf`

3. Verify that `mod_weblogic` configuration is in `httpd.conf`, either by inclusion of the appropriate configuration file or the configuration itself directly. For example, for Oracle HTTP Server 10g:

```
LoadModule weblogic_module ${ORACLE_HOME}/ohs/modules/mod_wl_20.so
<IfModule mod_weblogic.c>
    WebLogicHost yourHost.yourDomain.com
    WebLogicPort yourWlsPortNumber
```

```

</IfModule>

<Location /request-uri-pattern>
    SetHandler weblogic-handler
</Location>

```

18.1.2.2 Registering Oracle HTTP Server mod_osso with OSSO Server 10.1.4

The mod_osso module is an Oracle HTTP Server module that provides authentication to OracleAS applications. This module resides on the Oracle HTTP Server that enables applications protected by OracleAS Single Sign-On to accept HTTP headers in lieu of a user name and password once the user has logged into the OracleAS Single Sign-On server. The values for these headers are stored in a mod_osso cookie.

The mod_osso module enables single sign-on for Oracle HTTP Server by examining incoming requests and determining whether the requested resource is protected. If it is, then it retrieves the Oracle HTTP Server cookie.

Under certain circumstances, you must register Oracle HTTP Server mod_osso using the 10.1.4 Oracle Identity Manager single sign-on registration tool (ssoreg.sh or ssoreg.bat). [Table 18–2](#) provides a summary of parameters and values for this purpose. Running the tool updates the mod_osso registration record in osso.conf. The tool generates this file whenever it runs.

Table 18–2 *ssoreg Parameters to Register Oracle HTTP Server mod_osso*

Parameter	Description
-oracle_home_path	Path to the 10.1.4 SSO Oracle_Home
-site_name	Any site name to be covered
-config_mod_osso	TRUE. If set to TRUE, this parameter indicates that the application being registered is mod_osso. You must include config_mod_osso for osso.conf to be generated.
-mod_osso_url	URL for front-ending Oracle HTTP Server Host:port. This is the URL that is used to access the partner application. The value should be specified in the URL format: <code>http://oracle_http_host.domain:port</code>
-update_mode	Optional. CREATE, the default, generates a new record.
-remote_middier	Specifies that the mod_osso partner application to be registered is at a remote mid-tier. Use this option only when the mod_osso partner application to be configured is at a different ORACLE_HOME, and the OracleAS Single Sign-On server runs locally at the current ORACLE_HOME.
-config_file	Path where osso.conf is to be generated
[-admin_info	Optional. User name of the mod_osso administrator. If you omit this parameter, the Administrator Information field on the Edit Partner Application page is left blank.
admin_id	Optional. Any additional information, such as email address, about the administrator. If you omit this parameter, the Administrator E-mail field on the Edit Partner Application page is left blank.
<VirtualHost ...>	Host name. Optional. Include this parameter only if you are registering an Oracle HTTP virtual host with the single sign-on server. Omit the parameter if you are not registering a virtual host. If you are creating an HTTP virtual host, use the httpd.conf file to fill in the directive for each protected URL.

See Also: The following books on Oracle Technology Network at:
<http://www.oracle.com/technology/documentation/oim1014.html>

- *Oracle Application Server Single Sign-On Administrator's Guide 10g (10.1.4.0.1)* Part Number B15988-01
- *Oracle Identity Management Application Developer's Guide 10g (10.1.4.0.1)* Part Number B15997-01

The following procedure includes a sample command to register `mod_osso`. Values for your environment will be different.

To register `mod_osso`

1. Go to the following 10.1.4 Oracle Identity Manager directory path:

```
ORACLE_HOME/sso/bin/ssoreg
```

2. Run `ssoreg` with the following parameters and values for your environment. For example, on Unix, this might look like:

```
./ssoreg.sh -oracle_home_path \OraHome -site_name wls_server  
-config_mod_osso TRUE -mod_osso_url http://oracle_http_host.domain:7788  
-update_mode CREATE -remote_middtier -config_file \tmp\osso.conf
```

3. Verify that the module `mod_osso` of the required Oracle HTTP Server is registered.
4. Proceed to "[Configuring mod_osso to Protect Web Resources](#)".

18.1.2.3 Configuring `mod_osso` to Protect Web Resources

`mod_osso` redirects the user to the single sign-on server only if the URL you request is configured to be protected. You can secure URLs in one of two ways: statically or dynamically. Static directives simply protect the application, ceding control over user interaction to `mod_osso`. Dynamic directives not only protect the application, they also enable it to regulate user access.

For more information, see:

- [Configuring mod_osso with Static Directives](#)
- [Protecting URLs and Logout Dynamically \(without mod_osso\)](#)

18.1.2.3.1 Configuring `mod_osso` with Static Directives You can statically protect URLs with `mod_osso` by applying directives to the `mod_osso.conf` file. You must configure `mod_osso` to ensure that requests are intercepted properly. In addition, you specify the location of protected URIs, time out interval, and the authentication method. Oracle recommends that you place in the `httpd.conf` file the include statement for `mod_osso.conf` before the one wherein the `weblogic_module` statement is loaded.

The following procedure describes how to configure `mod_osso` by editing the `mod_osso.conf` file. This procedure provides details for two different releases. Ensure that you follow instructions for your OHS deployment:

- **Oracle HTTP Server 11g:** Requires Step 2 and `AuthType Osso` in Step 4. The path name in Step 5 differs for Oracle HTTP Server 11g.
- **Oracle HTTP Server 10g:** Requires Step 3 and `AuthType Basic` in Step 4. The path name in Step 5 differs for Oracle HTTP Server 10g.

To configure mod_osso to protect Web resources

1. Copy osso.conf from the location where it was generated to the following location:

From: */tmp/osso.conf*

To:

ORACLE_INSTANCE/config/OHS/<ohs_name>/osso/osso.conf

2. **Oracle HTTP Server 11g:** Copy mod_osso.conf from the disabled directory to the moduleconf directory for editing. For example:

From:

ORACLE_INSTANCE/config/OHS/<ohs_name>/disabled/mod_osso.conf

To:

ORACLE_INSTANCE/config/OHS/<ohs_name>/moduleconf/mod_osso.conf

3. **Oracle HTTP Server 10g:** Locate mod_osso.conf for editing. For example:

ORACLE_HOME/ohs/conf/mod_osso.conf

4. Edit mod_osso.conf to add the following information using values for your deployment. For example, using Oracle HTTP Server as an example (paths are different for 10g):

```
LoadModule osso_module ${ORACLE_HOME}/ohs/modules/mod_osso.so
<IfModule mod_osso.c>

OssoIdleTimeout off
OssoIpCheck on
OssoConfigFile ORACLE_INSTANCE/config/OHS/<ohs_name>/osso/osso.conf

#Location is the URI you want to protect
<Location />
require valid-user
#OHS 11g AuthType Osso
#OHS 10g AuthType Basic
AuthType Osso

</Location>

</IfModule>
```

5. Locate the httpd.conf file for editing. For example:

Oracle HTTP Server 10.1.3:

ORACLE_HOME/ohs/config/httpd.conf

Oracle HTTP Server 11g:

ORACLE_INSTANCE/config/OHS/<ohs_name>/httpd.conf

6. In the httpd.conf, confirm that the mod_osso.conf file path for your environment is included. For example:

```
include /ORACLE_INSTANCE/config/OHS/<ohs_name>/moduleconf/mod_osso.conf
```

7. Restart the Oracle HTTP Server.

Tip: If the interception of requests is not working properly, consider placing the include statement for `mod_osso.conf` before the `LoadModule weblogic_module` statement in the `httpd.conf`.

8. Proceed to ["Adding Providers to a WebLogic Domain for OSSO"](#).

18.1.2.3.2 Protecting URLs and Logout Dynamically (without `mod_osso`) Applications that use dynamic directives require no entry in `mod_osso.conf` because `mod_osso` protection is written directly into the application as one or more dynamic directives.

Dynamic directives are HTTP response headers that have special error codes that enable an application to request granular functionality from the single sign-on system without having to implement the intricacies of the single sign-on protocol. Upon receiving a directive as part of a simple HTTP response from the application, `mod_osso` creates the appropriate single sign-on protocol message and communicates it to the single sign-on server.

OracleAS supports dynamic directives for Java servlets and JSPs. The product does not currently support dynamic directives for PL/SQL applications. The JSPs that follow show how such directives are incorporated. Like their "static" counterparts, these sample "dynamic" applications generate user information:

- [Example 18–1, "SSO Authentication with Dynamic Directives"](#)
- [Example 18–2, "SSO Logout with Dynamic Directives"](#)

Note: After adding dynamic directives, be sure to restart the Oracle HTTP Server, and then proceed to ["Adding Providers to a WebLogic Domain for OSSO"](#).

Example 18–1 SSO Authentication with Dynamic Directives

The `home.jsp` includes `ssodynauth.jsp` that uses the `request.getUserPrincipal().getName()` method to check the user in the session. If the user is absent, it issues dynamic directive 499, a request for simple authentication. The key lines are in boldface.

```
//home.jsp

<%@ include file="ssodynauth.jsp" %>
<%
//page content goes here
%>

//ssodynauth.jsp

<%
response.setHeader("Cache-Control", "no-cache");
response.setHeader("Pragma", "no-cache");
response.setHeader("Expires", "0");
%>
<%
// Check for user
String ssoUser = null;
try
(
//ssoUser = request.getRemoteUser();
ssoUser = request.getUserPrincipal().getName( );
ssoUser = ssoUser.trim( );
```

```

    }
    catch(Exception e)
    {
ssoUser = null;
    }

    // If user is not authenticated then generate
    // dynamic directive for authentication
if((ssoUser == null) || (ssoUser.length() < 1))
    {
    response.sendError(499, "Oracle SSO");
    return;
    }%>

```

See Also: *Oracle Identity Management Application Developer's Guide 10g (10.1.4.0.1)* Part Number B15997-01 on Oracle Technology network at:
<http://www.oracle.com/technology/software/products/ias/hdocs/101401.html>

Example 18–2 SSO Logout with Dynamic Directives

To achieve global logout (also known as single log-out), applications are expected to first invalidate sessions and then make a call to OSSO logout. The `logout.jsp` issues dynamic directive 470, a request for OSSO logout. The `osso-return-logout` is set by the application to specify the return URL after logout.

The key lines for SSO logout with dynamic directives appear in boldface in the following example. In 11g, the `SSOFilter` handles session synchronization.

```

//logout.jsp
<%@page session="false"%>
<%
    response.setHeader("Osso-Return-Url", "http://my.oracle.com/");
    HttpSession session = null;
    session = request.getSession();
    if (null != session )
    {
        // necessary for achieving SLO
        session.invalidate();
    }
    response.sendError(470, "Oracle SSO");
}%>

```

See Also:

- "Synchronizing the User and SSO Sessions: SSO Synchronization Filter" on page 18-16
- *Oracle Identity Management Application Developer's Guide 10g (10.1.4.0.1)* Part Number B15997-01 on Oracle Technology Network at:
<http://www.oracle.com/technology/software/products/ias/hdocs/101401.html>

Note: After adding dynamic directives, be sure to restart the Oracle HTTP Server, and then proceed to "Adding Providers to a WebLogic Domain for OSSO".

18.1.2.4 Adding Providers to a WebLogic Domain for OSSO

You must add the OSSO Identity Asserter to a WebLogic domain. In addition to the OSSO Identity Asserter, Oracle recommends the following Authentication Providers:

- OSSO Identity Asserter
- DefaultAuthenticator
- OID Authenticator

See Also: ["About Oracle WebLogic Server Authentication and Identity Assertion Providers"](#) on page 16-19

You can add providers using either the Oracle WebLogic Administration Console or Oracle WebLogic Scripting Tool (WLST) command-line tool.

See Also:

- ["About Oracle WebLogic Server Authentication and Identity Assertion Providers"](#) on page 16-19
- *Oracle Fusion Middleware Oracle WebLogic Scripting Tool*
- *Oracle Fusion Middleware WebLogic Scripting Tool Command Reference*

The following procedure illustrates adding Authentication Providers using the Oracle WebLogic Administration Console. Before you begin, there is a condition to pay attention to:

Step 10: If your application requires the user in the same case as in Oracle Internet Directory (uppercase, lowercase, initial capitals), check **Use Retrieved User Name as Principal**. Otherwise, leave it unchecked.

To add providers to your WebLogic domain for OSSO Identity Assertion

1. Log in to the WebLogic Administration Console.
2. **OSSO Identity Asserter:** Perform the following steps to add this to the domain:
 - a. Click **Security Realms, Default Realm Name, Providers**.
 - b. Select **New** under the Authentication Providers table.
 - c. Enter a name for the new provider, select its type, and then click OK. For example:
Name: *OSSO Identity Asserter*
Type: *OSSOIdentityAsserter*
Ok
 - d. Click the name of the newly added provider.
 - e. On the Common tab, set the appropriate values for common parameters and set the Control Flag to SUFFICIENT and then save the settings.
3. **Default Authentication Provider:**
 - a. Click **Security Realms, Default Realm Name, Providers**.
 - b. Click Default Authentication Provider.
 - c. Set the control flag to OPTIONAL, and click Save

4. **OID Authenticator:** Perform the following steps to add this provider.
 - a. Click **Security Realms, Default Realm Name, Providers**.
 - b. Click New, and enter a name and type:
 - Name: *OID Authenticator*
 - Type: *OracleInternetDirectoryAuthenticator*
 Click Save.
 - c. Click the newly added authenticator to see the Settings page. Retain the default settings; do not change the Control Flag until you have verified that the Oracle Internet Directory configuration is valid.

Note: If OID Authenticator is the only provider, ensure the WebLogic Server user account and its granted group memberships are created in Oracle Internet Directory. Otherwise the WebLogic domain does not start properly.

- d. Click the **Provider Specific** tab and specify the following required settings:
 - Propagate Cause For Login Exception: *Check*
 - Principal: LDAP administrative user. For example: *cn=orcladmin*
 - Host: The Oracle Internet Directory hostname
 - Use Retrieved User Name as Principal: *Check*
 - Credential: LDAP administrative user password. For example: *password*
 - Confirm Credential: For example: *password*
 - Group Base DN: Oracle Internet Directory group search base
 - User Base DN: Oracle Internet Directory user search base.
 - Port: Oracle Internet Directory port
5. **Reorder Providers:** The order in which providers populate a subject with principals is *significant* and you might want to reorder the list of all providers in your realm and bring the newly added provider to the top of the list.
6. Save all configuration settings.
7. Stop and restart the Oracle WebLogic Server for the changes to take effect.
8. Log in to the WebLogic Administration Console:
 - a. Click **Security Realms, Default Realm Name, Providers**.
 - b. Select the **Users and Groups** tab to see a list of users and groups contained in the configured Authentication Providers.
 - You should see usernames from the Oracle Internet Directory configuration, which implicitly verifies that the configuration is working.
 - If the Oracle Internet Directory instance is configured successfully, you can change the Control Flag.
 - If the Oracle Internet Directory authentication is sufficient for an application to identify the user, then choose the SUFFICIENT flag. SUFFICIENT means that if a user can be authenticated against Oracle Internet Directory, no further authentication is processed. REQUIRED means that the Authentication

Provider must succeed even if another provider already authenticated the user.

9. **Application Requires User in Same Case as in Oracle Internet Directory:** Check Use Retrieved User Name as Principal. Otherwise, leave it unchecked.
10. Save the changes.
11. Activate the changes and restart Oracle WebLogic Server.
12. Proceed with ["Establishing Trust Between Oracle WebLogic Server and Other Entities"](#).

18.1.2.5 Establishing Trust Between Oracle WebLogic Server and Other Entities

The Oracle WebLogic Connection Filtering mechanism must be configured for creating access control lists and for accepting requests from only the hosts where Oracle HTTP Server and the front-end Web server are running.

Note: This topic is the same whether you are using OSSO or Oracle Access Manager. In the WebLogic Administration Console.

A *network connection* filter is a component that controls the access to network level resources. It can be used to protect resources of individual servers, server clusters, or an entire internal network. For example, a filter can deny non-SSL connections originating outside of a corporate network. A network connection filter functions like a firewall since it can be configured to filter protocols, IP addresses, or DNS node names. It is typically used to establish trust between Oracle WebLogic Server and foreign entities.

Connection Filter Rules: The format of filter rules differ depending on whether you are using a filter file to enter the filter rules or you enter the filter rules in the Administration Console. When entering the filter rules on the Administration Console, enter them in the following format:

```
targetAddress localAddress localPort action protocols
```

See Also: "Configuring Security in a WebLogic Domain" in *Oracle Fusion Middleware Securing Oracle WebLogic Server*

[Table 18–3](#) provides a description of each parameter in a connection filter.

Table 18–3 Connection Filter Rules

Parameter	Description
target	Specifies one or more systems to filter
localAddress	Defines the host address of the WebLogic Server instance. (If you specify an asterisk (*), the match returns all local IP addresses.)
localPort	Defines the port on which the WebLogic Server instance is listening. (If you specify an asterisk, the match returns all available ports on the server.)
action	Specifies the action to perform. This value must be allow or deny.
protocols	Is the list of protocol names to match. The following protocols may be specified: http, https, t3, t3s, giop, giops, dcom, ftp, ldap. If no protocol is defined, all protocols match a rule.

The Connection Logger Enabled attribute logs successful connections and connection data in the server. This information can be used to debug problems relating to server connections.

To configure a connection filter to allow requests from the host of the 11g Oracle HTTP Server

1. Log in to the Oracle WebLogic Administration Console.
2. Click Domain under Domain Configurations.
3. Click the Security tab, click the Filter tab.
4. Click the Connection Logger Enabled attribute to enable the logging of accepted messages for use when debugging problems relating to server connections.
5. Specify the connection filter to be used in the domain:
 - Default Connection Filter: In the Connection Filter attribute field, specify `weblogic.security.net.ConnectionFilterImpl`.
 - Custom Connection Filter: In the Connection Filter attribute field, specify the class that implements the network connection filter, which should also be specified in the CLASSPATH for Oracle WebLogic Server.
6. Enter the appropriate syntax for the connection filter rules.
7. Click Save.
8. Restart the Oracle WebLogic Server.
9. Proceed to "[Configuring the Application for the OSSO Identity Asserter](#)".

18.1.2.6 Configuring the Application for the OSSO Identity Asserter

This topic describes how to create the application authentication method for the OSSO Identity Asserter.

See Also: *Oracle Fusion Middleware Deploying Applications to Oracle WebLogic Server*

Oracle WebLogic Server supports adding multiple auth-methods. If you are setting up an OSSO Identity Asserter in the WebLogic Application Console, the Web application using the OSSO Identity Asserter must have its auth-method set to `CLIENT-CERT`.

After deploying the application on the Oracle WebLogic Server, all `web.xml` files in the application EAR file must include `CLIENT-CERT` in the element `auth-method` for the appropriate realm, as described in the following procedure.

To edit web.xml for the OSSO Identity Asserter

1. Locate the `web.xml` file in the application EAR file. For example:

```
WEB-INF/web.xml
```

2. Locate the `auth-method` for the appropriate realm and enter `CLIENT-CERT`. For example:

```
<login-config>
  <auth-method>CLIENT-CERT</auth-method>
  <realm-name>myRealm</realm-name>
</login-config>
```

3. Save the file.

4. Redeploy and restart the application.
5. Repeat for each web.xml file in the application EAR file.

18.2 Synchronizing the User and SSO Sessions: SSO Synchronization Filter

In Fusion Middleware 11g, a new component that synchronizes the container user session and SSO session has been introduced. SSO Sync Filter is an Oracle WebLogic system filter implementation that intercepts all requests to the container, acts on protected resource requests, and attempts to synchronize the container's user session with the user identifying header in OSSO (Proxy-Remote-User) or the user data in the Oracle Access Manager SSO session cookie (ObSSOCookie).

SSO Synchronization Filter (SSO Sync Filter) is an implementation of the Servlet Filter based on Java Servlet Specification version 2.3. SSO sync filter relieves applications from tracking the SSO user session and synchronizing it with their respective sessions. Instead, applications would only need to synchronize with container's user session.

SSO Sync Filter intercepts each request to the container and determines whether to act on it based on certain HTTP headers that are attached to the request. Filter expects SSO agent to have set those headers in the Web Tier. When access is made to unprotected areas of the application, the filter acts as a pass through. Once a protected resource is accessed, SSO agents in the Web Tier, direct user to perform authentication with SSO system such as Oracle Access Manager. After the authentication, Oracle Access Manager Identity Asserter helps establish a user identity in form of JAAS Subject to the container and a user session is created. WebLogic maintains the user session data as part of HTTP Session Cookie (JSESSIONID).

Subsequent access to the application resources provides two pieces of information to the SSO Sync Filter:

- User identifying header in OSSO (Proxy-Remote-User)
- User data in the Oracle Access Manager SSO session cookie (ObSSOCookie)

The job of SSO Sync Filter is to make sure that the user identity in the container matches with that of the SSO session. If there is a mismatch, filter invalidates the container's user session. As a result, the downstream application would only have to track container user session and react in a consistent fashion regardless of SSO environment in use.

Notes:

- **Enabled and Active by Default:** SSO Sync Filter fetches the user information from the configured tokens, gets the user from existing session (if any), invalidates the session and redirects to the requested URL in case the CurrentSessionUser does not match the incoming SSO User. Otherwise, the request is simply passed through.

If you have not configured the OSSO or Oracle Access Manager Assertion Providers in your domain, the filter disables automatically during WebLogic Server start-up.

- **Active for All URI's by Default (/*):** No changes are required in the application code.
- **Configured for the OSSO Tokens/Header:** Proxy-Remote-User, and performs a case insensitive match.

- **Configured for the Oracle Access Manager SSO Tokens/Header:** OAM_REMOTE_USER and REMOTE_USER, and does a case insensitive match.
- **Global Logout:** SSO Sync Filter is intended to provide the Single Logout Experience to the Oracle Fusion Middleware applications that use the OSSO or Oracle Access Manager Solutions. Is handled similarly to single sign-on. After global logout is performed, SSO filter reconciles the session when subsequent access to an application that has not cleaned up its session is made.

Any application that use the OSSO or Oracle Access Manager Solutions is expected to invalidate its session before making a call to OSSO logout or Oracle Access Manager logout. For more information on OSSO logout, see "[SSO Logout with Dynamic Directives](#)" on page 18-11. For details about Oracle Access Manager logout, see "[Configuring Global Logout for Oracle Access Manager 10g and 10g WebGates](#)" on page 17-10.

- **Application Session Time Out:** SSO cookies typically track user inactivity/idle times and force users to login when a time out occurs. OSSO and Oracle Access Manager are no exception. Oracle Access Manager takes a sophisticated approach at this and specifically tracks Maximum Idle Session Time and Longest Idle Session Time along with SSO session creation time and time when it was last refreshed.

The general recommendation for applications that are maintaining their own sessions when integrating with SSO systems is to configure their session time outs close to that of SSO session time outs so as to make user experience remains consistent across SSO and application session time outs.

You can alter the behavior of the SSO Sync Filter for application requirements by passing various over-riding system properties to WebLogic. To do this, you change the Oracle WebLogic startup script and check for EXTRA_JAVA_PROPERTIES in setDomainEnv.sh. The properties and Sync behavior is shown in [Table 18-4](#).

Table 18-4 SSO Sync Filter Properties and Sync Behavior

Area	Overriding System Property	Default value of System property	Default Behavior of the Sync Filter
Status (Active or Inactive)	sso.filter.enable	Not configured	Enabled
Case sensitive matches	sso.filter.name.exact.match	Not configured	Case Ignore Match
Configured Tokens	sso.filter.sstoken	Not configured	<ul style="list-style-type: none"> ■ OSSO: Look for Proxy-Remote-User ■ Oracle Access Manager: Look for OAM_REMOTE_USER and REMOTE_USER. OAM_REMOTE_USER takes precedence.
URI Mappings	Not Applicable	Not Applicable	/*

You cannot enable the filter for selected applications. The SSO Sync Filter is a system filter. As such, it is activated for all deployed applications (the URI mapping is /*).

Note: You cannot enable the filter for selected applications.

The following procedure gives some tips about modifying the SSO Sync filter properties and behavior.

To modify the SSO Sync Filter properties and behavior

1. **Disable the Filter:** Change the system property "sso.filter.enable" to "false" (pass as -D to the jvm) and restart the Oracle WebLogic Server. This toggles the filter status.
2. **User-Identifying Header Differs from Pre-Configured Sync Filter Tokens:** Over-ride the SSO token that the Sync Filter looks for using the system property "sso.filter.sstoken".

For example, pass to the WebLogic Server jvm in the WebLogic Server startup script -Dss.filter.sstoken=HEADERNAME, and restart the server.

When you contact Oracle Support you might be requested to set up debugging, as described in "[Setting Up Debugging in the WebLogic Administration Console](#)" on page 15-14.

18.3 Troubleshooting for an OSSO Identity Asserter Deployment

The troubleshooting items described in this section are grouped into the following categories:

- [SSO-Related Problems](#)
- [OSSO Identity Asserter-Related Problems](#)
- [URL Rewriting and JSESSIONID](#)
- [About mod_osso, OSSO Cookies, and Directives](#)
- [About Using IPv6](#)

See Also:

- ["Setting Up Debugging in the WebLogic Administration Console" on page 15-14](#)
- *Oracle Application Server Single Sign-On Administrator's Guide for 10g, Troubleshooting*, on the Oracle Technology Network at: http://www.oracle.com/technology/documentation/oi_m1014.html

18.3.1 SSO-Related Problems

This section addresses the following troubleshooting items:

- [OHS Is Not Redirecting to SSO - Internal Server Error 500](#)
- [Is Attribute AuthName Required?](#)
- [URL Request not Redirected to SSO](#)
- [Error 404 - Not Found is Issued \(OHS Side\)](#)
- [Error 404 - Not Found is Issued \(Oracle WebLogic Server Side\)](#)
- [Oracle SSO Failure - Unable to process request](#)
- [OSSO Solution for Applications Deployed on a Stand-alone WebLogic Server](#)
-

OHS Is Not Redirecting to SSO - Internal Server Error 500

The most likely source of this problem is an incorrect configuration.

The following sample uses Oracle HTTP Server 11g. Path names are different if you have Oracle HTTP Server 10g.

To address it, proceed as follows:

1. Open the file `mod_osso.conf` and ensure that the resource is protected. For example:

```
ORACLE_INSTANCE/config/OHS/<ohs_name>/moduleconf/mod_osso.conf

<Location /protected-resource-uri>
require valid-user
AuthType Basic
</Location>
```

2. Ensure that `osso.conf` is present and included in `mod_osso.conf`. For example, using Oracle HTTP Server 11g (paths are different for 10g)

```
OsoConfigFile ORACLE_INSTANCE/config/OHS/<ohs_name>/osso/osso.conf
```

Note: There is no set location for `osso.conf`. The value is determined at registration time; it can be any absolute path.

3. Ensure that `httpd.conf` includes `mod_osso.conf`. For example, using Oracle HTTP Server 11g (paths are different for 10g):

```
ORACLE_INSTANCE/config/OHS/<ohs_name>/httpd.conf

include /ORACLE_INSTANCE/config/OHS/<ohs_name>/moduleconf/mod_osso.conf
```

4. If all of the above were correctly specified, the SSO registration did not complete successfully and you must re-register SSO.

To register SSO, proceed as follows using the appropriate `ssoreg` tool for your platform. For example:

- a. Run `ssoreg.sh` in 10.1.4 `ORACLE_HOME/sso/bin` to produce the file `osso.conf`. The following is a sample usage of this utility that produces the file in `/tmp/osso.conf` (the arguments are displayed in different lines only for illustration):

```
>ssoreg.sh -oracle_home_path /OraHome
           -site_name wls_server
           -config_mod_osso TRUE
           -mod_osso_url http://host.domain.com:6666
           -update_mode CREATE
           -remote_midtier
           -config_file /tmp/osso.conf
```

- b. Copy the generated `osso.conf` to another file system directory. For example: `ORACLE_INSTANCE/config/OHS/<ohs_name>/osso.`
- c. Restart OHS.

Is Attribute AuthName Required?

Log messages might suggest that the attribute AuthName is required, and certain versions of Apache do require this attribute.

This example uses Oracle HTTP Server 11g. Path names are different for Oracle HTTP Server 10g.

To include this attribute, edit the file `mod_osso.conf` and insert a fragment like the following:

```
LoadModule osso_module modules/mod_osso.so
<IfModule mod_osso.c>
OsoIdleTimeout off
OsoIpCheck on
OsoConfigFile ORACLE_INSTANCE/config/OHS/<ohs_name>/osso/osso.conf

<Location />
AuthName "Oracle Single Sign On"
require valid-user
AuthType Basic
</Location>
</IfModule>
```

URL Request not Redirected to SSO

Once a URL request is issued, if a basic pop-up is displayed instead of being redirected to SSO, then, most likely, the URL request has been intercepted by the Apache authorization module.

To address this problem, proceed as follows:

1. Edit the file `httpd.conf` and comment out the loading authorization modules as illustrated in the following fragment:

```
ORACLE_INSTANCE/config/OHS/<ohs_name>/httpd.conf

LoadModule access_module modules/mod_access.so
#LoadModule auth_module modules/mod_auth.so
#LoadModule auth_anon_module modules/mod_auth_anon.so
#LoadModule auth_db_module modules/mod_auth_dbm.so
LoadModule proxy_module modules/mod_proxy.so
```

2. Restart OHS.

Error 404 - Not Found is Issued (OHS Side)

Typically, this error has the following format:

```
The requested URL <request-uri> was not found on this server
```

Most likely, the WebLogic redirect is not happening, and the request is attempting to grab an OHS resource not available.

To address this problem, verify that `mod_weblogic` is included in the file `httpd.conf` and that the WebLogic handler is set for the request pattern, as illustrated in the following fragment:

```
#httpd.conf
<IfModule mod_weblogic.c>
WebLogicHost <host>
WebLogicPort yourWlsPortNumber
</IfModule>
```

```
<Location /request-uri-pattern>
  SetHandler weblogic-handler
</Location>
```

Error 404 - Not Found is Issued (Oracle WebLogic Server Side)

Typically, this error has the following format:

```
Error 404--Not Found
```

Cause

This message informs that the Oracle WebLogic Server is not able to find a resource.

Solution

To address the problem, check that the resource is indeed deployed on the server. For example, if the pattern is `/private1/Hello`, check that `Hello` is accessible on the server with `private1` as root.

Oracle SSO Failure - Unable to process request

Problem

You receive a message stating:

```
Oracle SSO Failure - Unable to process request
Either the requested URL was not specified in terms of a fully-qualified host
name or Oracle HTTP Server single sign-on is incorrectly configured.
Please notify your administrator.
```

Solution

Modify the Oracle HTTP Server `httpd.conf` file to include a port number in the `ServerName` and restart the Web server. For example:

```
From: ServerName host.domain.com
```

```
To: ServerName host.domain.com:port
```

OSSO Solution for Applications Deployed on a Stand-alone WebLogic Server

This chapter describes how to configure single sign-on (SSO) for applications that are deployed on Oracle Fusion Middleware Oracle WebLogic Server. However, details for applications that are deployed on a stand-alone Oracle WebLogic Server (one without Fusion Middleware) are provided here:

- **Oracle Fusion Middleware with OSSO:** The required OSSO Identity Asserter (`ossoiap.jar`) is provided automatically when you install Oracle Fusion Middleware: Oracle Identity Management, Oracle SOA Suite, or Oracle WebCenter.

Note: Oracle Fusion Middleware with OSSO enables you to use either the Oracle HTTP Server 10g or 11g Web server.

- **Stand-Alone Oracle WebLogic Server with OSSO:** The required OSSO Identity Asserter (`ossoiap.jar`) must be acquired from the Oracle Web Tier, as described here.

Note: Without Fusion Middleware, OSSO requires Oracle HTTP Server 11g.

Whether you use OSSO for Oracle Fusion Middleware applications or other applications, the Identity Asserter performs the same functions as those illustrated and described in "Using the OSSO Identity Asserter".

Included in the following are additional, optional, details that you can use to configure and test Single Logout for session invalidation and synchronization between the SSO cookie and the JSESSIONID cookie. Required files must be acquired from the Oracle Web Tier.

Task overview: Deploying and configuring the OSSO Identity Asserter for applications on a stand-alone WebLogic Server

1. Install Oracle WebLogic Server 10.3.1+ and other required components as follows:
 - a. Perform Step 1, a-d as described in the "[Task overview: Deploying and configuring the OSSO Identity Asserter for applications on a stand-alone WebLogic Server](#)" on page 18-22.
 - b. Skip Step 1e and instead deploy your application.
2. Create a WebLogic security domain with the weblogin domain extension template that is supplied with Oracle WebLogic Server and can be used from `$WLS_HOME/common/bin/config.sh`.
3. Configure `mod_weblogic` to forward requests to Oracle WebLogic Server, as explained in "[Configuring mod_weblogic](#)" on page 18-6.
4. Register and configure the module `mod_osso` with the 10g SSO Server as a partner application, as described in "[New Users of the OSSO Identity Asserter](#)" on page 18-4.
 - a. Perform steps described in "[Registering Oracle HTTP Server mod_osso with OSSO Server 10.1.4](#)" on page 18-7.
 - b. Perform steps described in "[Configuring mod_osso to Protect Web Resources](#)" on page 18-8.
5. Add Authentication Providers to the appropriate security domain as follows:
 - a. Acquire the OSSO Identity Asserter (`ossoiap.jar` from the Oracle Web Tier at: `$ORACLE_INSTANCE/modules/oracle.ossoiap_11.1.1/ossoiap.jar`)
 - b. Copy `ossoiap.jar` into `$WLS_HOME/wlserver_10.x/server/lib/mbeantype`, then restart the Oracle WebLogic Server.
 - c. Configure providers as described in "[Adding Providers to a WebLogic Domain for OSSO](#)" on page 18-12.
6. Configure the Oracle WebLogic Connection Filtering mechanism to create access control lists and accept requests from the hosts where Oracle HTTP Server and the front-end Web server are running, as explained in "[Establishing Trust Between Oracle WebLogic Server and Other Entities](#)" on page 18-14.

Note: Test the secured application to ensure that it is working with the default authenticator using the Oracle WebLogic Server host and port.

7. Configure the application authentication method for the OSSO Identity Asserter (all `web.xml` files in the application EAR file must include `CLIENT-CERT` in the element `auth-method`), as explained in "[Configuring the Application for the OSSO Identity Asserter](#)" on page 18-15.

Note: Test the application with users authenticated by OSSO while accessing the application with the Oracle HTTP Server host and port.

8. **Optional:** You can configure and test Single Logout [Session Invalidation and synchronization between the SSO cookie and JSESSIONID cookie] as follows:

See Also: "[Synchronizing the User and SSO Sessions: SSO Synchronization Filter](#)" on page 18-16 for details on SSOFilter

- a. Acquire `ssofilter.jar` and configure Oracle WebLogic Server to use it as follows:

1. Acquire `ssofilter.jar` from the Oracle Web Tier at:

```
$ORACLE_INSTANCE/modules/oracle.ssofilter_11.1.1/ssofilter.jar
```

2. Copy it to an appropriate directory in Oracle Middleware home: `WLS_INSTALL/Oracle/Middleware/modules` directory, for example.

3. Add the absolute path of `ssofilter.jar` to the Oracle WebLogic Server classpath (by editing the `setDomainEnv.sh` script `POST_CLASSPATH` variable or `CLASSPATH` variable).

- b. Deploy `system-filters.war` as a system filter, as follows:

1. Acquire `system-filters.war` from the Oracle Web Tier at:

```
$ORACLE_INSTANCE/modules/oracle.jrf_11.1.1/system-filters.war
```

2. Copy `system-filters.war` to an appropriate directory in Oracle Middleware home: `WLS_INSTALL/Oracle/Middleware/modules` directory, for example.

3. Deploy `system-filters.war` as an application library: From the WebLogic Administration Console, click `Deployment`, select `New`, and choose the location of file.

4. Restart the Oracle WebLogic Server, if asked.

- c. Enable Logs to verify that the SSOFilter is working, as follows:

1. From the WebLogic Administration Console, click `Domain`, `Environment`, `Servers`, `AdminServer`.

2. Click the `Logging` tab.

3. From the `Advanced` drop-down, select "`Minimum Severity to Log`" as "`Debug`".

4. From the `Advanced` drop-down, "`Message destinations`", select `LogFile`: `Severity Level` as "`Debug`".

5. Save changes and restart the Oracle WebLogic Server.

- d. Confirm that the SSOFilter is loaded as a system filter:

1. Open the `AdminServer.log` file in

```
DomainHome/Servers/AdminServer/log/AdminServer.log.
```

2. Search for "SSOFilter" and confirm that you can see <Debug> messages, which indicate SSOFilter initialization and confirm a filter load
- e. Test the filter functionality to confirm that the SSO and JSESSIONID cookie are cleaned up after user logout and that attempts to access the application after logout require another login.

Note: You must have OSSO Identity Asserter configured in the WebLogic security domain, otherwise the filter will automatically disable during its initialization.

- f. Test logout with applications to confirm that the session ends cleanly.

SSO Users Specified in "Users to Always Audit" Must Be Uppercase

When you specify SSO users in the Oracle HTTP Server audit configuration "Users to Always Audit" section, the SSO username must be specified in uppercase characters.

A comma-separated list of users can be specified to force the audit framework to audit events initiated by these users. Auditing occurs regardless of the audit level or filters that have been specified. This is true for all authentication types.

For more information, see "Managing Audit Policies" in the chapter "Configuring and Managing Auditing" in the *Oracle Fusion Middleware Application Security Guide*.

18.3.2 OSSO Identity Asserter-Related Problems

This section addresses the following troubleshooting items:

- [Error 403 - Forbidden](#)
- [Error 401 - Unauthorized](#)
- [OSSO Identity Assertion Not Getting Invoked](#)

Error 403 - Forbidden

This message informs that the user does not have the required permission to access a resource. This message is shown, for example, when the application has been configured to allow access to users belonging to WLS Group SSOUsers and the asserted user belongs to a different group.

If you have verified that this is not a permissions issue, then check whether the JAAS Control Flag for the Default Identity Authenticator is set to REQUIRED, and if so, change the setting to OPTIONAL or to SUFFICIENT, as appropriate.

Error 401 - Unauthorized

This message informs that the access to a resource requires the user to be first authenticated.

Solution

1. Check that the user is indeed authenticated.
2. Check whether the server is being hit without first going through authentication using SSO.

OSSO Identity Assertion Not Getting Invoked

Situations in which the OSSO Identity Asserter is not getting invoked for a protected source, typically, involve incorrect configuration. Make sure that your environment accurately includes a configuration as that described in "[Configuring the Application for the OSSO Identity Asserter](#)" on page 18-15.

18.3.3 URL Rewriting and JSESSIONID

In some cases when an application resource (URL) is accessed and the JSESSIONID cookie is not found, WebLogic Server rewrites the URL by including the JSESSIONID as part of the URL. If the URL in question is protected, Oracle Access Manager and OSSO Web agents might have issues matching the re-written URL.

To avoid issues of a mismatch, you can append an asterisk, *, to the end of the protected resource specified in mod_osso.conf. For example, if the protected URL is:

```
/myapp/login
```

The location in the mod_osso entry would be:

```
<Location /myapp/login*>
valid user
AuthType OSSO
</Location>
```

18.3.4 About mod_osso, OSSO Cookies, and Directives

Mod_osso module provides communication between the SSO-enabled login server and the Oracle HTTP Server listener. The mod_osso module is controlled by editing the mod_osso.conf file:

- Oracle HTTP Server 11g installation includes mod_osso and mod_weblogic.
- OHS 10g, available in the companion CD release Oracle HTTP Server 10.1.3, includes mod_osso.

See Also: The following topic and Release 1 (11.1.1) manuals

- "[Configuring mod_osso to Protect Web Resources](#)" on page 18-8
- *Oracle Fusion Middleware Installation Guide for Web Tier*
- *Oracle Fusion Middleware Administrator's Guide for Oracle HTTP Server*

This section provides the following information:

- [New OssoHTTPOnly Directive in mod_osso](#)
- [OssoSecureCookies Directive in mod_osso](#)
- [Mod_osso Does Not Encode the Return URL](#)
- [mod_osso: "Page Not found" error After Default Installation](#)

18.3.4.1 New OssoHTTPOnly Directive in mod_osso

A new configuration directive has been added to mod_osso to configure setting the HTTPOnly flag on OSSO cookies. The new Directive is: OssoHTTPOnly. Values are On (to enable) and Off (to disable) the flag. By default, the HTTPOnly flag is set to On; the directive is not set in the configuration.

This directive appends the `HttpOnly` flag to the OSSO cookies set in the browser. This purpose of this flag is to prevent cross-site scripting. Cookies that have this flag set are not accessible by javascript code or applets running on the browser. Cookies that have this flag set is only sent to the server that set the cookie for the particular domain across over `http` or `https`.

This is a per `VirtualHost` directive. It can only be set at the global scope or inside a `VirtualHost` section. The following example shows the new directive:

```
<VirtualHost *.7778>
OssConfigFile conf/osso.conf
OssHttpOnly On
---
---
---
<Location /osso>
AuthType Oso
---
---
</Location>

</VirtualHost>
```

18.3.4.2 OssoSecureCookies Directive in mod_osso

In `mod_osso 10g`, the `OssoSecureCookies` directive is disabled by default. However, in `mod_osso 11g`, this behavior is enabled by default. In `mod_osso 11g`, to disable the `OssoSecureCookies` directive you must set `OssoSecureCookies` to `Off` in the corresponding configuration file. When `mod_osso` is enabled, the `mod_osso.conf` file is available at:

```
ORACLE_INSTANCE/config/OHS/<ohs_name>/moduleconf/mod_osso.conf
```

Set the `OssoSecureCookies` directive as follows:

```
OssoSecureCookies "Off"
```

18.3.4.3 Mod_osso Does Not Encode the Return URL

`Mod_osso` does not encode the return URL in the query when redirecting to the Oracle SSO Server for logout.

To fix this issue, the encoded URL must be passed. For example:
`response.setHeader("Osso-Return-Url", encoded-url)`

18.3.4.4 mod_osso: "Page Not found" error After Default Installation

The following causes might result in a "Page Not Found" error when trying to display SSO page:

- Multiple routing relationships with the same OHS in the absence of load balancer: This is not supported.
- No routing relationship

Solutions: Multiple Routing Relationships

Locate and remove the extra routing relationship that is not related to this `oc4j_im`. Leave the routing relationship that is related to this `oc4j_im`.

1. Use the following command to display all routing relationships in your environment:

```
asctl:/imha/inst1/ohs_im>ls -a -l
oc4j_im_ohs_im_routing_relationship -> /imha/inst12/oc4j_im
oc4j_im_ohs_im_routing_relationship_ -> /imha/inst11/oc4j_im
```

2. Remove the routing relationship that is not related to this specific oc4j_im using the following command with values for your environment. For example:

```
asctl:/imha/inst1/ohs_im> rmrel(name='oc4j_im_ohs_im_routing_relationship_
',pt='/imha/inst11/oc4j_im')
```

3. Stop and start both OHS Web server and oc4j_im.
4. Confirm that the SSO page displays.

Solutions: No Routing Relationships

By default, the installer creates a routing relationship between each OHS and each oc4j_im. If there is no routing relationship between OHS and oc4j_im, you must create one.

1. Use the following command to create a routing relationship using values for your environment:

```
createRoutingRelationship(name='rr1',ut='/imha/inst1/ohs_im',pt='/imha/inst12/
@ oc4j_im')
```

2. Stop and start both OHS Web server and oc4j_im.
3. Confirm that the SSO page displays.

18.3.5 About Using IPv6

Oracle Fusion Middleware supports Internet Protocol Version 4 (IPv4) and Internet Protocol Version 6 (IPv6.) Among other features, IPv6 supports a larger address space (128 bits) than IPv4 (32 bits), providing an exponential increase in the number of computers that can be addressable on the Web.

See Also: *Oracle Fusion Middleware Administrator's Guide* for details about using IPv6 with the Oracle Single Sign-on Server.

Part V

Developing with Oracle Platform Security Services APIs

This part explains how to develop custom security solutions in your applications using OPSS APIs, and it contains the following chapters:

- [Chapter 19, "Integrating Application Security with OPSS"](#)
- [Chapter 20, "The OPSS Policy Model"](#)
- [Chapter 21, "Manually Configuring Java EE Applications to Use OPSS"](#)
- [Chapter 22, "Authentication for Java SE Applications"](#)
- [Chapter 24, "Developing with the Credential Store Framework"](#)
- [Chapter 23, "Authorization for Java SE Applications"](#)
- [Chapter 25, "Developing with the User and Role API"](#)

Integrating Application Security with OPSS

This chapter describes a number of security-related use cases and the typical life cycle of an ADF application security. It also lists code and configuration samples presented elsewhere in this Guide.

This chapter contains the following sections:

- [Introduction](#)
- [Security Integration Use Cases](#)
- [Some Use Cases Details](#)
- [Appendix - Security Life Cycle of an ADF Application](#)
- [Appendix - Code and Configuration Examples](#)

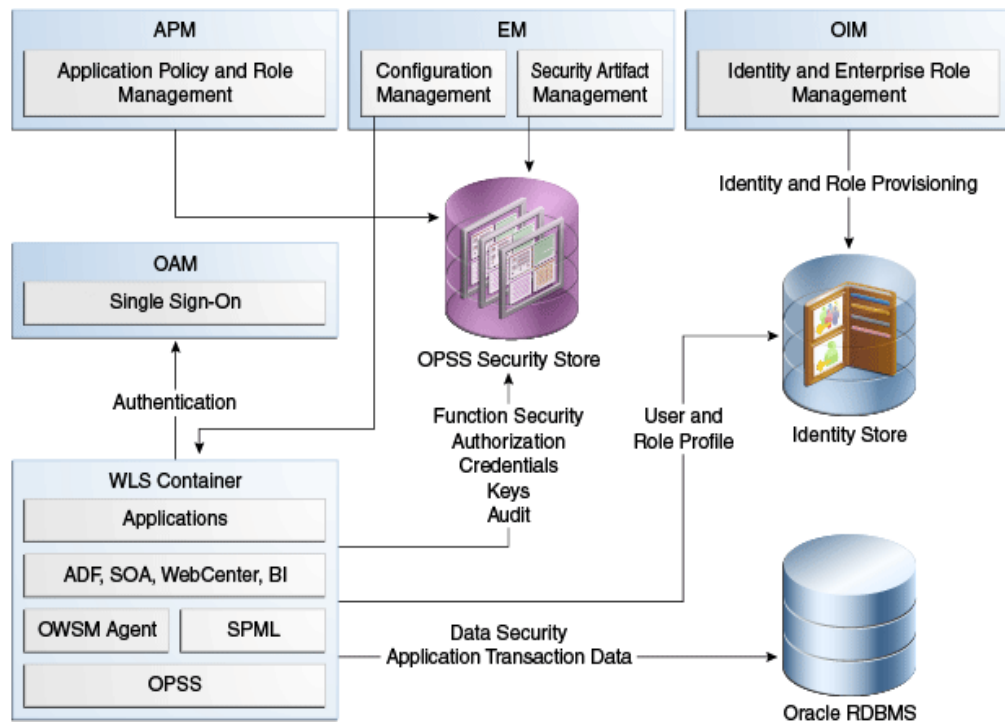
19.1 Introduction

The audience for the material presented in this chapter are developers, security architects, and security administrators. The presentation is not feature-driven, as in most topics in this Guide, but use case-driven: a number of use cases that solve typical application security challenges are introduced as a departing point to solve particular application security requirements. Some of the use cases describe a declarative approach (and do not require changes in application code); others provide a programmatic approach; and others require both approaches.

The top security issues that security architects and developers face include managing users, user passwords, and access to resources. OPSS is a suite of security services that provides solutions to these challenges by supporting:

- Externalizing security artifacts and the security logic from the application
- A declarative approach to security
- A complete user identity life cycle
- Policy-driven access controls

[Figure 19-1](#) illustrates how applications access the security stores and the tools to manage those stores.

Figure 19–1 Applications, Security Stores, and Management Tools**Links to Related Documentation**

Topics explained elsewhere include the following:

- The OPSS Security Architecture - see [Section 1.2, "OPSS Architecture Overview."](#)
- Single Sign On - see [Part IV](#).
- ADF applications - see *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.
- Oracle Development Tools - see *Oracle Fusion Middleware Reference for Oracle Security Developer Tools*.

For the list of OPSS APIs, see [Appendix H, "References."](#)

19.2 Security Integration Use Cases

This section introduces a number of use cases categorized according to a main security feature or security artifact, in the following sections:

- [Authentication](#)
- [Identities](#)
- [Authorization](#)
- [Credentials](#)
- [Audit](#)
- [Identity Propagation](#)
- [Administration and Management](#)
- [Integration](#)

Each use case contains a brief description of the problem it attempts to solve, the security artifacts required, the features involved, and links to details solving the stated problem. Unless otherwise stated, all the descriptions apply to the Oracle WebLogic Application Server and to the WebSphere Application Server.

19.2.1 Authentication

The authentication use cases are the following:

- [Java EE Application Requiring Authenticated Users](#) - Users must be authenticated in order to access a Java EE application.
- [Java EE Application Requiring Programmatic Authentication](#) - Java EE application requires authenticating a user programmatically.
- [Java SE Application Requiring Authentication](#) - Java SE application requires authenticating against a domain identity store.

19.2.1.1 Java EE Application Requiring Authenticated Users

In order to access a Java EE application, users must be authenticated against the identity store in cases where the identity store is any of the following:

- Single LDAP-based store
- Several LDAP-based stores of the same kind (such as all OID, for example)
- Several LDAP-based stores of different kinds; in particular two LDAP-based stores: one AD LDAP and a second one OID LDAP
- Single DB-based store
- Several LDAP- and DB-based stores

This use case requires:

- Allowing access to the application to only authenticated users
- Not modifying the application code, even when customers have user identities in different repositories

This use case features:

- Deploying an application to a WebLogic container
- Configuring the appropriate authenticators according to the particular set of user repositories
- Configuring the OVD authenticator in case of a mixed LDAP types or mixed LDAP and DB types

According to the repository used, the details of this use case are split into the following scenarios:

- Single user repository - Configure the appropriate authenticator with the WebLogic console
- Multiple user repositories (or split profiles across LDAP of the same of different kinds) - Configure the OVD authenticator
- DB-based repositories - Configure the OVD authenticator

For details, see [Section 3.1.2, "Oracle WebLogic Authenticators."](#)

19.2.1.2 Java EE Application Requiring Programmatic Authentication

A Java EE application, not using deployment descriptors, must authenticate the user programmatically against the configured identity store(s); it applies only to Java EE applications deployed to the Oracle WebLogic Application Server.

This use case requires using the OPSS public API to authenticate a user, and it features:

- Configuring authenticators for a Java EE container
- Using the LoginService API to authenticate the user

For details about this use case, see [Section 22.1, "Links to Authentication Topics for Java EE Applications."](#)

19.2.1.3 Java SE Application Requiring Authentication

A Java SE application must authenticate users against the LDAP identity store in use in a domain; the application code requesting authentication must be same regardless of the specifics of the domain's identity store.

This use case requires configuring the identity store(s) against which the authentication should take place and using the LoginService; note that a Java SE application can use only one id login module.

For details about this use case, see [Section 22.2.4, "Using the OPSS API LoginService in Java SE Applications."](#)

19.2.2 Identities

The identity use cases are the following:

- [Application Running in Two Environments](#) - Application, running in two different environments, needs to access user profile information in an LDAP-based store.
- [Application Accessing User Profiles in Multiple Stores](#) - Application needs to access user profile information stored in multiple LDAP-based stores.

19.2.2.1 Application Running in Two Environments

An application, which runs in two different environments, needs to access user profile information, such as a user's email address, stored in an LDAP-based store; the LDAP server can be of any of the supported types and that type may differ with the environment. For details on supported types, see [Section 4.1, "Supported LDAP-, DB-, and File-Based Services."](#)

More specifically, this use case assumes that:

- The application uses the method `UserProfile.getEmail()`.
- In one environment, there is an AD LDAP configured as follows:

```
mail.attr = msad_email
```

- In the second environment, there is an OID LDAP configured as follows:

```
mail.attr = mail
```

In order for the application to retrieve the correct information without modifying the code and regardless of the environment (first or second) in which it runs, the identity store provider must be configured with the correct property in each of those two environments.

In the first environment (AD LDAP), the identity store provider is set to have the following property:

```
<property name="mail.attr" value="msad_mail">
```

In the second one (OID LDAP), the identity store provider is set to have the following property:

```
<property name="mail.attr" value="mail"
```

For details about this use case, see [Section 7.2, "Configuring the Identity Store Provider."](#)

19.2.2.2 Application Accessing User Profiles in Multiple Stores

An application needs access to user profile information located in more than one LDAP-based stores.

This use case requires configuring the environment for multiple LDAP-based stores.

For details about:

- [Configuring multiple LDAPs](#), see [Section 7.3.2.6, "Examples of the Configuration File"](#)
- [Configuring the identity store service](#), see [Section 7.3, "Configuring the Identity Store Service"](#)

19.2.3 Authorization

The authorization use cases are the following:

- [Java EE Application Accessible by Specific Roles](#) - Java EE application accessible only by users configured in web descriptors.
- [ADF Application Requiring Fine-Grained Authorization](#) - ADF application requires fine-grained authorization.
- [Web Application Securing Web Services](#) - Web services application requires securing web services.
- [Java EE Application Requiring Codebase Permissions](#) - Java EE application requires codebase permissions.
- [Non-ADF Application Requiring Fine-Grained Authorization](#) - Non-ADF application requires fine-grained authorization.

19.2.3.1 Java EE Application Accessible by Specific Roles

A Java EE application needs to be accessible only by users that had been assigned specific roles in web descriptors; the group-to-role assignment must be configurable at deployment based on the customer's environment.

For details about this use case, see sections [Using Declarative Security with Web Applications](#) and [Using Declarative Security with EJBs](#) in *Oracle Fusion Middleware Programming Security for Oracle WebLogic Server*.

19.2.3.2 ADF Application Requiring Fine-Grained Authorization

An ADF application in container requires fine-grained authorization at the level of individual controls on the pages in the web application; while the application initiates the authorization check, the policies need to be externalized and customizable per customer post application deployment.

For details on how to develop and secure Oracle ADF applications, see chapter 30, *Enabling ADF Security in a Fusion Web Application*, in *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

For general information about ADF applications, see [Section 1.5.2, "Scenario 2: Securing an Oracle ADF Application."](#)

For details about the life cycle of an ADF application, see [Appendix - Security Life Cycle of an ADF Application](#).

19.2.3.3 Web Application Securing Web Services

A web application requires securing web services with fine grained policies.

For details about web services security administration, see *Oracle Fusion Middleware Security and Administrator's Guide for Web Services*.

19.2.3.4 Java EE Application Requiring Codebase Permissions

A Java EE application requires codebase permissions to perform specific actions; typical examples are reading a credential from the credential store or looking up policies in the policy store.

For details about creating codebase policies with Fusion Middleware Control, see [Section 9.2.3, "Managing System Policies."](#)

19.2.3.5 Non-ADF Application Requiring Fine-Grained Authorization

A non-ADF application needs to be secured with fine-grained authorization checks.

This use case requires:

- Placing checks in the application code at the appropriate places
- Configuring the appropriate policies

For details see [Section 20.3, "The JAAS/OPSS Authorization Model."](#)

19.2.4 Credentials

The credential use case is the following:

- [Application Requiring Credentials to Access System](#) - Application requires credentials to access a back-end system.

19.2.4.1 Application Requiring Credentials to Access System

An application requires a credential to connect to a back-end system, such as a database or an LDAP server. The application code should reference this credential in such a way that the specifics of the credential can be changed per customer post deployment without modifying the application code. Furthermore, this use case also requires specifying who can access the credential store and what operations an authorized user can perform on credential data.

This use case features:

- Using the credential store to persist credentials
- Fetching credentials at runtime with the CSF API in application code
- Defining and enforcing system policies on codebase

For details about:

- Configuration and code examples, see [Section 24.3, "Setting the Java Security Policy Permissions,"](#) and [Section 24.7, "Examples"](#)
- Credential management, see [Section 10.3, "Managing the Credential Store"](#)
- Packaging, see [Section 21.3.2, "Packaging Credentials with Application."](#)

19.2.5 Audit

The audit use cases are the following:

- [Auditing Security-Related Activity](#) - An application requires recording security-related activity.
- [Auditing Business-Related Activity](#) - An application requires recording business activity in the context of a flow.

19.2.5.1 Auditing Security-Related Activity

An application needs to record security-related activity in several security areas; specifically, the application requires logging the following information:

- Changes to a policy: what and when
- The policies that were evaluated in a particular time interval
- Changes to credentials or keys: what and when

The settings explained in this use case apply to all applications and components in a domain.

This use case requires that auditable applications:

- Integrate with the Common Audit Framework (CAF)
- Have built-in capabilities to log security activities
- Set the proper audit filter level to capture activities in specific security areas

This use case features:

- Integrating with the Common Audit Framework
- Allowing applications to define their own audit categories and events in security areas, and making the application audit-aware
- Allowing applications to set the appropriate filter level

For details about:

- Integrating with CAF, see [Section 28.4, "Integrating the Application with the Audit Framework."](#)
- Registering applications, see [Section 28.6, "Register Application with the Registration Service."](#)
- Log audit events, see [Section 28.7, "Add Application Code to Log Audit Events."](#)

19.2.5.2 Auditing Business-Related Activity

An application needs to record business-related activity in the context of a functional flow; specifically, the application requires logging the users and the business actions performed by them in a particular time interval.

The settings explained in this use case apply to all applications and components in a domain.

This use case requires that applications:

- Create their own audit events based on their business needs
- Be able to log business activities with runtime attributes to audit data repository
- Generate audit reports from audit events
- Manage runtime audit policies
- Modify audit event definitions, if necessary

This use case features:

- Allowing applications to define business functional areas (as audit categories), business activities (as audit events in categories), and attributes in each category.
- Registering applications at deployment; updating audit definitions; deregistering applications after deployment.
- Managing audit artifacts with Fusion Middleware Control or WSLT scripts.

For details about:

- Integrating with CAF, see [Section 28.4, "Integrating the Application with the Audit Framework."](#)
- Registering applications, see [Section 28.6, "Register Application with the Registration Service."](#)
- Log audit events, see [Section 28.7, "Add Application Code to Log Audit Events."](#)
- A sample `component_events.xml` file, see [Section 28.5, "Create Audit Definition Files."](#)
- Managing audit policies, see [Section 13.3, "Managing Audit Policies."](#)

19.2.6 Identity Propagation

The identity propagation use cases are the following:

- [Propagating the Executing User Identity](#) - Propagating the executing user identity to a web service over SOAP.
- [Propagating a User Identity](#) - Propagating a user identity to a web service over SOAP.
- [Propagating Identities Across Domains](#) - Propagating a user identity across WebLogic domains.
- [Propagating Identities over HTTP](#) - Propagating a user identity over HTTP.

19.2.6.1 Propagating the Executing User Identity

A client application in container needs to propagate the executing user identity to a web service over SOAP; the web service can be running on a different managed server, in the same domain, or in a different domain.

This use case requires that the current executing user identity be propagated to a web service over SOAP.

The features that facilitate this use case are primarily those of Oracle Web Services Manager (OWSM).

For details about OWSM, see chapter 4, Examining the Rearchitecture of Oracle Web Services Manager in Oracle Fusion Middleware, in Oracle Fusion Middleware Security and Administrator's Guide for Web Services.

For details about propagating identities over SOAP, see chapter 11, *Configuring Policies*, in *Oracle Fusion Middleware Security and Administrator's Guide for Web Services*.

19.2.6.2 Propagating a User Identity

A client application in container needs to propagate a user identity (which is not the executing user identity) to a web service over SOAP; the identity to be propagated is stored in the OPSS security store.

This use case requires that an identity of a user, distinct from the current executing user, be propagated to a web service over SOAP.

This use case features:

- The OPSS security store, where credentials are stored, from where the application gets the specific identity that needs to be propagated as a PasswordCredential.
- Oracle Web Services Manager ability to fetch and propagate the identity to a remote web service.

For details about this use case, see chapter 9, *Creating and Managing Policies Sets*, in *Oracle Fusion Middleware Security and Administrator's Guide for Web Services*.

19.2.6.3 Propagating Identities Across Domains

A client application in container in a WebLogic domain needs to propagate a user identity (stored in the OPSS security store) to a different WebLogic domain over RMI.

For details about this use case, see section *Enabling Trust Between WebLogic Server Domains in shar*.

19.2.6.4 Propagating Identities over HTTP

A client application in container (in a WebLogic domain or a WAS cell) needs to propagate identities over HTTP.

For requirements and details about this use case, see [Propagating Identities over HTTP](#).

19.2.7 Administration and Management

The administration use cases are the following:

- [Application Requiring a Central Store](#) - Application requires a central repository of security artifacts where those artifacts are managed.
- [Application Requiring Custom Management Tool](#) - Application requires a custom tool to manage a central repository of externalized security artifacts.
- [Application Running in a Multiple Server Environment](#) - Application requires modifying security artifacts in a multiple node server environment.

19.2.7.1 Application Requiring a Central Store

An application requires a central repository of policies, credentials, audit configuration, trusts, and keys, and a set of tools to manage that central repository, which is the OPSS security store.

This use case features:

- The OPSS security store
- Managing security artifacts with Fusion Middleware Control

- Managing security artifacts with WLST scripts

For details about:

- The OPSS security store, see [Section 8.1, "Introduction to the OPSS Security Store."](#)
- Managing security artifacts, see:
 - [Section 9.2, "Managing Policies with Fusion Middleware Control"](#)
 - [Section 9.3, "Managing Application Policies with OPSS Scripts"](#)
 - [Section 10.4, "Managing Credentials with Fusion Middleware Control"](#)
 - [Section 10.5, "Managing Credentials with OPSS Scripts"](#)
 - [Chapter 11, "Managing Keys and Certificates with the Keystore Service"](#)

19.2.7.2 Application Requiring Custom Management Tool

An application requires a custom tool to manage externalized security artifacts in a context that is meaningful to the application's business.

This use case requires building a custom graphical user interface with calls to OPSS APIs to display and manage security artifacts in the OPSS security store in a context that is meaningful to the application.

This use case features:

- Managing security artifacts with OPSS API

For details about:

- Code sample illustrating the use of the OPSS API to implement some of the operations needed to manage security artifacts, see [A Custom Graphical User Interface](#).
- The list of OPSS APIs, see [Appendix H, "References."](#)

19.2.7.3 Application Running in a Multiple Server Environment

Application running in a WebLogic domain where several server instances are distributed across multiple machines requires modifying security artifacts; changes must take effect in all components of the application regardless of where they are running.

This use case features:

- Propagating changes to security artifacts whenever those changes are initiated on the administration server; data on managed server nodes is refreshed based on caching policies.
- Using the MBeans API or Management API to modify security artifacts.

For details about:

- Multiple server nodes, see [Section 8.2.1, "Multiple-Node Server Environments"](#)
- OPSS services and MBeans, see [Appendix E.2, "Configuring OPSS Services with MBeans"](#)

19.2.8 Integration

The integration use case is the following:

- [Application Running in Multiple Domains](#) - Several WebLogic domains sharing a single repository of security artifacts.

19.2.8.1 Application Running in Multiple Domains

A product requires multiple WebLogic domains to run and those domains share a single central OPSS security store.

This use case features:

- OPSS support for several domains to share a security store

For details about:

- Domains sharing a credential store, see [Section 10.2, "Encrypting Credentials"](#)
- Using `reassociateSecurityStore` to join to an existing OPSS security store, see [Section 9.3.29, "reassociateSecurityStore"](#)

19.3 Some Use Cases Details

This section describes the following use cases in some detail:

- [Propagating Identities over HTTP](#)
- [A Custom Graphical User Interface](#)

19.3.1 Propagating Identities over HTTP

This section explains how an identity can be propagated across containers and domains using the OPSS trust service and the HTTP protocol.

- [The OPSS Trust Service](#)
- [Propagating Identities over the HTTP Protocol](#)
- [Domains Using Both Protocols](#)

19.3.1.1 The OPSS Trust Service

The OPSS trust service allows the propagation of identities across HTTP-enabled applications by providing and validating tokens. The OPSS trust service uses an asserter that is available only on the following platforms:

- Oracle WebLogic Application Server - the Identity Asserter
- IBM WebSphere Application Server - the Trust Asserter Interceptor (TAI)

Even though the scenarios in this section are illustrated with applications running on WebLogic domains, they also apply to applications running on WebSphere cells; except for the asserter configuration, all other configurations and samples are identical on both platforms. For configuration properties, see [Section F.2.6, "Trust Service Properties."](#)

There is one asserter per WebLogic domain or WebSphere cell; the keystore stores digital certificates, private keys, and trusted CA certificates; the storage service used by the keystore is JKS.

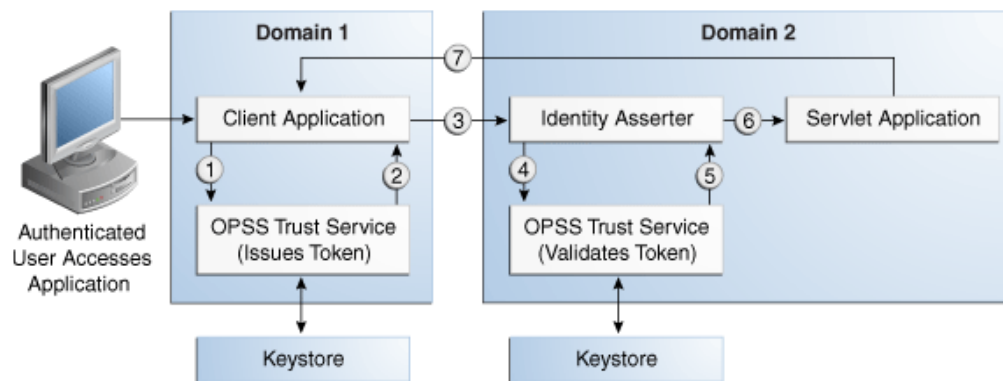
19.3.1.2 Propagating Identities over the HTTP Protocol

Identity propagation using HTTP calls typically runs as follows (see [Figure 19–2](#)):

1. A client application in Domain1 requests a token for an authenticated user from Domain1's OPSS trust service instance.
2. The trust service accesses Domain1's keystore and issues a token to the client application.

3. The client application encodes the token in an HTML header and dispatches an HTTP request to a servlet application in Domain2. Domain 2's asserter intercepts the request and extracts the token.
4. The asserter requests a validation of that token from Domain2's OPSS trust service instance.
5. The trust service accesses Domain2's keystore to validate the token and returns a response.
6. Assuming that the validation is successful, the asserter sends the request to the servlet application using the asserted identity.
7. The servlet application sends an HTTP response to the client application request.

Figure 19–2 Identity Propagation with HTTP Calls



The remainder of this section explains and illustrates the configuration required for the above scenario to work, in the following sections:

- [Single Domain Scenario](#)
- [Multiple Domain Scenario](#)

19.3.1.2.1 Single Domain Scenario In this scenario, the client and the servlet applications use the same trust service instance to issue and validate tokens. The following code and configuration samples illustrate a sample client and a servlet applications running in the same domain.

Client Application Code Sample

The following sample illustrates a client application; note that the file `jps-api.jar` must be included the class path for the code to compile.

```
// Authentication type name
public static final String AUTH_TYPE_NAME = "OIT";
// The authenticated username
String user = "weblogic";
// URL of the target application
URL url = "http://host:port/destinationApp";
//-----
JpsContextFactory ctxFactory = JpsContextFactory.getContextFactory();
JpsContext jpsCtx = ctxFactory.getContext();
final TrustService trustService = jpsCtx.getServiceInstance(TrustService.class);
final TokenManager tokenMgr = trustService.getTokenManager();
final TokenContext ctx = tokenMgr.createTokenContext(
    TokenConfiguration.PROTOCOL_EMBEDDED);
UsernameToken ut = WSSTokenUtils.createUsernameToken("wsuid", user);
```

```

GenericToken gtok = new GenericToken(ut);
ctx.setSecurityToken(gtok);
ctx.setTokenType(SAML2URI.ns_saml);
Map<String, Object> ctxProperties = ctx.getOtherProperties();
ctxProperties.put(TokenConstants.CONFIRMATION_METHOD,
    SAML2URI.confirmation_method_bearer);

AccessController.doPrivileged(new PrivilegedAction<String>() {
    public String run() {
        try {
            tokenMgr.issueToken(ctx);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }
});

Token token = ctx.getSecurityToken();
String b64Tok = TokenUtil.encodeToken(token);

URLConnection connection = (URLConnection) url.openConnection();
connection.setRequestMethod("GET");
connection.setDoOutput(true);
connection.setReadTimeout(10000);
connection.setRequestProperty("Authorization", AUTH_TYPE_NAME + " " + b64Tok);
connection.connect();
BufferedReader rd = new BufferedReader(new InputStreamReader(
    connection.getInputStream()));
StringBuilder sb = new StringBuilder();

String line = null;
while ((line = rd.readLine()) != null) {
    sb.append(line);
}
connection.disconnect();
System.out.println(sb.toString());

```

Keystore Service Configuration

Assuming that the domain name is `jrfServer_admin`, the following command illustrates the creation of the domain keystore, represented by the generated file `default-keystore.jks`:

```

JAVA_HOME/bin/keytool -genkeypair
    -alias jrfServer_admin
    -keypass welcome
    -keyalg RSA
    -dname "CN=jrfServer_admin,O=Oracle,C=US"
    -keystore default-keystore.jks
    -storepass password

cp default-keystore.jks ${domain.home}/config/fmwconfig

```

Make sure that the keystore service configured in the file `jps-config.xml` points to the generated `default-keystore.jks`; the following sample illustrates a keystore service configuration:

```

<!-- KeyStore Service Instance -->
<serviceInstance name="keystore"
    provider="keystore.provider" location="./default-keystore.jks">

```

```
<description>Default JPS Keystore Service</description>
<property name="keystore.provider.type" value="file"/>
<property name="keystore.file.path" value="."/ />
<property name="keystore.type" value="JKS"/>
<property name="keystore.csf.map" value="oracle.wsm.security"/>
<property name="keystore.pass.csf.key" value="keystore-csf-key"/>
<property name="keystore.sig.csf.key" value="sign-csf-key"/>
<property name="keystore.enc.csf.key" value="enc-csf-key"/>
</serviceInstance >
```

CSF Configuration

Create a map/key pair used to open the keystore and another map/key pair used to issue tokens. The following commands illustrate these operations using the OPSS script `createCred`:

```
// JKS keystore opening password
createCred(map="oracle.wsm.security", key="keystore-csf-key",
          user="keystore", password="password")

// Private key password to issue tokens
createCred(map="oracle.wsm.security", key="sign-csf-key",
          user="orakey", password="password")
```

For details about the OPSS script `createCred`, see [Section 10.5, "Managing Credentials with OPSS Scripts."](#)

Grant Configuration

Add a grant like the following to the policy store, which allows the client application to use the trust service API:

```
<grant>
  <grantee>
    <codesource>
      <url>file:${oracle.deployed.app.dir}/<MyApp>${oracle.deployed.app.ext}</url>
    </codesource>
  </grantee>
  <permissions>
    <permission>
<class>oracle.security.jps.service.trust.TrustServiceAccessPermission</class>
      <name>appId=*</name>
      <actions>issue</actions>
    </permission>
  </permissions>
</grant>
```

The Oracle WebLogic Server must be stopped and re-started for the above grant to take effect.

Servlet Code

The following sample illustrates how a servlet can obtain an asserted user name:

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    String username = request.getRemoteUser();
    ServletOutputStream out = response.getOutputStream();
    out.print("Asserted username: " + username);
    out.close();
}
```

web.xml Configuration

Set the appropriate login method in the file `web.xml`, as illustrated in the following snippet:

```
<web-app id="WebApp_ID"
...
  <login-config>
    <auth-method>CLIENT-CERT</auth-method>
    <realm-name>Identity Assertion</realm-name>
  </login-config>
...
</web-app>
```

WebLogic Asserter and Trust Service Configuration

To configure the WebLogic asserter, proceed as follows:

1. Copy the WebLogic identity asserter JAR `jps-wls-trustprovider.jar` to the location `${domain.home}/lib/mbeantypes`, as illustrated by the following command, and then restart the WebLogic Server:

```
cp ${common.components.home}/modules/oracle.jps_
11.1.1/jps-wls-trustprovider.jar ${domain.home}/lib/mbeantypes
```

2. Use WebLogic Console to configure the asserter, as follows:
 1. Login to the console as an administrator.
 2. Navigate to **Security Settings > Security Realms > myrealm > Providers Tab > Authentication**, and click **New** to open the **Create a New Authentication Provider** dialog.
 3. In that dialog, enter `TrustServiceIdentityAsserter` in the name box, and select `TrustServiceIdentityAsserter` from the pull-down in the type box; then click **OK**.
3. Verify that a grant like the following is present in the policy store; this grant is required for the asserter to use the OPSS trust service API; if necessary, use WSLT scripts to specify the grant:

```
<grant>
  <grantee>
    <codesource>

    <url>file:${domain.home}/lib/mbeantypes/jps-wls-trustprovider.jar</url>
    </codesource>
  </grantee>
  <permissions>
    <permission>

    <class>oracle.security.jps.service.trust.TrustServiceAccessPermission</class>
    <name>appId=*</name>
    <actions>validate</actions>
    </permission>
  </permissions>
</grant>
```

Any changes to the file `jps-config.xml`) requires the server to be re-started before updates take effect.

WebSphere Trust Asserter Interceptor Configuration

For details on this topic, see section *Configuring the Trust Association Interceptor* in *Oracle Fusion Middleware Third-Party Application Server Guide*.

19.3.1.2.2 Multiple Domain Scenario In this scenario there are two different domains: Domain1 and Domain2. The client application is running in Domain1; the servlet application is running in Domain2. It is assumed that each of these two domains have each a trust store service and keystore properly configured as explained under the heading WebLogic Asserter and Trust Store Configuration in the [Single Domain Scenario](#). In this scenario, the client application uses Domain1's trust service for token generation, and the servlet application uses Domain2's trust service for token validation.

In Domain1, the client sample code and the following configurations are identical to those described in the [Single Domain Scenario](#):

- the client application is illustrated by the code under the heading Client Application Code Sample.
- the configuration of the keystore is illustrated under the heading Keystore Service Configuration.
- the CSF configuration is illustrated under the heading CSF Configuration.
- the grant configuration is illustrated under the heading Grant Configuration.

In Domain 2, the servlet sample code and `web.xml` configuration are identical to those described in the [Single Domain Scenario](#), but there is some extra setup required:

- The servlet application code is illustrated by the code under the heading Servlet Code in the [Single Domain Scenario](#).
- The configuration of the file `web.xml` is illustrated under the heading `web.xml` Configuration in the [Single Domain Scenario](#).
- The client certificate that is used to sign the token in Domain1 must be present in Domain2's keystore; therefore, the administrator proceeds as follows:

1. Exports the certificate from Domain 1's keystore, as illustrated by the following command:

```
JAVA_HOME/bin/keytool -export
-alias jrfServer_admin.cer
-keystore default-keystore.jks
-storepass password
```

2. Imports the certificate into Domain 2's keystore as illustrated by the command below. Note that the alias passed must be the same as the alias used in step 1 for the export; if you overwrite the issuer name in the client side then that issuer name should be used as the alias.

```
JAVA_HOME/bin/keytool -importcert
-alias jrfServer_admin
-keypass welcome
-keyalg RSA
-dname "CN=jrfServer_admin,O=Oracle,C=US"
-keystore default-keystore.jks
-storepass password
```

3. Sets the Domain2's keystore password in the (Domain2's) credential store using the OPSS script `createCred` as follows:

```
createCred(map="oracle.wsm.security", key="keystore-csf-key",
user="keystore", password="password")
```

19.3.1.3 Domains Using Both Protocols

In this scenario, applications use either the HTTP protocol or the SOAP protocol, and not all applications in the domain use the same protocol. In such scenario, the keystore can be shared by the trust service used by the HTTP protocol and the SOAP service used by Oracle Web Services Manager. But in order for the trust service to work in this case, some special configurations in the file `jps-config.xml` are required as explained in the following sections:

- [Single Domain Scenario](#)
- [Multiple Domain Scenario](#)

19.3.1.3.1 Single Domain Scenario In this scenario, there is one keystore. The following snippet illustrates the configuration required for a certificate with alias `orakey`:

```
<propertySet name="trust.provider.embedded">
  <property name="trust.provider.className"
value="oracle.security.jps.internal.trust.provider.embedded.EmbeddedProviderImpl"/
>
  <property name="trust.clockSkew" value="60"/>
  <property name="trust.token.validityPeriod" value="1800"/>
  <property name="trust.token.includeCertificate" value="false"/>

  <!-- The alias used to get the signing certificate from JKS -->
  <property name="trust.aliasName" value="orakey"/>

  <!-- The issuer name to be added in the token used by the destination
trust service instance as an alias to pick up the corresponding certificate
to validate the token signature -->
  <property name="trust.issuerName" value="orakey"/>
</propertySet>
```

19.3.1.3.2 Multiple Domain Scenario In this scenario, there are two domains and two keystores. The following snippet illustrates the configuration required in the domain that is issuing tokens for a certificate with alias `orakey`:

```
<!-- issuer domain trust store must have a signing certif. w. alias orakey -->
<propertySet name="trust.provider.embedded">
  <property name="trust.provider.className"
value="oracle.security.jps.internal.trust.provider.embedded.EmbeddedProviderImpl"/
>
  <property name="trust.clockSkew" value="60"/>
  <property name="trust.token.validityPeriod" value="1800"/>
  <property name="trust.token.includeCertificate" value="false"/>

  <!-- the signing certificate alias in local JKS -->
  <property name="trust.aliasName" value="orakey"/>

  <!-- the token issuer's name -->
  <property name="trust.issuerName" value="domain1"/>
</propertySet>
```

The following snippet illustrates the configuration required in the domain that is receiving tokens for a certificate with alias `orakey`:

```
<!-- important: recipient domain must have a token validation certificate for
domain1,
which is the one was used to sign the token with alias "domain1" -->
<propertySet name="trust.provider.embedded">
```

```
<property name="trust.provider.className"
value="oracle.security.jps.internal.trust.provider.embedded.EmbeddedProviderImpl" /
>
  <property name="trust.clockSkew" value="60" />
  <property name="trust.token.validityPeriod" value="1800" />
  <property name="trust.token.includeCertificate" value="false" />

  <!-- the signing certificate alias in local JKS -->
  <property name="trust.aliasName" value="orakey" />

  <!-- the token issuer's name -->
  <property name="trust.issuerName" value="domain2" />
</propertySet>
```

19.3.2 A Custom Graphical User Interface

This use case illustrates some of the operations needed, for example, when implementing a custom graphic UI to manage policies. The samples presented use the OPSS APIs and demonstrate the following operations:

- Querying users in the identity store.
- Querying application roles in the policy store.
- Querying the mapping of users and groups to application roles; specifically, given a user identify all the application roles mapped to that user (Recall that the mapping of users and groups to application roles is a many-to-many relationship).
- Creating, reading, updating, and deleting the mapping of users and groups to application roles.

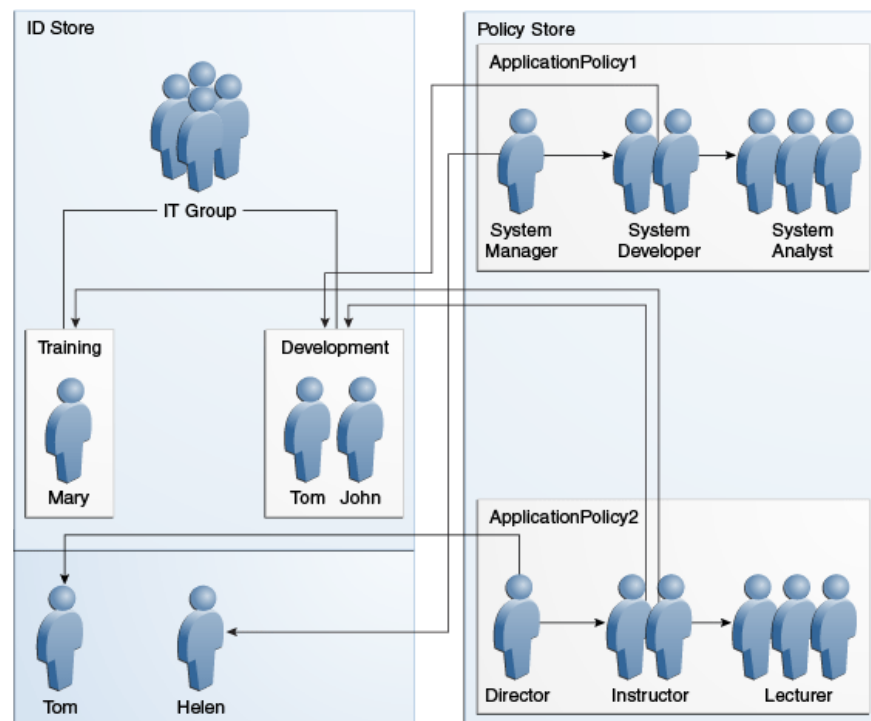
This use case assumes that:

- The identity store is an OID LDAP-based store.
- The policy store is an OID LDAP-based store.
- The identity store contains the following hierarchy of users and groups (enterprise roles):
 - The users Mary, John, Tom, and Helen.
 - The groups IT, Training, and Development.
 - The groups Training and Development are members of the group IT.
 - The user Mary is a member of the group Training.
 - The users Tom and John are members of the group Development.
- The policy store contains the following application policies and hierarchy of application roles:
 - The application policies ApplicationPolicy1 and ApplicationPolicy2.
 - The roles System Manager, System Developer, and System Analyst are application roles referenced in the policy ApplicationPolicy1; the System Manager role is a member of the System Developer role; the System Developer role is a member of the System Analyst role.
 - The roles Director, Instructor, and Lecturer are application roles referenced in the application policy ApplicationPolicy2; the Director role is a member of the Instructor role; the Instructor role is member of the Lecturer role.

- The mapping of application roles to users and groups is as follows:
 - The role System Manager is mapped to the user Helen.
 - The role System Developer is mapped to the group Development.
 - The role Director is mapped to the user Tom.
 - The role Instructor is mapped to the groups Training and Development.

Figure 19–3 illustrates the hierarchy of application roles, the users and groups, and the mapping of application roles to users and groups, as assumed in this use case.

Figure 19–3 Mapping of Application Roles to Users and Groups



Note that the above role hierarchy implies, for instance, that a user in the System Manager role is also in the System Developer role, and similarly with the other roles. Therefore the role membership for each of the four users is as follows:

- User Tom is a member of the following application roles: System Developer, System Analyst, Director, Instructor, and Lecturer.
- User Helen is a member of the following application roles: System Manager, System Developer, and System Analyst.
- User Mary is a member of the following application roles: Instructor and Lecturer.
- User John is a member of the following application roles: System Developer, System Analyst, Instructor, and Lecturer.

The code samples are detailed in the following sections:

- [Imports Assumed](#) - List of imports
- [Code Sample 1](#) - Querying the identity store.
- [Code Sample 2](#) - Creating application roles and assigning members to a role.

- [Code Sample 3](#) - Querying application roles.
- [Code Sample 4](#) - Mapping application roles to users and groups.
- [Code Sample 5](#) - Getting all the roles that have a given user as a member.
- [Code Sample 6](#) - Removing the mapping of an application role to a group.

19.3.2.1 Imports Assumed

The sample codes in this use case assume the following import statements:

```
import java.security.AccessController;
import java.security.Policy;
import java.security.Principal;
import java.security.PrivilegedExceptionAction;
import java.security.Security;
import java.util.HashSet;
import java.util.List;
import java.util.Set;
import javax.security.auth.Subject;
import oracle.security.idm.Identity;
import oracle.security.idm.IdentityStore;
import oracle.security.idm.ObjectNotFoundException;
import oracle.security.idm.Role;
import oracle.security.idm.RoleManager;
import oracle.security.idm.SearchParameters;
import oracle.security.idm.SearchResponse;
import oracle.security.idm.SimpleSearchFilter;
import oracle.security.idm.User;
import oracle.security.idm.UserProfile;
import oracle.security.jps.ContextFactory;
import oracle.security.jps.JpsContext;
import oracle.security.jps.JpsContextFactory;
import oracle.security.jps.principals.JpsApplicationRole;
import oracle.security.jps.service.idstore.IdentityStoreService;
import oracle.security.jps.service.policystore.ApplicationPolicy;
import oracle.security.jps.service.policystore.PolicyObjectNotFoundException;
import oracle.security.jps.service.policystore.PolicyStore;
import oracle.security.jps.service.policystore.PolicyStoreException;
import oracle.security.jps.service.policystore.entitymanager.AppRoleManager;
import oracle.security.jps.service.policystore.info.AppRoleEntry;
import oracle.security.jps.service.policystore.search.AppRoleSearchQuery;
import oracle.security.jps.service.policystore.search.ComparatorType;
import oracle.security.jps.util.JpsAuth;
import weblogic.security.principal.PrincipalFactory;
```

19.3.2.2 Code Sample 1

The following sample code illustrates two queries to users in the identity store:

```
private void queryUsers() throws Exception {
    // Using IDM U/R to query ID store
    IdentityStore idmStore = idStore.getIdmStore();

    // Query an individual user by name
    User employee = idmStore.searchUser(USER_TOM);
    log("-----");
    log("### Query individual user (Tom) from ID store ###");
    log(USER_TOM + ": " + employee.getName() + " GUID: " +
        employee.getGUID());
    log();
}
```

```

// Get all users whose name is not "Paul"
SimpleSearchFilter filter =
    idmStore.getSimpleSearchFilter(UserProfile.NAME,
                                   SimpleSearchFilter.TYPE_NOTEQUAL,
                                   "Paul");

SearchParameters sps =
    new SearchParameters(filter, SearchParameters.SEARCH_USERS_ONLY);
SearchResponse result = idmStore.searchUsers(sps);
log("-----");
log("### Query all users (whose name is not Paul) from ID store ###");
log("Found the following users:");
while (result.hasNext()) {
    Identity user = result.next();
    log("\t user: " + user.getName() + ", GUID: " + user.getGUID());
}
log();
}

```

19.3.2.3 Code Sample 2

The following sample code illustrates how to create an application role and how to make a role a member of another role:

```

private void createAppRoles1() throws Exception {
    AppRoleManager arm1 = apl.getAppRoleManager();
    log("-----");
    log("### Creating app roles in app policy1 with hierachy ###");

    AppRoleEntry sysAnalystRole =
        arm1.createAppRole(APP_ROLE_SYS_ANALYST, APP_ROLE_SYS_ANALYST,
                          APP_ROLE_SYS_ANALYST);
    AppRoleEntry sysDeveloperRole =
        arm1.createAppRole(APP_ROLE_SYS_DEVELOPER, APP_ROLE_SYS_DEVELOPER,
                          APP_ROLE_SYS_DEVELOPER);
    AppRoleEntry sysManagerRole =
        arm1.createAppRole(APP_ROLE_SYS_MANAGER, APP_ROLE_SYS_MANAGER,
                          APP_ROLE_SYS_MANAGER);

    apl.addPrincipalToAppRole(sysManagerRole, APP_ROLE_SYS_DEVELOPER);
    apl.addPrincipalToAppRole(sysDeveloperRole, APP_ROLE_SYS_ANALYST);
    log("### App roles in app policy #1 have been created ###");
    log();
}

```

19.3.2.4 Code Sample 3

The following code sample illustrates several ways to query application roles:

```

private void queryAppRolesInApplicationPolicy1() throws Exception {
    AppRoleManager arm1 = apl.getAppRoleManager();

    // Get role that matches a name
    AppRoleEntry are = arm1.getAppRole(APP_ROLE_SYS_MANAGER);
    log("-----");
    log("### Query app roles in application policy #1, by name ###");
    log("Found " + are.getName() + " by app role name.");
    log();

    // Get the role that matches a name exactly
}

```

```

AppRoleSearchQuery q =
    new AppRoleSearchQuery(AppRoleSearchQuery.SEARCH_PROPERTY.NAME,
        false, ComparatorType.EQUALITY,
        APP_ROLE_SYS_ANALYST,
        AppRoleSearchQuery.MATCHER.EXACT);
List<AppRoleEntry> arel = arm1.getAppRoles(q);
log("### Query app roles in application policy #1, by exact query ###");
log("Found " + arel.get(0).getName() + " by exact query.");
log();

// Get roles with names that begin with a given string
q =
new AppRoleSearchQuery(AppRoleSearchQuery.SEARCH_PROPERTY.NAME, false,
    ComparatorType.EQUALITY,
    APP_ROLE_SYS_DEVELOPER.subSequence(0, 7),
    AppRoleSearchQuery.MATCHER.BEGINS_WITH);
arel = arm1.getAppRoles(q);
log("### Query app roles in app policy #1, by begins_with query ###");
log("Found " + arel.get(0).getName() + " by begins_with query.");
log();

// Get roles with names that contain a given substring
q =
new AppRoleSearchQuery(AppRoleSearchQuery.SEARCH_PROPERTY.NAME, false,
    ComparatorType.EQUALITY, "dummy",
    AppRoleSearchQuery.MATCHER.ANY);
arel = arm1.getAppRoles(q);
log("### Query app roles in app policy #1, by matcher any ###");
log("Found " + arel.size() + " app roles by matcher any.");
for (AppRoleEntry ar : arel) {
    log("\t" + ar.getName());
}
log();
}

```

19.3.2.5 Code Sample 4

The following sample illustrates how to map application roles to users and groups:

```

private void assignAppRoleToUsersAndGroups() throws Exception {
    // Obtain the user/group principals
    IdentityStore idmStore = idStore.getIdmStore();
    User tom = idmStore.searchUser(USER_TOM);
    User helen = idmStore.searchUser(USER_HELEN);

    Role trainingRole =
        idmStore.searchRole(IdentityStore.SEARCH_BY_NAME, GROUP_TRAINING);
    Role devRole =
        idmStore.searchRole(IdentityStore.SEARCH_BY_NAME, GROUP_DEV);

    Principal tomPrincipal =
        PrincipalFactory.getInstance().createWLSUser(tom.getName(),
            tom.getGUID(),
            tom.getUniqueName());

    Principal helenPrincipal =
        PrincipalFactory.getInstance().createWLSUser(helen.getName(),
            helen.getGUID(),
            helen.getUniqueName());

    Principal trainingPrincipal =
        PrincipalFactory.getInstance().createWLSGroup(trainingRole.getName(),

```

```

trainingRole.getGUID(),

trainingRole.getUniqueName());
    Principal devPrincipal =
        PrincipalFactory.getInstance().createWLSGroup(devRole.getName(),
                                                    devRole.getGUID(),

devRole.getUniqueName());

    // Application policy #1
    log("-----");
    log("### Assigning appl roles to users and groups, app policy #1 ###");
    ap1.addPrincipalToAppRole(helenPrincipal, APP_ROLE_SYS_MANAGER);
    ap1.addPrincipalToAppRole(devPrincipal, APP_ROLE_SYS_DEVELOPER);

    // Application policy #2
    log("### Assigning app roles to users and groups, app policy #2 ###");
    ap2.addPrincipalToAppRole(tomPrincipal, APP_ROLE_DIRECTOR);
    ap2.addPrincipalToAppRole(devPrincipal, APP_ROLE_INSTRUCTOR);
    ap2.addPrincipalToAppRole(trainingPrincipal, APP_ROLE_INSTRUCTOR);

    log("### App roles have been assigned to users and groups ###");
    log();
}

```

19.3.2.6 Code Sample 5

The following code sample illustrates how to get all the roles that have a given user as a member:

```

private void showAppRoles() throws Exception {
    Subject tomSubject = getUserSubject(USER_TOM);
    Subject helenSubject = getUserSubject(USER_HELEN);
    Subject johnSubject = getUserSubject(USER_JOHN);
    Subject marySubject = getUserSubject(USER_MARY);

    Set<String> applications = new HashSet<String>();
    applications.add(APPLICATION_NAME1);
    applications.add(APPLICATION_NAME2);

    log("-----");
    log("### Query application roles for Tom ###");
    showAppRoles(applications, USER_TOM, tomSubject);
    log();

    log("### Query application roles for Helen ###");
    showAppRoles(applications, USER_HELEN, helenSubject);
    log();

    log("### Query application roles for John ###");
    showAppRoles(applications, USER_JOHN, johnSubject);
    log();

    log("### Query application roles for Mary ###");
    showAppRoles(applications, USER_MARY, marySubject);
    log();
}

private Subject getUserSubject(String userName) throws Exception {
    Subject subject = new Subject();

```

```

// Query users from ID store using user/role API,for user principal
IdentityStore idmStore = idStore.getIdmStore();
User user = idmStore.searchUser(userName);

Principal userPrincipal =
    PrincipalFactory.getInstance().createWLSUser(user.getName(),
                                                user.getGUID(),
                                                user.getUniqueName());

subject.getPrincipals().add(userPrincipal);

// Query users from ID store using user/role API, for enterprise roles
RoleManager rm = idmStore.getRoleManager();
SearchResponse result = null;
try {
    result = rm.getGrantedRoles(user.getPrincipal(), false);
} catch (ObjectNotFoundException onfe) {
    // ignore
}

// Add group principals to the subject
while (result != null && result.hasNext()) {
    Identity role = result.next();
    Principal groupPrincipal =
        PrincipalFactory.getInstance().createWLSGroup(role.getName(),
                                                    role.getGUID(),
                                                    role.getUniqueName());
    subject.getPrincipals().add(groupPrincipal);
}

// The subject now contains both user and group principals.
// In the WebLogic Server, this setting is done by a login module
return subject;
}

private void showAppRoles(Set<String> applications, String user, Subject subject)
{
    // Get all granted application roles for this subject
    Set<JpsApplicationRole> result = null;
    try {
        result = JpsAuth.getAllGrantedAppRoles(subject, applications);
    } catch (PolicyStoreException pse) {
        log(pse.toString());
    }

    if (result.size() <= 1) {
        log(user + " has " + result.size() + " application role.");
        if (result.size() == 1) {
            for (JpsApplicationRole ar : result) {
                log("\tApplication role: " + ar.getName());
            }
        }
    } else {
        System.out.println(user + " has " + result.size() +
            " application roles.");
        for (JpsApplicationRole ar : result) {
            log("\tApplication role: " + ar.getAppID() + "/" +
                ar.getName());
        }
    }
}

```

```

}
```

19.3.2.7 Code Sample 6

The following sample code illustrates how to remove the mapping of an application role to a group:

```

private void removeAppRoleForUserDirector() throws Exception {
    // Remove instructor role from Dev group
    log("-----");
    log("### Removing Instructor application role from Dev group ###");

    IdentityStore idmStore = idStore.getIdmStore();
    Role devRole =
        idmStore.searchRole(IdentityStore.SEARCH_BY_NAME, GROUP_DEV);
    Principal devPrincipal =
        PrincipalFactory.getInstance().createWLSGroup(devRole.getName(),
                                                    devRole.getGUID(),

devRole.getUniqueName());

    ap2.removePrincipalFromAppRole(devPrincipal, APP_ROLE_INSTRUCTOR);
    log("### Instructor app role has been removed from Dev group ###");
    log();

    log("-----");
    log("### Now query application roles for user John, again ###");
    Set<String> applications = new HashSet<String>();
    applications.add(APPLICATION_NAME1);
    applications.add(APPLICATION_NAME2);

    Subject johnSubject = getUserSubject(USER_JOHN);
    showAppRoles(applications, USER_JOHN, johnSubject);
    log();
}

```

19.4 Appendix - Security Life Cycle of an ADF Application

This section explains the phases that the security of an application goes through. It is assumed that the application uses ADF and that it is developed in the Oracle JDeveloper environment.

The phases of the security life cycle of an application are the development phase, the deployment phase, and the management phase. The participants are the product manager or application architect, application developers, and application security administrators. For a summary of tasks, see [Summary of Tasks per Participant per Phase](#).

19.4.1 Development Phase

In the development phase developers design the application to work with the full range of security options available in Oracle Fusion Middleware. Developers have access to a rich set of security services exposed by Oracle JDeveloper, the built-in ADF framework, and the Oracle WebLogic Server. All these components are based on OPSS, which ensures a consistent approach to security throughout the application's life span.

Typically, a developer uses the ADF Security Wizard (an authorization editor) and an expression language editor, all within Oracle JDeveloper; additionally and optionally,

he may use OPSS APIs to implement more complex security tasks. Thus, some parts of the application use declarative security, others use programmatic security, and they both rely on security features available in the development and run-time environment.

Application developers also define a number of application entitlements and roles (policy seed data) required to secure the application. This policy seed data is kept in a source control system together with the application source code.

19.4.2 Deployment Phase

Once developed, the application is typically tested in a staging environment before being deployed to a production environment. In a production environment, both the application and the run-time services are integrated with other security components, such as user directories, single sign-on systems, user provisioning systems, and auditing. The security services usually change with the phase: for example, during development, a developer relies on a file or Oracle Wallet to store user credentials, but, in a production environment, credentials are stored in an LDAP directory (the OPSS security store).

In the deployment phase, typically, an administrator migrates the policy seed data to the production policy store (the OPSS security store), and maps application roles to enterprise groups to effect application security policies.

19.4.3 Management Phase

The management phase starts once an application has been deployed to a production environment. In this phase, application administrators or enterprise security administrators manage day-to-day security tasks, such as granting users access to application resources, reviewing audit logs, responding to security incidents, and applying security patches.

19.4.4 Summary of Tasks per Participant per Phase

The following tables summarize the major responsibilities per participant in each of the security life cycle phases and [Figure 19-4](#) illustrates the basic flow.

Figure 19-4 Application Life Cycle Phases

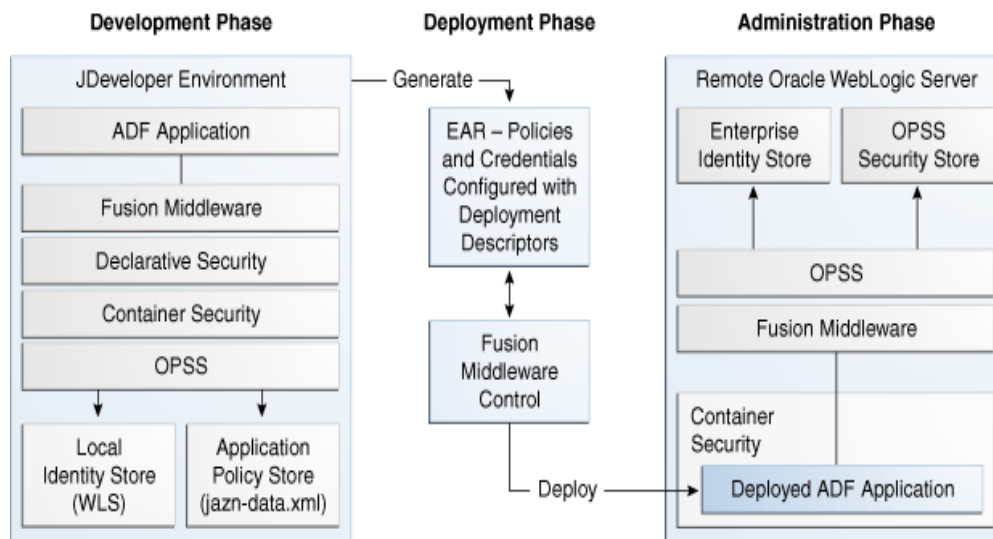


Table 19–1 Security Tasks for the Application Architect

Phase	Task
Development	<p>Defines high-level application roles based on functional security and data security requirements.</p> <p>Populates the initial file-based application policy store (<code>jazn-data.xml</code>).</p>
Deployment	Defines real-world customer scenarios to be tested by the QA team.
Management	<p>Understands and identifies the requirements to customize application policies.</p> <p>Considers defining templates for vertical industries.</p>

Table 19–2 Security Tasks for the Application Developer

Phase	Task
Development	<p>Uses tools and processes, specifically Oracle JDeveloper, to build the application and to create security artifacts, such as application roles and permissions.</p> <p>Uses FND Grants to specify data-level security.</p> <p>Tests the application using a local policy store with sample users and roles.</p>
Deployment	Assists the QA team to troubleshoot and resolve runtime issues.

Table 19–3 Security Tasks for the Application Security Administrator

Phase	Task
Deployment	<p>Uses deployment services to migrate security seed data in <code>jazn-data.xml</code> to the production policy store.</p> <p>Maps application roles to enterprise groups so that security policies can be enforced.</p>
Management	<p>Applies patches and upgrades software, as necessary.</p> <p>Manages users and roles, as enterprise users and the application role hierarchy changes overtime.</p> <p>Manages policies packed with the application and creates new ones.</p> <p>Integrates with and manages the IAM infrastructure.</p>

19.5 Appendix - Code and Configuration Examples

This section lists most of the code and configuration samples found elsewhere in this Guide, and a fully-written code example.

- [Code Examples](#)
- [Configuration Examples](#)
- [Full Code Example of a Java EE Application with Integrated Security](#)

19.5.1 Code Examples

The following list includes typical security-related programming tasks and links to sample code illustrating implementations:

- [Querying an LDAP identity store](#) - See [Section 7.4, "Querying the Identity Store Programmatically."](#)

- Querying application roles and the mapping of users and groups to application roles - See [A Custom Graphical User Interface](#).
- Invoking the method `isUserInRole` - See [Section 20.2.2.2, "Programmatic Authorization."](#)
- Managing policies - See [Section 20.3.2, "Managing Policies."](#)
- Checking policies - See [Section 20.3.3, "Checking Policies."](#)
- Using the class `ResourcePermission` - See [Section 20.3.4, "The Class ResourcePermission."](#)
- Using the Identity Store Login Module for authentication in Java SE applications - See [Section 22.2.3.2, "Using the Identity Store Login Module for Authentication."](#)
- Using the Identity Store Login Module for assertion in Java SE applications - See [Section 22.2.3.3, "Using the Identity Login Module for Assertion."](#)

19.5.2 Configuration Examples

The following list includes typical security-related configuration tasks and links to sample configuration:

- Configuring an OAM SSO provider - See [Section 8.7.3.3, "OAM Configuration Example."](#)
- Configuring resource permissions - See [Section 20.3.4, "The Class ResourcePermission."](#)
- Configuring the servlet filter and the EJB interceptor - See [Section 21.1, "Configuring the Servlet Filter and the EJB Interceptor."](#)
- Configurations involved with `migrateSecurityStore` - See [Section 6.5.2.1, "Migrating Policies Manually,"](#) and [Section 6.5.2.2, "Migrating Credentials Manually."](#)
- Configuring an LDAP identity store - See [Section 7.3.2.6, "Examples of the Configuration File,"](#) and [Section 22.2.2, "Configuring an LDAP Identity Store in Java SE Applications."](#)
- Configuring the policy and credential stores in Java SE applications - See [Section 23.1, "Configuring Policy and Credential Stores in Java SE Applications."](#)

19.5.3 Full Code Example of a Java EE Application with Integrated Security

`ezshare` is a full example of a Java EE application whose security has been integrated with OPSS that uses permission-based grants and available at the Oracle Network. To locate the example, search for the keyword `ezshare`.

The OPSS Policy Model

This chapter explains the OPSS policy and authorization models in the following sections:

- [The Security Policy Model](#)
- [Authorization Overview](#)
- [The JAAS/OPSS Authorization Model](#)

20.1 The Security Policy Model

For details about the OPSS policy model and the security artifacts used in it, see *Oracle Fusion Middleware Administrator's Guide for Oracle Entitlements Server*.

20.2 Authorization Overview

This section compares and contrasts the authorization available in the Java EE and the JAAS models, in the following sections:

- [Introduction to Authorization](#)
- [The Java EE Authorization Model](#)
- [The JAAS Authorization Model](#)

20.2.1 Introduction to Authorization

A Java 2 policy specifies the permissions granted to signed code loaded from a given location. A JAAS policy extends Java 2 grants by allowing an optional list of principals; permissions are granted only to code from a given location, possibly signed, and run by a user represented by those principals.

The Policy Store is a repository of system and application-specific policies and roles. Application roles can be granted (mapped) to enterprise users and groups specific to the application (such as administrative roles). A policy can grant permissions to any of these roles, groups, or users as principals.

For more details about policy-related security artifacts, see [Chapter 3.2, "Policy Store Basics."](#)

An application can delegate the enforcement of authorization to the container, or it can implement its own enforcement of policy checking with calls to methods such as `checkPermission`, `checkBulkAuthorization`, or `getGrantedResources`.

For details about policy checking with API calls, see [Checking Policies](#).

20.2.2 The Java EE Authorization Model

The Java EE authorization model uses role membership to control access to EJB methods and web resources that are referenced by URLs; policies assign permissions to users and roles, and they are enforced by the container to protect resources.

In the Java EE model, authorization is implemented in either of the following ways:

- Declaratively, where authorization policies are specified in deployment descriptors; the container reads those policies from deployment descriptors and enforces them. No special application code is required to enforce authorization.
- Programmatically, where authorization policies are checked in application code; the code checks whether a subject has the appropriate permission to execute specific sections of code. If the subject fails to have the proper permission, the code throws an exception.

Table 20–1 shows the advantages and disadvantages of each approach.

Table 20–1 Comparing Authorization in the Java EE Model

Authorization Type	Advantages	Disadvantages
Declarative	No coding needed; easy to update by modifying just deployment descriptors.	Authorization is coarse-grained and specified at the URL level or at the method level (for EJBs).
Programmatic	Specified in application code; can protect code at a finer levels of granularity.	Not so easy to update, since it involves code changes and recompilation.

A container can provide authorization to applications running in it in two ways: declaratively and programmatically; these topics and an example are explained in the following sections:

- [Declarative Authorization](#)
- [Programmatic Authorization](#)
- [Java EE Code Example](#)

20.2.2.1 Declarative Authorization

Declarative authorization allows to control access to URL-based resources (such as servlets and pages) and methods in EJBs.

The basic steps to configure declarative authorization are the following:

1. In standard deployment descriptors, specify the resource to protect, such as a web URL or an EJB method, and a logical role that has access to the resource.
Alternatively, since Java EE 1.5 supports annotations, use code annotations instead of deployment descriptors.
2. In proprietary deployment descriptors (such as `web.xml`), map the logical role defined in step 1 to an enterprise group.

For details, see the chapter *Using Security Services in Oracle Fusion Middleware Enterprise JavaBeans Developer's Guide for Oracle Containers for Java EE*.

20.2.2.2 Programmatic Authorization

Programmatic authorization provides a finer grained authorization than the declarative approach, and it requires that the application code invoke the method

`isUserInRole` (for servlets and JSPs) or the method `isCallerInRole` (for EJBs), both available from standard Java APIs.

Although these methods still depend on role membership to determine authorization, they give finer control over authorization decisions since the controlling access is not limited at the resource level (EJB method or URL).

20.2.2.3 Java EE Code Example

The following example illustrates a servlet calling the method `isUserInRole`. It is assumed that the EAR file packing the servlet includes the configuration files `web.xml` and `weblogic-application.xml`, and that these files include the following configuration fragments:

web.xml

```
<!-- security roles -->
<security-role>
  <role-name>sr_developer</role-name>
</security-role>
```

weblogic-application.xml

The following snippet shows the mapping between the user `weblogic` and the security role `sr_developer`:

```
<wls:security-role-assignment>
  <wls:role-name>sr_developer</wls:role-name>
  <wls:principal-name>weblogic</wls:principal-name>
</wls:security-role-assignment>
```

Code Example Invoking isUserInRole

```
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletOutputStream;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.util.Date;

public class PolicyServlet extends HttpServlet {

    public PolicyServlet() {
        super();
    }

    public void init(ServletConfig config)
        throws ServletException {
        super.init(config);
    }

    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        final ServletOutputStream out = response.getOutputStream();

        response.setContentType("text/html");
        out.println("<HTML><BODY bgcolor=\\"#FFFFFF\\">");
        out.println("Time stamp: " + new Date().toString());
        out.println(" <br>request.getRemoteUser = " + request.getRemoteUser() +
"<br>");
```

```
        out.println("request.isUserInRole('sr_developer') = " +
request.isUserInRole("sr_developer") + "<br>");
        out.println("request.getUserPrincipal = " + request.getUserPrincipal() +
"<br>");
        out.println("</BODY>");
        out.println("</HTML>");
    }
}
```

20.2.3 The JAAS Authorization Model

The JAAS authorization introduces permissions but can still use the notion of roles. An authorization policy binds permissions with a Subject (role, group, or user) and, optionally, with source code. Granting to a role is achieved through calls to `addPrincipalsToAppRole`.

Permissions are evaluated by calls to the `AccessController`, and the model allows fine-grained control to resources.

In this model, an authorization policy specifies the following information:

- Application roles and enterprise groups.
- Permissions granted to users, groups, and code sources. For users and groups, they determine what a user or the member of a group is allowed to access. For code sources, they determine what actions the code is allowed to perform.

When programming with this model, sensitive lines of code are preceded with calls to check whether the current user or role is granted the appropriate permissions to access the code. If the user has the appropriate permissions, the code is run. Otherwise, the code throws an exception.

For details about JAAS standard permissions, see <http://java.sun.com/JavaSE/6/docs/technotes/guides/security/permissions.html>.

20.3 The JAAS/OPSS Authorization Model

JAAS/OPSS authorization is based on controlling the operations that a class can perform when it is loaded and run in the environment.

This section is divided into the following sections:

- [The Resource Catalog](#)
- [Managing Policies](#)
- [Checking Policies](#)
- [The Class ResourcePermission](#)

20.3.1 The Resource Catalog

OPSS supports the specification and runtime support of the resource catalog in file-, LDAP-, and DB-based policy stores.

Using the resource catalog provides the following benefits:

- Describes policies and secured artifacts in human-readable terms.
- Allows defining and modifying policies independently of and without knowledge of the application source code.
- Allows browsing and searching secured artifacts.

- Allows grouping of secured artifacts in building blocks (entitlements or permission sets) which can be later used in authorization policies.

20.3.2 Managing Policies

Resource catalog artifacts can be managed with the policy management API. Specifically, the following interfaces, all subinterfaces of the interface `oracle.security.jps.service.policystore.EntityManager`, are directly relevant to the artifacts in the resource catalog:

- `GrantManager` - This interface includes methods to query grants using search criteria, to obtain list of grants that satisfy various combinations of resource catalog artifacts, and to grant or revoke permissions to principals.
- `PermissionSetManager` - This interface includes methods to create, modify, and query permission sets (entitlements).
- `ResourceManager` - This interface includes methods to create, delete, and modify resource (instances).
- `ResourceTypeManager` - This interface includes methods to create, delete, modify, and query resource types.

For details about these interfaces, see the Javadoc document *Oracle Fusion Middleware Java API Reference for Oracle Platform Security Services*.

The following code snippet illustrates the creation of a resource type, a resource instance, actions, and a permission set:

```
import oracle.security.jps.service.policystore.entitymanager.*;
import oracle.security.jps.service.policystore.search.*;
import oracle.security.jps.service.policystore.info.resource.*;
import oracle.security.jps.service.policystore.info.*;
import oracle.security.jps.service.policystore.*;
import java.util.*;

public class example {
    public static void main(String[] args) throws Exception {
        ApplicationPolicy ap;

        ResourceTypeManager rtm = ap.getEntityManager(ResourceTypeManager.class);
        ResourceTypeSearchQuery query = new ResourceTypeSearchQuery();
        query.setANDMatch();
        query.addQuery(ResourceTypeSearchQuery.SEARCH_PROPERTY.NAME, false,
            ComparatorType.EQUALITY, "resourceType", BaseSearchQuery.MATCHER.EXACT);
        List<ResourceTypeEntry> allResourceTypes = rtm.getResourceTypes(query);

        ResourceManager rm = ap.getEntityManager(ResourceManager.class);
        ResourceSearchQuery ResourceQuery = new ResourceSearchQuery();
        ResourceQuery.setANDMatch();
        ResourceQuery.addQuery(ResourceSearchQuery.SEARCH_PROPERTY.NAME, false,
            ComparatorType.EQUALITY, "R2", BaseSearchQuery.MATCHER.EXACT);
        List<ResourceEntry> allResources = rm.getResources("RT2", ResourceQuery);

        PermissionSetManager psm = ap.getEntityManager(PermissionSetManager.class);
        PermissionSetSearchQuery pssq = new PermissionSetSearchQuery();
        pssq.setANDMatch();
        pssq.addQuery(PermissionSetSearchQuery.SEARCH_PROPERTY.NAME, false,
            ComparatorType.EQUALITY, "PS1", BaseSearchQuery.MATCHER.EXACT);
        List<PermissionSetEntry> allPermSets = psm.getPermissionSets(pssq);
    }
}
```

```
RoleCategoryManager rcm = ap.getEntityManager(RoleCategoryManager.class);
RoleCategorySearchQuery rcsq = new RoleCategorySearchQuery();
rcsq.setANDMatch();
rcsq.addQuery(RoleCategorySearchQuery.SEARCH_PROPERTY.NAME, false,
ComparatorType.EQUALITY, "roleCategoryCartoon",
BaseSearchQuery.MATCHER.EXACT);

List<RoleCategoryEntry> allRoleCategories = rcm.getRoleCategories(rcsq);
}
}
```

The following code snippet illustrates a complex query involving resource catalog elements:

```
//ApplicationPolicy ap as in the preceeding example
ResourceTypeManager rtm = ap.getEntityManager(ResourceTypeManager.class);
ResourceTypeSearchQuery query = new ResourceTypeSearchQuery();
query.setANDMatch();
query.addQuery(ResourceTypeSearchQuery.SEARCH_PROPERTY.NAME, false,
ComparatorType.EQUALITY, "resourceType", BaseSearchQuery.MATCHER.EXACT);
List<ResourceTypeEntry> enties = rtm.getResourceTypes(query);

ResourceManager rm = ap.getEntityManager(ResourceManager.class);
ResourceSearchQuery ResourceQuery = new ResourceSearchQuery();
ResourceQuery.setANDMatch();
ResourceQuery.addQuery(ResourceSearchQuery.SEARCH_PROPERTY.NAME, false,
ComparatorType.EQUALITY, "R2", BaseSearchQuery.MATCHER.EXACT);
ArrayList<BaseSearchQuery> querries = ResourceQuery.getQueries();
List<ResourceEntry> resources = rm.getResources("RT2", ResourceQuery);

PermissionSetManager psm = ap.getEntityManager(PermissionSetManager.class);
PermissionSetSearchQuery pssq = new PermissionSetSearchQuery();
pssq.setANDMatch();
pssq.addQuery(PermissionSetSearchQuery.SEARCH_PROPERTY.NAME, false,
ComparatorType.EQUALITY, "PS1", BaseSearchQuery.MATCHER.EXACT);
List<PermissionSetEntry> psets = psm.getPermissionSets(pssq);

RoleCategoryManager rcm = ap.getEntityManager(RoleCategoryManager.class);
RoleCategorySearchQuery rcsq = new RoleCategorySearchQuery();
rcsq.setANDMatch();
rcsq.addQuery(RoleCategorySearchQuery.SEARCH_PROPERTY.NAME, false,
ComparatorType.EQUALITY, "roleCategoryCartoon", BaseSearchQuery.MATCHER.EXACT);
ArrayList<BaseSearchQuery> queries = rcsq.getQueries();
List<RoleCategoryEntry> rcs = rcm.getRoleCategories(rcsq);
```

The following code sample illustrates how to create a grant:

```
GrantManager gm = ap.getEntityManager(GrantManager.class);
Set<PrincipalEntry> pe = new HashSet<PrincipalEntry>();
List<AppRoleEntry> are = ap.searchAppRoles(appRoleName);
pe.addAll(are);
gm.grant(pe, null, permissionSetName);
```

20.3.3 Checking Policies

This section illustrates several ways to check policies programmatically, in the following sections:

- [Using the Method checkPermission](#)
- [Using the Methods doAs and doAsPrivileged](#)

- [Using the Method checkBulkAuthorization](#)
- [Using the Method getGrantedResources](#)

Important Note 1: Authorization failures are not visible, by default, in the console. To have authorization failures sent to the console you must set the system variable `jps.auth.debug` as follows:

```
-Djps.auth.debug=true
```

In particular, to have `JpsAuth.checkPermission` failures sent to the console, you must set the variable as above.

Important Note 2: The OPSS policy provider *must* be explicitly set in Java SE applications, as illustrated in the following snippet:

```
java.security.Policy.setPolicy(new
oracle.security.jps.internal.policystore.JavaPolicyProvider())
```

Not setting the policy provider explicitly in a Java SE application may cause runtime methods (such as `JpsAuth.checkPermission`) to return incorrect values.

20.3.3.1 Using the Method checkPermission

Oracle Fusion Middleware supports the use of the method `checkPermission` in the classes `java.security.AccessController` and `oracle.security.jps.util.JpsAuth`.

Oracle recommends the use of `checkPermission` in the class `JpsAuth` because it provides better debugging support, better performance, and audit support.

The static method `AccessController.checkPermission` uses the default access control context (the context inherited when the thread was created). To check permissions on some other context, call the instance method `checkPermission` on a particular `AccessControlContext` instance.

The method `checkPermission` behaves according to the value of the JAAS mode (see JAAS mode in [Chapter 21.1, "Configuring the Servlet Filter and the EJB Interceptor"](#)), as listed in the following table:

Table 20–2 Behavior of checkPermission According to JAAS Mode

JAAS Mode Setting	checkPermission
off or undefined	Enforces codebase security based on the security policy in effect, and there is no provision for subject-based security.
doAs	Enforces a combination of codebase and subject-based security using the access control context created through the <code>doAs</code> block.
doAsPrivileged	Enforces subject-based security using a null access control context.
subjectOnly	Takes into consideration grants involving principals <i>only</i> (and it disregards those involving codebase) when evaluating a permission.

Note: If `checkPermission` is called inside a `doAs` block and the check permission call fails, to display the failed protection domain you must set the system property `java.security.debug=access, failure`.

The following example illustrates a servlet checking a permission. It is assumed that the EAR file packing the servlet includes the configuration files `jazn-data.xml` and `web.xml`.

jazn-data.xml

The application file-based policy store is as follows:

```
<?xml version="1.0" ?>
<jazn-data>
  <policy-store>
    <applications>
      <application>
        <name>MyApp</name>

        <app-roles>
          <app-role>
            <name>AppRole</name>
            <display-name>AppRole display name</display-name>
            <description>AppRole description</description>
            <guid>F5494E409CFB11DEBFEB11296284F58</guid>
            <class>oracle.security.jps.service.policystore.ApplicationRole</class>
          </app-role>
        </app-roles>

        <role-categories>
          <role-category>
            <name>MyAppRoleCategory</name>
            <display-name>MyAppRoleCategory display name</display-name>
            <description>MyAppRoleCategory description</description>
          </role-category>
        </role-categories>

        <resource-types>
          <resource-type>
            <name>MyResourceType</name>
            <display-name>MyResourceType display name</display-name>
            <description>MyResourceType description</description>
            <provider-name>MyResourceType provider</provider-name>
            <matcher-class>oracle.security.jps.ResourcePermission</matcher-class>
            <actions-delimiter>,</actions-delimiter>
            <actions>write,read</actions>
          </resource-type>
        </resource-types>

        <resources>
          <resource>
            <name>MyResource</name>
            <display-name>MyResource display name</display-name>
            <description>MyResource description</description>
            <type-name-ref>MyResourceType</type-name-ref>
          </resource>
        </resources>
      </application>
    </applications>
  </policy-store>
</jazn-data>
```

```

<permission-sets>
  <permission-set>
    <name>MyEntitlement</name>
    <display-name>MyEntitlement display name</display-name>
    <description>MyEntitlement description</description>
    <member-resources>
      <member-resource>
        <type-name-ref>MyResourceType</type-name-ref>
        <resource-name>MyResource</resource-name>
        <actions>write</actions>
      </member-resource>
    </member-resources>
  </permission-set>
</permission-sets>

<jazn-policy>
  <grant>
    <grantee>
      <principals>
        <principal>
          <class>
            oracle.security.jps.service.policystore.ApplicationRole</class>
          <name>AppRole</name>
          <guid>F5494E409CFB11DEBFEB11296284F58</guid>
        </principal>
      </principals>
    </grantee>

    <!-- entitlement-based permissions -->
    <permission-set-refs>
      <permission-set-ref>
        <name>MyEntitlement</name>
      </permission-set-ref>
    </permission-set-refs>
  </grant>
</jazn-policy>
</application>
</applications>
</policy-store>
<jazn-policy></jazn-policy>
</jazn-data>

```

web.xml

The filter `JpsFilter` is configured as follows:

```

<web-app>
  <display-name>PolicyTest: PolicyServlet</display-name>
  <filter>
    <filter-name>JpsFilter</filter-name>
    <filter-class>oracle.security.jps.ee.http.JpsFilter</filter-class>
    <init-param>
      <param-name>application.name</param-name>
      <param-value>PolicyServlet</param-value>
    </init-param>
  </filter>
  <filter-mapping>
    <filter-name>JpsFilter</filter-name>
    <servlet-name>PolicyServlet</servlet-name>
    <dispatcher>REQUEST</dispatcher>
  </filter-mapping>...

```

Code Example

In the following example, `Subject.doAsPrivileged` may be replaced by `JpsSubject.doAsPrivileged`:

```
import javax.security.auth.Subject;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletOutputStream;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.PrintWriter;
import java.io.StringWriter;
import java.security.*;
import java.util.Date;
import java.util.PropertyPermission;
import java.io.FilePermission;

public class PolicyServlet extends HttpServlet {

    public PolicyServlet() {
        super();
    }

    public void init(ServletConfig config)
        throws ServletException {
        super.init(config);
    }

    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        final ServletOutputStream out = response.getOutputStream();

        response.setContentType("text/html");
        out.println("<HTML><BODY bgcolor=\"#FFFFFF\">");
        out.println("Time stamp: " + new Date().toString());
        out.println(" <br>request.getRemoteUser = " + request.getRemoteUser() +
"<br>");
        out.println("request.isUserInRole('sr_developer') = " +
request.isUserInRole("sr_developer") + "<br>");
        out.println("request.getUserPrincipal = " + request.getUserPrincipal() +
"<br>");

        Subject s = null;
        s = Subject.getSubject(AccessController.getContext());

        out.println("Subject in servlet " + s);
        out.println("<br>");
        final RuntimePermission rtPerm = new RuntimePermission("getClassLoader");
        try {
            Subject.doAsPrivileged(s, new PrivilegedAction() {
                public Object run() {
                    try {
                        AccessController.checkPermission(rtPerm);
                        out.println("<br>");
                        out.println("CheckPermission passed for permission: " +
rtPerm+ " seeded in application policy");
                        out.println("<br>");
                    } catch (IOException e) {
```

```

        e.printStackTrace();
        printException ("IOException", e, out);
    } catch (AccessControlException ace) {
        ace.printStackTrace();
        printException ("Accesscontrol Exception", ace, out);
    }
    return null;
}
}, null);

} catch (Throwable e) {
    e.printStackTrace();
    printException("application policy check failed", e, out);
}
out.println("</BODY>");
out.println("</HTML>");
}

void printException(String msg, Throwable e, ServletOutputStream out) {
    Throwable t;
    try {
        StringWriter sw = new StringWriter();
        PrintWriter pw = new PrintWriter(sw, true);
        e.printStackTrace(pw);

        out.println("<p>" + msg + "<p>");
        out.println("<code>");
        out.println(sw.getBuffer().toString());
        t = e;
        /* Print the root cause */
        while ((t = t.getCause()) != null) {
            sw = new StringWriter();
            pw = new PrintWriter(sw, true);
            t.printStackTrace(pw);

            out.println("<hr>");
            out.println("<p> Caused By ... </p>");
            out.println(sw.getBuffer().toString());
        }
        out.println("</code><p>");
    } catch (IOException ioe) {
        ioe.printStackTrace();
    }
}
}
}

```

20.3.3.2 Using the Methods doAs and doAsPrivileged

Oracle Fusion Middleware supports the methods `doAs` and `doAsPrivileged` in the standard class `javax.security.auth.Subject`.

Oracle recommends, however, the use of these methods in the class `oracle.security.jps.util.JpsSubject` because they render better performance and provide auditing.

Note: If `checkPermission` is called inside a `doAs` block and the check permission call fails, to display the failed protection domain you must set the system property `java.security.debug=access, failure`.

20.3.3.3 Using the Method `checkBulkAuthorization`

The method `checkBulkAuthorization` determines whether a Subject has access to one or more resource actions. Specifically, the method returns the set of resource actions the passed Subject is authorized to access in the passed resources.

When invoking this method (in a Java SE application), make sure that:

1. The system property `java.security.policy` has been set to the location of the OPSS/Oracle WebLogic Server policy file.
2. Your application *must* call first the method `setPolicy` to explicitly set the policy provider, as illustrated in the following lines:

```
java.security.Policy.setPolicy(new
oracle.security.jps.internal.policystore.JavaPolicyProvider())
```

3. Your application calls `checkBulkAuthorization()` after the call to `setPolicy`.

In any application, `checkBulkAuthorization` assumes that the caller can provide:

- A Subject with User and Enterprise Role Principals.
- A list of resources including the stripe each resource belongs to.

Grants using resource permissions must include the required resource type.

`checkBulkAuthorization` also assumes that the application has visibility into the policy store stripes configured in the domain where the application is running.

`checkBulkAuthorization` does not require resources to be present in the policy store.

20.3.3.4 Using the Method `getGrantedResources`

The method `getGrantedResources` provides a runtime authorization query to fetch all granted resources on a given Subject by returning the resource actions that have been granted to the Subject; only permissions associated with resource types (directly or indirectly through permission sets) are returned by this method, and it is available only when the policy store is LDAP-based.

20.3.4 The Class `ResourcePermission`

A permission class provides the means to control the actions that a grantee is allowed on a resource. Even though a custom permission class provides the application designer complete control over the actions, target matching, and the "implies" logic, to work as expected at runtime, a custom permission class must be specified in the system classpath of the server so that it is available and can be loaded when required. But modifying the system class path in environments is difficult and, in some environments, such modification might not be even possible.

OPSS includes the class `oracle.security.jps.ResourcePermission` that can be used as the permission class within any application grant to protect application or system resources. Therefore, the application developer no longer needs to write custom permission classes, since the class `ResourcePermission` is available out-of-the-box and can be readily used in permissions within application grants stored in any supported policy provider. This class is not designed to be used in system policies, but only in application policies.

Configuring Resource Permissions

A permission that uses the class `ResourcePermission` is called a *resource permission*, and it specifies the resource type, the resource name, and an optional list of actions according to the format illustrated in the following XML sample:

```
<permission>
  <class>oracle.security.jps.ResourcePermission</class>
  <name>resourceType=type,resourceName=name</name>
  <actions>character-separated-list-of-actions</actions>
</permission>
```

The above specification *requires* that the resource type encoded in the type name be defined. Even though the resource type information is not used at runtime, its definition must be present for a resource permission to be migrated successfully; moreover, resource types help administrators model resources and manage their use.

The following fragments illustrate the specifications of resource permissions and the corresponding required resource types:

```
<permission>
  <class>oracle.security.jps.ResourcePermission</class>
  <name>resourceType=epm.calcmgr.permission,resourceName=EPM_Calc_Manager</name>
</permission>
```

```
<resource-types>
  <resource-type>
    <name>epm.calcmgr.permission</name>
    <display-name>CalcManager ResourceType</display-name>
    <description>Resourcetype for managing CalcManager grants</description>
    <provider-name></provider-name>
    <matcher-class>oracle.security.jps.ResourcePermission</matcher-class>
    <actions-delimiter></actions-delimiter>
    <actions></actions>
  </resource-type>
</resource-types>
```

```
<permission>
  <class>oracle.security.jps.ResourcePermission</class>
  <name>resourceType=oracle.bi.publisher.Reports,resourceName=GLReports</name>
  <actions>develop;schedule</actions>
</permission>
```

```
<resource-types>
  <resource-type>
    <name>oracle.bi.publisher.Reports</name>
    <display-name>BI Publisher Reports</display-name>
    <provider-name></provider-name>
    <matcher-class>oracle.security.jps.ResourcePermission</matcher-class>
    <actions-delimiter></actions-delimiter>
    <actions>view;develop;schedule</actions>
  </resource-type>
</resource-types>
```

Note that a resource type associated with a resource permission can have an empty list of actions. The following important points apply to a resource permission:

- The name must conform to the following format:

```
resourceType=aType,resourceName=aName
```

The resource type of a resource permission must be defined and it is returned by the method `ResourcePermission.getType()`.

- The character-separated list of actions is optional; if specified, it must be a subset of the actions specified in the associated resource type. This list is returned by the method `ResourcePermission.getActions()`.

The character used to separate the items of the list must equal to the character specified in the `<actions-delimiter>` of the associated resource type.

- The display name of a resource used in a permission is returned by the method `ResourcePermission.getResourceName()`.
- No wildcard use is supported in a resource permission.

Managing and Checking Resource Permissions

The code snippet below illustrates the instantiation of a resource permission and how to check it programmatically; the following code snippet is based on one of the configuration examples described in [Configuring Resource Permissions](#):

```
ResourcePermission rp =
    new ResourcePermission("oracle.bi.publisher.Reports", "GLReports", "develop");
JpsAuth.checkPermission(rp);
```

At runtime the permission check will succeed if the resource permission satisfies all the following four conditions:

- The permission is an instance of the class `ResourcePermission`.
- The resource type name (first argument) matches (ignoring case) the name of a resource type.
- The resource (second argument) name matches exactly the name of a resource instance.
- The list of actions (third argument) is a comma-separated subset of the set of actions specified in the resource type.

About the Matcher Class for a Resource Type

When creating a resource type, a matcher class can be optionally supplied. If unspecified, it defaults to `oracle.security.jps.ResourcePermission`.

If, however, two or more resource types are to share the same resource matcher class, then that class must be one of the following:

- The class `oracle.security.jps.ResourcePermission`.
- A concrete class extending the abstract class `oracle.security.jps.AbstractTypedPermission`, as illustrated by the class `MyAbstractTypedPermission` in the following sample:

```
public class MyAbstractTypedPermission extends AbstractTypedPermission {
    private static final long serialVersionUID = 8665318227676708586L;
    public MyAbstractTypedPermission(String resourceType,
                                    String resourceName,
                                    String actions) {super(resourceType,
                                                            resourceName, actions);
    }
}
```

- A class implementing the class `oracle.security.jps.TypePermission` and extending the class `java.security.Permission`.

Manually Configuring Java EE Applications to Use OPSS

This chapter describes the manual configuration and packaging recommended for Java EE applications that use OPSS but do not use Oracle ADF security. Note that, nevertheless, some topics apply also to Oracle ADF applications.

The information is directed to developers that want to configure and package a Java EE application outside Oracle JDeveloper environment.

This chapter is divided into the following sections:

- [Configuring the Servlet Filter and the EJB Interceptor](#)
- [Choosing the Appropriate Class for Enterprise Groups and Users](#)
- [Packaging a Java EE Application Manually](#)
- [Configuring Applications to Use OPSS](#)

The files relevant to application management during development, deployment, runtime, and post-deployment are the following:

- `DOMAIN_HOME/config/fmwconfig/jps-config.xml`
- `DOMAIN_HOME/config/fmwconfig/system-jazn-data.xml`
- `jazn-data.xml` (in application EAR file)
- `cwallet.sso` (in application EAR file)
- `web.xml` (in application EAR file)
- `weblogic-application.xml` (in application EAR file)

21.1 Configuring the Servlet Filter and the EJB Interceptor

OPSS provides a servlet filter, the `JpsFilter`, and an EJB interceptor, the `JpsInterceptor`. The first one is configured in the file `web.xml` packed in a WAR file; the second one in the file `ejb-jar.xml` packed in a JAR file. OPSS also provides a way to configure in the file `web.xml` the stripe that application Mbeans should access; for details, see [Configuring the Application Stripe for Application MBeans](#).

All of them are available on WebLogic and WebSphere. The configuration available differs slightly according to the server platform as follows:

On WebLogic, the `JpsFilter` is out-of-the-box automatically set with default parameter values and need not be explicitly configured in the deployment descriptor; it needs to be configured manually only if a value different from the default value is required. The `JpsInterceptor` must be manually configured.

On WebSphere, both the `JpsFilter` and the `JpsInterceptor` must be manually configured.

Note: Oracle JDeveloper automatically inserts the required servlet filter (`JpsFilter`) and EJB interceptor (`JpsInterceptor`) configurations for Oracle ADF applications.

The manual configurations explained in this section are required *only* if you are packaging or configuring a Java EE application using the OPSS features detailed next *outside* the Oracle JDeveloper environment.

OPSS allows the specification of the application stripe used by MBeans; for details, see [Configuring the Application Stripe for Application MBeans](#).

The servlet filter and the EJB interceptor can be configured using the same set of parameters to customize the following features of a servlet or of an Enterprise Java Bean (EJB):

- [Application Name \(Stripe\)](#)
- [Application Roles Support](#)
- [Anonymous User and Anonymous Role Support](#)
- [Authenticated Role Support](#)
- [JAAS Mode](#)

The application name, better referred to as the *application stripe* and optionally specified in the application `web.xml` file, is used at runtime to determine which set of policies are applicable. If the application stripe is not specified, it defaults to the application id (which includes the application name).

An application stripe defines a subset of policies in the policy store. An application wanting to use that subset of policies would define its application stripe with a string identical to that application name. In this way, different applications can use the same subset of policies in the policy store.

The function of the anonymous and authenticated roles is explained in sections [The Anonymous User and Role](#) and [The Authenticated Role](#).

A servlet specifies the use a filter with the element `<filter-mapping>`. There must be one such element per filter per servlet.

An EJB specifies the use of an interceptor with the element `<interceptor-binding>`. There must be one such element per interceptor per EJB. For more details, see [Interceptor Configuration Syntax](#).

For a summary of the available parameters, see [Summary of Filter and Interceptor Parameters](#).

Application Name (Stripe)

This value is controlled by the following parameter:

`application.name`

The specification of this parameter is optional and case sensitive; if unspecified, it defaults to the name of the deployed application. Its value defines the subset of policies in the policy store that the application intends to use.

One way of specifying the application stripe is withing the filter element, as illustrated in the following sample:

```
<filter>
  <filter-name>JpsFilter</filter-name>
  <filter-class>oracle.security.jps.ee.http.JpsFilter</filter-class>
  <init-param>
    <param-name>application.name</param-name>
    <param-value>stripeid</param-value>
  </init-param>
</filter>
```

Another way to specify it, is to specify it is within the context-param element as illustrated in the following sample:

```
<context-param>
  <description>JPS custom stripe id</description>
  <param-name>application.name</param-name>
  <param-value>stripeid</param-value>
</context-param>
```

This last configuration is required if the application contains MBeans accessing the application policy store and the application name is different from the application stripe name. For details, see [Configuring the Application Stripe for Application MBeans](#).

Configuration Examples

The following two samples illustrate the configuration of this parameter for a servlet and for an EJB.

The following fragment of a `web.xml` file shows how to configure two different servlets, `MyServlet1` and `MyServlet2`, to be enabled with the filter so that subsequent authorization checks evaluate correctly. Note that servlets in the same WAR file always use the same policy stripe.

```
<filter>
  <filter-name>JpsFilter</filter-name>
  <filter-class>oracle.security.jps.ee.http.JpsFilter</filter-class>
  <init-param>
    <param-name>application.name</param-name>
    <param-value>MyAppName</param-value>
  </init-param>
</filter>
<filter-mapping>
  <filter-name>JpsFilter</filter-name>
  <servlet-name>MyServlet1</servlet-name>
  <dispatcher>REQUEST</dispatcher>
</filter-mapping>
<filter-mapping>
  <filter-name>JpsFilter</filter-name>
  <servlet-name>MyServlet2</servlet-name>
  <dispatcher>REQUEST</dispatcher>
</filter-mapping>
```

The following fragment of an `ejb-jar.xml` file illustrates the setting of the application stripe of an interceptor to `MyAppName` and the use of that interceptor by the EJB `MyEjb`:

```
<interceptor>
  <interceptor-class>oracle.security.jps.ee.ejb.JpsInterceptor</interceptor-class>
```

```

    <env-entry>
      <env-entry-name>application.name</env-entry-name>
      <env-entry-type>java.lang.String</env-entry-type>
      <env-entry-value>MyAppName</env-entry-value>
      <injection-target>
        <injection-target-class>
          oracle.security.jps.ee.ejb.JpsInterceptor</injection-target-class>
          <injection-target-name>application_name</injection-target-name>
        </injection-target>
      </env-entry>
    </interceptor>
    ...
    <interceptor-binding>
      <ejb-name>MyEjb</ejb-name>
      <interceptor-class>
        oracle.security.jps.ee.ejb.JpsInterceptor</interceptor-class>
      </interceptor-binding>

```

Note how the preceding example satisfies the interceptor configuration syntax requirements.

Application Roles Support

The addition of application roles to a subject is controlled by the following parameter, which can be set to true or false:

```
add.application.roles
```

To add application roles to a subject, set the property to true; otherwise, set it to false. The default value is true.

The principal class for the application role is:

```
oracle.security.jps.service.policystore.ApplicationRole
```

Anonymous User and Anonymous Role Support

The use of anonymous for a servlet is controlled by the following parameters, which can be set to true or false:

```
enable.anonymous
remove.anonymous.role
```

For an EJB, only the second parameter above is available, since the use of the anonymous user and role is always enabled for EJBs.

To enable the use of the anonymous user for a servlet, set the first property to true; to disable it, set it to false. The default value is true.

To remove the anonymous role from a subject, set the second property to true; to retain it, set it to false. The default value is false. Typically, one would want to remove the anonymous user and role after authentication, and only in special circumstances would want to retain them after authentication.

The default name and the principal class for the anonymous user are:

```
anonymous
oracle.security.jps.internal.core.principals.JpsAnonymousUserImpl
```

The default name and the principal class for the anonymous role are:

```
anonymous-role
oracle.security.jps.internal.core.principals.JpsAnonymousRoleImpl
```

The following fragment of a `web.xml` file illustrates a setting of these parameters and the use of the filter `JpsFilter` by the servlet `MyServlet`:

```
<filter>
  <filter-name>JpsFilter</filter-name>
  <filter-class>oracle.security.jps.ee.http.JpsFilter</filter-class>
  <init-param>
    <param-name>enable.anonymous</param-name>
    <param-value>true</param-value>
  </init-param>
  <init-param>
    <param-name>remove.anonymous.role</param-name>
    <param-value>>false</param-value>
  </init-param>
</filter>
<filter-mapping>
  <filter-name>JpsFilter</filter-name>
  <servlet-name>MyServlet</servlet-name>
  <dispatcher>REQUEST</dispatcher>
</filter-mapping>
```

The following fragment of an `ejb-jar.xml` file illustrates the setting of the second parameter to false and the use of the interceptor by the Enterprise Java Bean `MyEjb`:

```
<interceptor>
  <interceptor-class>oracle.security.jps.ee.ejb.JpsInterceptor</interceptor-class>
  <env-entry>
    <env-entry-name>remove.anonymous.role</env-entry-name>
    <env-entry-type>java.lang.Boolean</env-entry-type>
    <env-entry-value>>false</env-entry-value>
    <injection-target>
      <injection-target-class>
        oracle.security.jps.ee.ejb.JpsInterceptor</injection-target-class>
      <injection-target-name>remove_anonymous_role</injection-target-name>
    </injection-target>
  </env-entry>
</interceptor>
...
<interceptor-binding>
  <ejb-name>MyEjb</ejb-name>
  <interceptor-class>
    oracle.security.jps.ee.ejb.JpsInterceptor</interceptor-class>
</interceptor-binding>
```

The following fragments illustrate how to access programmatically the anonymous subject, and the anonymous role and anonymous user from a subject:

```
import oracle.security.jps.util.SubjectUtil;

// The next call returns the anonymous subject
javax.security.auth.Subject subj = SubjectUtil.getAnonymousSubject();

// The next call extracts the anonymous role from the subject
java.security.Principal p =
SubjectUtil.getAnonymousRole(javax.security.auth.Subject subj)
// Remove or retain anonymous role
...

// The next call extracts the anonymous user from the subject
java.security.Principal p =
SubjectUtil.getAnonymousUser(javax.security.auth.Subject subj)
```

```
// Remove or retain anonymous user
...
```

Authenticated Role Support

The use of the authenticated role is controlled by the following parameter, which can be set to true or false:

```
add.authenticated.role
```

To add the authenticated role to a subject, set the parameter to true; otherwise it, set it to false. The default value is true.

The default name and the principal class for the authenticated role are:

```
authenticated-role
oracle.security.jps.internal.core.principals.JpsAuthenticatedRoleImpl
```

The following fragment of a `web.xml` file illustrates a setting of this parameter and the use of the filter `JpsFilter` by the servlet `MyServlet`:

```
<filter>
  <filter-name>JpsFilter</filter-name>
  <filter-class>oracle.security.jps.ee.http.JpsFilter</filter-class>
  <init-param>
    <param-name>add.authenticated.role</param-name>
    <param-value>>false</param-value>
  </init-param>
</filter>
<filter-mapping>
  <filter-name>JpsFilter</filter-name>
  <servlet-name>MyServlet</servlet-name>
  <dispatcher>REQUEST</dispatcher>
</filter-mapping>
```

JAAS Mode

The use of JAAS mode is controlled by the following parameter:

```
oracle.security.jps.jaas.mode
```

This parameter can be set to:

```
doAs
doAsPrivileged
off
undefined
subjectOnly
```

The default value is `doAsPrivileged`. For details on how these values control the behavior of the method `checkPermission`, see [Section 20.3.3.1, "Using the Method `checkPermission`."](#)

The following two samples illustrate configurations of a servlet and an EJB that use this parameter.

The following fragment of a `web.xml` file illustrates a setting of this parameter and the use of the filter `JpsFilter` by the servlet `MyServlet`:

```
<filter>
  <filter-name>JpsFilter</filter-name>
  <filter-class>oracle.security.jps.ee.http.JpsFilter</filter-class>
  <init-param>
    <param-name>oracle.security.jps.jaas.mode</param-name>
```

```

        <param-value>doAs</param-value>
    </init-param>
</filter>
<filter-mapping>
    <filter-name>JpsFilter</filter-name>
    <servlet-name>MyServlet</servlet-name>
    <dispatcher>REQUEST</dispatcher>
</filter-mapping>

```

The following fragment of an `ejb-jar.xml` file illustrates a setting of this parameter to `doAs` and the use of the interceptor `JpsInterceptor` by the Enterprise Java Bean `MyEjb`:

```

<interceptor>
    <interceptor-class>oracle.security.jps.ee.ejb.JpsInterceptor</interceptor-class>
    <env-entry>
        <env-entry-name>oracle.security.jps.jaas.mode</env-entry-name>
        <env-entry-type>java.lang.String</env-entry-type>
        <env-entry-value>doAs</env-entry-value>
        <injection-target>
            <injection-target-class>
                oracle.security.jps.ee.ejb.JpsInterceptor</injection-target-class>
            <injection-target-name>oracle_security_jps_jaas_mode
                </injection-target-name>
        </injection-target>
    </env-entry>
</interceptor>
...
<interceptor-binding>
    <ejb-name>MyEjb</ejb-name>
    <interceptor-class>
        oracle.security.jps.ee.ejb.JpsInterceptor</interceptor-class>
</interceptor-binding>

```

21.1.1 Interceptor Configuration Syntax

The following requirements and characteristics of the specifications apply to all parameters configured for the `JpsInterceptor`:

- The setting of a parameter requires specifying its type (in the element `<env-entry-type>`).
- The setting of a parameter requires the element `<injection-target>`, which specifies the same class as that of the interceptor (in the element `<injection-target-class>`), and the parameter name rewritten as a string where the dots are replaced by underscores (in the element `<injection-target-name>`).
- The binding of an interceptor to an EJB is specified by the EJB name and the interceptor's class, that is, the interceptor is referred to by its class, not by name.

21.1.2 Summary of Filter and Interceptor Parameters

The following table summarizes the description of the parameters used by the `JpsFilter` and the `JpsInterceptor`:

Table 21–1 Summary of JpsFilter and JpsInterceptor Parameters

Parameter Name	Values	Default	Function	Notes
application.name	Any valid string. The value is case sensitive.	The name of the deployed application.	To specify the subset of policies that the servlet or EJB is to use.	It should be specified if several servlets or EJBs are to share the same subset of policies in the policy store.
add.application.roles	TRUE or FALSE	TRUE	To add application roles to a Subject.	Since it defaults to TRUE, it must be set (to FALSE) only if the application is not to add application roles to a Subject.
enable.anonymous	TRUE or FALSE	TRUE	To enable or disable the anonymous user in a Subject.	If set to TRUE, it creates a Subject with the anonymous user and the anonymous role.
remove.anonymous.role	TRUE or FALSE	FALSE	To keep or remove the anonymous role from a Subject after authentication.	Available for servlets only. For EJBs, the anonymous role is always removed from a Subject. If set to FALSE, the Subject retains the anonymous role after authentication; if set to TRUE, it is removed after authentication.
add.authenticated.role	TRUE or FALSE	TRUE	To allow addition of the authenticated role in a Subject.	Since it defaults to TRUE, it needs be set (to FALSE) only if the authenticated role is not be included in a Subject.
oracle.security.jps.jaas.mode	doAsPrivileged doAs off undefined subjectOnly	doAsPrivileged	To set the JAAS mode.	

21.1.3 Configuring the Application Stripe for Application MBeans

If your application satisfies the following conditions:

- It contains MBeans that access the policy store and perform authorization checks.
- The application stripe name is not equal to the application name.

then, for the MBean to access the application stripe in the domain security store, the stripe name must be specified by the global parameter (or context parameter) `application.name` in the file `web.xml`, as illustrated in the following sample:

```
<context-param>
  <description>JPS custom stripe id</description>
  <param-name>application.name</param-name>
  <param-value>stripeid</param-value>
</context-param>
```


21.2 Choosing the Appropriate Class for Enterprise Groups and Users

Note: If you are using Oracle JDeveloper, the tool chooses the appropriate classes. Therefore, the configuration explained next is only necessary if policies are entered outside the Oracle JDeveloper environment.

The classes specified in members of an application role must be either other application role class or one of the following:

```
weblogic.security.principal.WLSUserImpl
weblogic.security.principal.WLSGroupImpl
```

The following fragment illustrates the use of these classes in the specification of enterprise groups (in bold face).

Important: Application role names are case *insensitive*; for example, `app_operator` in the following sample.

Enterprise user and group names are case *sensitive*; for example, `Developers` in the following sample.

For related information about case, see [Section L.4, "Failure to Grant or Revoke Permissions - Case Mismatch."](#)

```
<app-role>
  <name>app_monitor</name>
  <display-name>application role monitor</display-name>
  <class>oracle.security.jps.service.policystore.ApplicationRole</class>
  <members>
    <member>
      <class>oracle.security.jps.service.policystore.ApplicationRole</class>
      <name>app_operator</name>
    </member>
    <member>
      <class>weblogic.security.principal.WLSGroupImpl</class>
      <name>Developers</name>
    </member>
  </members>
</app-role>
```

21.3 Packaging a Java EE Application Manually

This section explains the packaging requirements for a servlet or an EJB (using custom policies and credentials) that is to be deployed on WebLogic Application Server or WebSphere Application Server.

Application policies are defined in the file `jazn-data.xml`. The *only* supported way to include this file with an application is to package it in the directory `META-INF` of an EAR file.

Servlets are packaged in a WAR file that contains the configuration file `web.xml`; EJBs are packaged in a WAR file that contains the configuration file `ejb-jar.xml`. The WAR file must include the configuration of the filter `JpsFilter` (for servlets) or of the interceptor `JpsInterceptor` (for EJBs) in the corresponding configuration file.

The description that follows considers the packaging of a servlet and the configuration of the `JpsFilter` in the file `web.xml`, but it applies equally to the packaging of an EJB and the configuration of the `JpsInterceptor` in the file `ejb-jar.xml`.

Important: Currently *all* `JpsFilter` configurations in *all* `web.xml` files in an EAR file *must* have the same configuration. Same constraints apply to the `JpsInterceptor`.

For details about the `JpsFilter` and the `JpsInterceptor`, see [Configuring the Servlet Filter and the EJB Interceptor](#).

The packaging requirements and assumptions for a Java EE application that wants to use custom policies and credentials are the following:

- The application to be deployed must be packaged in a single EAR file.
- The EAR file must contain exactly one file `META-INF/jazn-data.xml`, where application policies and roles are specified; these apply equally to all components in the EAR file.
- The EAR file may contain one or more WAR files.
- Each WAR or JAR file in the EAR file must contain exactly one `web.xml` (or `ejb-jar.xml`) where the `JpsFilter` (or `JpsInterceptor`) is configured, and such configurations in all EAR files must be identical.
- Component credentials in `cwallet.sso` files can be packaged in the EAR file. These credentials can be migrated to the credential store when the application is deployed with Oracle Enterprise Manager Fusion Middleware Control.

Note: If a component should require a filter configuration different from that of other components, then it must be packaged in a separate EAR file and deployed separately.

21.3.1 Packaging Policies with Application

Application policies are defined in the file `jazn-data.xml`. The *only* supported way to include this file with an application is to package it in the directory `META-INF` of an EAR file. The EAR file may contain zero or more WAR files, but the policies can be specified only in that XML file located in that EAR directory. To specify particular policies for a component in a WAR file, that component must be packaged in a separate EAR file with its own `jazn-data.xml` file as specified above. No other policy package combination is supported in this release, and policy files other than the top `jazn-data.xml` are disregarded.

21.3.2 Packaging Credentials with Application

Application credentials are defined in a file that must be named `cwallet.sso`. The *only* supported way to include this file with an application is to package it in the directory `META-INF` of an EAR file. The EAR file may contain zero or more WAR files, but credentials can be specified only in that `cwallet.sso` file located in that EAR directory. To specify particular credentials for a component in a WAR file, that component must be packaged in a separate EAR file with its own `cwallet.sso` file as specified above. No other credential package combination is supported in this release, and credential files other than the top `cwallet.sso` are disregarded.

21.4 Configuring Applications to Use OPSS

This section describes several configurations that a developer would perform manually for a Java EE application developed outside the Oracle JDeveloper environment, in the following sections:

- [Parameters Controlling Policy Migration](#)
- [Policy Parameter Configuration According to Behavior](#)
- [Parameters Controlling Credential Migration](#)
- [Credential Parameter Configuration According to Behavior](#)
- [Using a Wallet-Based Credential Store](#)
- [Supported Permission Classes](#)
- [Specifying Bootstrap Credentials Manually](#)
- [Migrating Identities with migrateSecurityStore](#)
- [Example of Configuration File jps-config.xml](#)

Note: Use the system property `jps.deployment.handler.disabled` to disable the migration of application policies and credentials for applications deployed in a WebLogic Server.

When this system property is set to `TRUE`, the migration of policies and credentials at deployment is disabled for *all* applications regardless of the particular application settings in the application file `weblogic-application.xml`.

21.4.1 Parameters Controlling Policy Migration

The migration of application policies at deployment is controlled by several parameters configured in the file `META-INF/weblogic-application.xml`.

For details about the specification of parameters on WebSphere, see *Oracle Fusion Middleware Third-Party Application Server Guide*.

The parameters that control migration of policies during application deployment or redeployment, and the removal of policies during undeployment are the following:

- Migration
 - [jps.policystore.migration](#)
 - [jps.apppolicy.idstoreartifact.migration](#)
 - [jps.policystore.removal](#)
- Listener
 - [JpsApplicationLifecycleListener](#)
- Principal Validation
 - [jps.policystore.migration.validate.principal](#)
- Target of Migration (application stripe)
 - [jps.policystore.applicationid](#)

The configuration and function of each of the above is explained next.

Notes: Fusion Middleware Control allows setting of most of these parameters when the application is deployed, redeployed, or undeployed. For details, see [Section 6.2.1, "Deploying Java EE and Oracle ADF Applications with Fusion Middleware Control."](#)

The configurations explained next need be entered manually *only if* you are not using Fusion Middleware Control to manage your application.

When deploying an application that is using *file-based* stores to a managed server running in a computer different from that where the administration server is running, do not use the life cycle listener. Otherwise, the data maintained by the managed server and the administration server would not match, and security may not work as expected. Instead of employing the life cycle listener, use the OPSS script `migrateSecurityStore` to migrate application policies and credentials to the domain stores.

The above remark applies *only* when using file-based stores.

jps.policystore.migration

This parameter specifies whether the migration should take place, and, when it does, whether it should merge with or overwrite matching policies present in the target store.

On WebLogic, it is configured as illustrated in the following fragment:

```
<wls:application-param>
  <wls:param-name>jps.policystore.migration</wls:param-name>
  <wls:param-value>Option</wls:param-value>
</wls:application-param>
```

Option stands for one of the following value is MERGE, OVERWRITE, or OFF.

For details about the configuration of this parameter on WebSphere, see *Oracle Fusion Middleware Third-Party Application Server Guide*.

Set to OFF to prevent policy migration; otherwise, set to MERGE to migrate and merge with existing policies, or to OVERWRITE to migrate and overwrite existing policies. The default value (at deploy) is MERGE.

jps.policystore.applicationid

This parameter specifies the target stripe into which policies are migrated.

On WebLogic, it is configured as illustrated in the following fragment:

```
<wls:application-param>
  <wls:param-name>jps.policystore.applicationid</wls:param-name>
  <wls:param-value>myApplicationStripe</wls:param-value>
</wls:application-param>
```

For details about the configuration of this parameter on WebSphere, see *Oracle Fusion Middleware Third-Party Application Server Guide*.

This parameter's value can be any valid string; if unspecified, Oracle WebLogic Server picks up a stripe name based on the application name and version, namely, *application_name#version*.

The value of this parameter must match the value of `application.name` specified for the `JpsServlet` (in the file `web.xml`) or for the `JpsInterceptor` (in the file `ejb-jar.xml`). For details, see [Application Name \(Stripe\)](#).

The value picked from `weblogic-application.xml` is used at deploy time; the value picked from `web.xml` or `ejb-jar.xml` is used at runtime.

JpsApplicationLifecycleListener

This parameter is supported on WebLogic only, and it must be set as illustrated in the following fragment:

```
<wls:listener>
  <wls:listener-class>
    oracle.security.jps.wls.listeners.JpsApplicationLifecycleListener
  </wls:listener-class>
</wls:listener>
```

jps.apppolicy.idstoreartifact.migration

This parameter is supported on WebLogic only, and it specifies whether the policy migration should exclude migrating references to enterprise users or groups, such as application roles grants to enterprise users or groups, and permission grants to enterprise users or groups; thus it allows the migration of *just* application policies and, when enabled, the migration ignores the mapping of application roles to enterprise groups or users.

It is configured as illustrated in the following fragment:

```
<wls:application-param>
  <wls:param-name>jps.apppolicy.idstoreartifact.migration</wls:param-name>
  <wls:param-value>Option</wls:param-value>
</wls:application-param>
```

Option stands for one of the values TRUE or FALSE. Set to FALSE to exclude the migration of artifacts referencing enterprise users or groups; otherwise, set it to TRUE; if unspecified, it defaults to TRUE.

Important: When an application is deployed with this parameter set to **FALSE** (that is, to exclude the migration of non-application specific policies), before the application can be used in the domain, the administrator should perform the mapping of application roles to enterprise groups or users with Fusion Middleware Control or the WebLogic Administration Console.

Note how this setting allows the administrator further control over application roles.

The following examples show fragments of the same `jazn-data.xml` files. This file, packaged in the application EAR file, describes the application authorization policy.

The file `system-jazn-data.xml` represents the domain file-based policy store into which application policies are migrated (and used in the example for simplicity).

It is assumed that the parameter `jps.apppolicy.idstoreartifact.migration` has been set to FALSE.

```
<!-- Example 1: app role applicationDeveloperRole in jazn-data.xml that references
the enterprise group developers -->
<app-role>
```

```

<class>weblogic.security.principal.WLSGroupImpl</class>
  <name>applicationDeveloperRole</name>
  <display-name>application role applicationDeveloperRole</display-name>
  <members>
    <member>
      <class>weblogic.security.principal.WLSGroupImpl</class>
      <name>developers</name>
    </member>
  </members>
</app-role>

<!-- app role applicationDeveloperRole in system-jazn-data.xml after migration:
notice how the role developers has been excluded -->
<app-role>
  <name>applicationDeveloperRole</name>
  <display-name>application role applicationDeveloperRole</display-name>
  <guid>CB3633A0D0E811DDBF08952E56E4544A</guid>
  <class>weblogic.security.principal.WLSGroupImpl</class>
</app-role>

<!-- Example 2: app role viewerApplicationRole in jazn-data.xml makes reference
to the anonymous role -->
<app-role>
  <name>viewerApplicationRole</name>
  <display-name>viewerApplicationRole</display-name>
  <class>weblogic.security.principal.WLSGroupImpl</class>
  <members>
    <member>
      <class>
oracle.security.jps.internal.core.principals.JpsAnonymousRoleImpl
      </class>
      <name>anonymous-role</name>
    </member>
  </members>
</app-role>

<!-- app role viewerApplicationRole in system-jazn-data.xml after migration:
notice that references to the anonymous role are never excluded -->
<app-role>
  <name>viewerApplicationRole</name>
  <display-name>viewerApplicationRole</display-name>
  <guid>CB3D86A0D0E811DDBF08952E56E4544A</guid>
  <class>weblogic.security.principal.WLSGroupImpl</class>
  <members>
    <member>
      <class>
oracle.security.jps.internal.core.principals.JpsAnonymousRoleImpl
      </class>
      <name>anonymous-role</name>
    </member>
  </members>
</app-role>

```

jps.policystore.removal

This parameter specifies whether the removal of policies at undeployment should *not* take place.

On WebLogic, it is configured as illustrated in the following fragment:

```
<wls:application-param>
```

```
<wls:param-name>jps.policystore.removal</wls:param-name>
<wls:param-value>OFF</wls:param-value>
</wls:application-param>
```

For details about the configuration of this parameter on WebSphere, see *Oracle Fusion Middleware Third-Party Application Server Guide*.

When set, the parameter's value must be OFF. By default, it is not set.

Set to OFF to prevent the removal of policies; if not set, policies are removed.

The above setting should be considered when multiple applications are sharing the same application stripe. The undeploying application would choose not to remove application policies because other applications may be using the common set of policies.

Note: Deciding to set this parameter to OFF for a given application requires knowing, at the time the application is deployed, whether the application stripe is shared by other applications.

jps.policystore.migration.validate.principal

This parameter is supported on WebLogic only, and it specifies whether the check for principals in system and application policies at deployment or redeployment should take place.

It is configured as illustrated in the following fragment:

```
<wls:application-param>
  <wls:param-name>jps.policystore.migration.validate.principal</wls:param-name>
  <wls:param-value>TRUE</wls:param-value>
</wls:application-param>
```

When set, the parameter's value must be TRUE or FALSE.

When set to TRUE the system checks the validity of enterprise users and groups: if a principal (in an application or system policy) refers to an enterprise user or group not found in the identity store, a warning is issued. When set to FALSE, the check is skipped.

If not set, the parameter value defaults to FALSE.

Validation errors are logged in the server log, and they do not terminate the operation.

21.4.2 Policy Parameter Configuration According to Behavior

This section describes the settings required to manage application policies with the following behaviors:

- [To Skip Migrating All Policies](#)
- [To Migrate All Policies with Merging](#)
- [To Migrate All Policies with Overwriting](#)
- [To Remove \(or Prevent the Removal of\) Application Policies](#)
- [To Migrate Policies in a Static Deployment](#)

Any value settings other than the ones described in the following sections are not recommended and may lead to unexpected migration behavior. For more details, see [Recommendations](#).

All behaviors can be specified with Fusion Middleware Control when the application is deployed, redeployed, or undeployed with that tool.

21.4.2.1 To Skip Migrating All Policies

The following matrix shows the settings that prevent the migration from taking place:

Table 21–2 Settings to Skip Policy Migration

	Valid at deploy or redeploy
JpsApplicationLifecycleListener	Set
jps.policystore.migration	OFF

Typically, you would skip migrating policies when redeploying the application when you want to keep domain policies as they are, but you would migrate policies when deploying the application for the first time.

21.4.2.2 To Migrate All Policies with Merging

The following matrix shows the setting of required and optional parameters that migrates only policies that are not in the target store (optional parameters are enclosed in between brackets):

Table 21–3 Settings to Migrate Policies with Merging

	Valid at deploy or redeploy
JpsApplicationLifecycleListener	Set
jps.policystore.migration	MERGE
[jps.policystore.applicationid]	Set to the appropriate string. Defaults to servlet or EJB name.
[jps.apppolicy.idstoreartifact.migration]	Set to FALSE to exclude migrating policies that reference enterprise artifacts; otherwise set to TRUE. Defaults to TRUE.
[jps.policystore.migration.validate.principal]	Set to TRUE to validate enterprise users and roles in application and system policies. Set to FALSE, otherwise. If unspecified, it defaults to FALSE.

Typically, you would choose migrating policies with merging at redeploy when the policies have changed and you want to add to the existing policies.

21.4.2.3 To Migrate All Policies with Overwriting

The following matrix shows the setting that migrates all policies overwriting matching target policies (optional parameters are enclosed in between brackets):

Table 21–4 Settings to Migrate Policies with Overwriting

	Valid at deploy or redeploy
JpsApplicationLifecycleListener	Set
jps.policystore.migration	OVERWRITE
[jps.policystore.migration.validate.principal]	Set to TRUE to validate enterprise users and roles in application and system policies. Set to FALSE, otherwise. If unspecified, it defaults to FALSE.

Typically, you would choose migrating policies with overwriting at redeploy when a new set of policies should replace existing policies. Note that if the optional parameter `jps.policy.migration.validate.principal` is needed, it must be set manually.

21.4.2.4 To Remove (or Prevent the Removal of) Application Policies

The removal of application policies at undeployment is limited since code source grants in the system policy are *not* removed. For details, see example in [What Gets Removed and What Remains](#).

The following matrix shows the setting that removes policies at undeployment:

Table 21–5 Settings to Remove Policies

	Valid at undeploy
JpsApplicationLifecycleListener	Set
jps.policystore.removal	Not set (default)

Note: The policies removed at undeploy are determined by the stripe that the application specified at deploy or redeploy. If an application is redeployed with a stripe specification different than the original one, then policies in that stripe (the original) are not removed.

The following matrix shows the setting that *prevents* the removal of application policies at undeployment:

Table 21–6 Settings to Prevent the Removal of Policies

	Valid at undeploy
JpsApplicationLifecycleListener	Set
jps.policystore.removal	OFF

Note: Deciding to set this parameter to OFF for a given application requires knowing, at the time the application has been deployed, whether the application stripe is shared by other applications.

What Gets Removed and What Remains

Consider the application `myApp`, which has been configured for automatic migration and removal of policies. The following fragment of the application's `jazn-data.xml` file (packed in the application EAR file) illustrates the application policies that are migrated when the application is deployed with Fusion Middleware Control and those that are and are *not* removed when the application is undeployed with Fusion Middleware Control:

```
<jazn-data>
  <policy-store>
    <applications>
      <!-- The contents of the following element application is migrated
           to the element policy-store in domain system-jazn-data.xml;
           when myApp is undeployed with EM, it is removed from domain store -->
      <application>
        <name>myApp</name>
```

```

    <app-roles>
      <app-role>
        <class>oracle.security.jps.service.policystore.SomeRole</class>
        <name>applicationDeveloperRole</name>
        <display-name>application role applicationDeveloperRole</display-name>
        <members>
          <member>
            <class>oracle.security.somePath.JpsXmlEnterpriseRoleImpl</class>
            <name>developers</name>
          </member>
        </members>
      </app-role>
    </app-roles>
  <jazn-policy>
    <grant>
      <grantee>
        <principals>
          <principal>
            <class>oracle.security.jps.service.policystore.ApplicationRole</class>
            <name>applicationDeveloperRole</name>
          </principal>
        </principals>
      </grantee>
      <permissions>
        <permission>
          <class>oracle.security.jps.JpsPermission</class>
          <name>loadPolicy</name>
        </permission>
      </permissions>
    </grant>
  </jazn-policy>
</application>
</applications>
</policy-store>

<jazn-policy>
<!-- The following codebase application grant is migrated to the element
      jazn-policy in domain system-jazn-data.xml; when myApp is undeployed
      with EM, it is not removed from domain store -->
    <grant>
      <grantee>
        <codesource>
          <url>file:${domain.home}/servers/${weblogic.Name}/Foo.ear/-</url>
        </codesource>
      </grantee>
      <permissions>
        <permission>
          <class>oracle.security.jps.service.credstore.CredentialAccessPermission</class>
          <name>context=SYSTEM,mapName=*,keyName=*</name>
          <actions>*</actions>
        </permission>
      </permissions>
    </grant>
  </jazn-policy>
</jazn-data>

```

To summarize: in regards to what gets removed, the important points to remember are the following:

- All data inside the element `<application>` can be automatically removed at undeployment. In case of an LDAP-based policy store, the application scoped authorization policy data nodes get cleaned up.
- All data inside the element `<jazn-policy>` *cannot* be automatically removed at undeployment.

21.4.2.5 To Migrate Policies in a Static Deployment

Table 21–7 shows the setting that migrates application policies when the application is statically deployed. The MERGE or OVERWRITE operation takes place only if the application policies do not already exist in the domain.

Table 21–7 Settings to Migrate Policies with Static Deployments

<code>JpsApplicationLifecycleListener</code>	Set
<code>jps.policystore.migration</code>	MERGE or OVERWRITE

Table 21–8 shows the setting that skip the migration of application policies when the application is statically deployed.

Table 21–8 Settings Not to Migrate Policies with Static Deployments

<code>JpsApplicationLifecycleListener</code>	Set
<code>jps.policystore.migration</code>	OFF

21.4.2.6 Recommendations

Keep in mind the following suggestions:

When a LDAP-based policy store is used and the application is to be deployed to multiple managed servers, then choose to migrate to one of the servers only. The rest of the deployments should choose *not* to migrate policies. This ensures that the policies are migrated only once from the application store to the policy store.

All the deployments must use the same application id.

Attempting policy migration to the same node for the same application multiple times (for example, on different managed servers) can result in policy migration failures. An alternative is to migrate the policy data to the store outside of the deployment process using the OPSS script `migrateSecurityStore`.

If, however, the application is deployed to several servers and the policy store is file-based, the deployment must include the administration server for the migration to update the policy file `$_DOMAIN_HOME/config/fmwconfig/system-jazn-data.xml`.

21.4.3 Using a Wallet-Based Credential Store

The content of a wallet-based credential store is defined in a file that must be named `cwallet.sso`. A wallet-based credential store is also referred to as a file-based credential store.

For instructions on how to create a wallet, see section Common Wallet Operations in *Oracle Fusion Middleware Administrator's Guide*.

The location of the file `cwallet.sso` is specified in the configuration file `jps-config.xml` with the element `<serviceInstance>`, as illustrated in the following example:

```
<serviceInstance name="credstore" provider="credstoressp">
  <property name="location" value="myCredStorePath"/>
</serviceInstance>
```

For other types of credential storage, see chapter Managing Keystores, Wallets, and Certificates in *Oracle Fusion Middleware Administrator's Guide*.

21.4.4 Parameters Controlling Credential Migration

The migration of application credentials at deployment is controlled by several parameters configured in the file `META-INF/weblogic-application.xml`.

For details about the specification of these parameters on WebSphere, see *Oracle Fusion Middleware Third-Party Application Server Guide*.

The parameter that controls credential migration is [jps.credstore.migration](#). The listener is [JpsApplicationLifecycleListener - Credentials](#).

jps.credstore.migration

This parameter specifies whether the migration should take place, and, when it does, whether it should merge with or overwrite matching credentials present in the target store.

On WebLogic, it is configured as illustrated in the following fragment:

```
<wls:application-param>
  <wls:param-name>jps.credstore.migration</wls:param-name>
  <wls:param-value>behaviorValue</wls:param-value>
</wls:application-param>
```

For details about the specification this parameter on WebSphere, see *Oracle Fusion Middleware Third-Party Application Server Guide*.

If set, this parameter's value must be one of the following: MERGE, OVERWRITE, or OFF. The OVERWRITE value is available on WebLogic only and when the server is running in development mode.

If not set, the migration of credentials takes place with the option MERGE.

JpsApplicationLifecycleListener - Credentials

This listener is supported only on WebLogic and it is configured as illustrated in the following fragment:

```
<wls:listener>
  <wls:listener-class>
    oracle.security.jps.wls.listeners.JpsApplicationLifecycleListener
  </wls:listener-class>
</wls:listener>
```

21.4.5 Credential Parameter Configuration According to Behavior

This section describes the manual settings required to migrate application credentials with the following behaviors:

- [To Skip Migrating Credentials](#)
- [To Migrate Credentials with Merging](#)
- [To Migrate Credentials with Overwriting](#)

Any value settings other than the ones described in the following sections are not recommended and may lead to unexpected migration behavior.

If the migration target is an LDAP-based credential store, it is recommended that the application be deployed to just one managed server or cluster. Otherwise, application credentials may not work as expected.

Note: Credentials are not deleted upon an application undeployment. A credential may have started its life as being packaged with an application, but when the application is undeployed credentials are *not* removed.

21.4.5.1 To Skip Migrating Credentials

The following matrix shows the setting that prevents the migration from taking place:

Table 21–9 Settings to Skip Credential Migration

	Valid at deploy or redeploy
jps.credstore.migration	OFF

21.4.5.2 To Migrate Credentials with Merging

The following matrix shows the setting of required and optional parameters that migrates only credentials that are not present in the target store (optional parameters are enclosed in between brackets):

Table 21–10 Settings to Migrate Credentials with Merging

	Valid at deploy or redeploy
JpsApplicationLifecycleListener	Set
jps.credstore.migration	MERGE

21.4.5.3 To Migrate Credentials with Overwriting

This operation is valid on WebLogic only and when the server is running in development mode. The following matrix shows the setting that migrates all credentials overwriting matching target credentials:

Table 21–11 Settings to Migrate Credentials with Overwriting

	Valid at deploy or redeploy
JpsApplicationLifecycleListener	Set
jps.credstore.migration	OVERWRITE
jps.app.credential.override.allowed	This system property must be set to TRUE

21.4.6 Supported Permission Classes

The components of a permission are illustrated in the following snippet from a `system-jazn-data.xml` file:

```
<grant>
  <grantee>
    <codesource>
      <url>file:${oracle.deployed.app.dir}/<MyApp>${oracle.deployed.app.ext}</url>
```

```

        </codesource>
    </grantee>
    <permissions>
        <permission>
            <class>
oracle.security.jps.service.policystore.PolicyStoreAccessPermission
            </class>
            <name>context=SYSTEM</name>
            <actions>getConfiguredApplications</actions>
        </permission>
        <permission>
            <class>
oracle.security.jps.service.policystore.PolicyStoreAccessPermission
            </class>
            <name>context=APPLICATION, name=*</name>
            <actions>getApplicationPolicy</actions>
        </permission>
    </permissions>
</grant>

```

This section describes the supported values for the elements `<class>`, `<name>`, and `<actions>` within a `<permission>`.

Important: All permission classes used in policies must be included in the class path, so the policy provider can load them when a service instance is initialized.

21.4.6.1 Policy Store Permission

Class name:

```
oracle.security.jps.service.policystore.PolicyStoreAccessPermission
```

When the permission applies to a particular application, use the following pattern for the corresponding element `<name>`:

```
context=APPLICATION, name=appStripeName
```

When the permission applies to all applications, use the following name pattern for the corresponding element `<name>`:

```
context=APPLICATION, name=*
```

When the permission applies to all applications and system policies, use the following name pattern for the corresponding element `<name>`:

```
context=APPLICATION
```

The list of values allowed in the corresponding element `<actions>` are the following (* stands for any allowed action):

```

*
createPolicy
getConfiguredApplications
getSystemPolicy
getApplicationPolicy
createApplicationPolicy
deleteApplicationPolicy
grant
revoke

```

```

createAppRole
alterAppRole
removeAppRole
addPrincipalToAppRole
removePrincipalFromAppRole
hasPermission
containsAppRole

```

21.4.6.2 Credential Store Permission

Class name:

```
oracle.security.jps.service.credstore.CredentialAccessPermission
```

When the permission applies to a particular map and a particular key in that map, use the following pattern for the corresponding element <name>:

```
context=SYSTEM, mapName=myMap, keyName=myKey
```

When the permission applies to a particular map and all keys in that map, use the following pattern for the corresponding element <name>:

```
context=SYSTEM, mapName=myMap, keyName=*
```

The list of values allowed in the corresponding element <actions> are the following (* stands for any allowed action):

```

*
read
write
update
delete

```

21.4.6.3 Generic Permission

Class name:

```
oracle.security.jps.JpsPermission
```

When the permission applies to an assertion performed by a callback instance of `oracle.security.jps.callback.IdentityCallback`, use the following pattern for the corresponding element <name>:

```
IdentityAssertion
```

The only value allowed in the corresponding element <actions> is the following:

```
execute
```

21.4.7 Specifying Bootstrap Credentials Manually

This topic is for an administrator who is not using Oracle Fusion Middleware Control to perform reassociation to an LDAP-based store.

The credentials needed for an administrator to connect to and access an LDAP directory must be specified in a separate file named `cwallet.sso` (bootstrap credentials) and configured in the file `jps-config.xml`. These credentials are stored after the LDAP reassociation process. Bootstrap credentials are always file-based.

Every instance of an LDAP-based policy or credential store must specify bootstrap credentials in a <jpsContext> element that must be named `bootstrap_credstore_context`, as illustrated in the following excerpt:

```

<serviceInstances>
  ...
  <serviceInstance location="./bootstrap" provider="credstoressp"
name="bootstrap.cred">
    <property value="./bootstrap" name="location"/>
  </serviceInstance>
  ...
</serviceInstances>

<jpsContext name="bootstrap_credstore_context">
  <serviceInstanceRef ref="bootstrap.cred"/>
</jpsContext>

```

In the example above, the bootstrap credential `cwallet.sso` is assumed located in the directory `bootstrap`.

An LDAP-based policy or credential store instance references its credentials using the properties `bootstrap.security.principal.key` and `bootstrap.security.principal.map`, as illustrated in the following instance of an LDAP-based policy store:

```

<serviceInstance provider="ldap.policystore.provider" name="policystore.ldap">
  ...
  <property value="bootstrapKey" name="bootstrap.security.principal.key"/>
  ...
</serviceInstance>

```

If the property `bootstrap.security.principal.map` is not specified in the service instance, its value defaults to `BOOTSTRAP_JPS`.

To modify or add bootstrap credentials with OPSS scripts, see [Section 10.5.5, "modifyBootstrapCredential,"](#) and [Section 10.5.6, "addBootstrapCredential."](#)

21.4.8 Migrating Identities with `migrateSecurityStore`

Identity data can be migrated manually from a source repository to a target LDAP repository using the OPSS script `migrateSecurityStore`. The script produces an LDIF file that (after minor manual editing) can be imported into an LDAP-based identity store and can be used with any source 10g or 11g file-based identity store.

For example, this script can be used to convert user and role information in a 10.1.x `jazn-data.xml` file to user and role information in WebLogic LDIF format; the LDIF output file can then be imported into the WebLogic embedded LDAP identity store after changing the password for each user (see note at the end of this section).

This script is **offline**, that is, it does not require a connection to a running server to operate; therefore, the configuration file passed to the argument `configFile` need not be an actual domain configuration file, but it can be assembled *just* to specify the source and destination repositories of the migration.

This script can be run in interactive mode or in script mode, on WebLogic Server, and in interactive mode only, on WebSphere. In interactive mode, you enter the script at a command-line prompt and view the response immediately after. In script mode, you write scripts in a text file (with a `py` file name extension) and run it without requiring input, much like the directives in a shell script.

For platform-specific requirements to run an OPSS script, see [Important Note](#).

Script and Interactive Modes Syntaxes

To migrate identities on WebLogic, use the script (first) or interactive (second) syntaxes (arguments are written in separate lines for clarity):

```
migrateSecurityStore -type idStore
                    -configFile jpsConfigFileLocation
                    -src srcJpsContext
                    -dst dstJpsContext
                    [-dstLdifFile LdifFileLocation]
```

```
migrateSecurityStore(type="idStore", configFile="jpsConfigFileLocation",
src="srcJpsContext", dst="dstJpsContext", [dstLdifFile="LdifFileLocation"])
```

For details about running OPSS scripts on WebSphere Application Server, see

The meaning of the arguments (all required except `dstLdifFile`) is as follows:

- `configFile` specifies the location of a configuration file `jps-config.xml` relative to the directory where the script is run.
- `src` specifies the name of a jps-context in the configuration file passed to the argument `configFile`, where the source store is specified.
- `dst` specifies the name of another jps-context in the configuration file passed to the argument `configFile`, where the destination store is specified. The destination store must be an LDAP-based identity store. For list of supported types, see [Section 3.1.1, "Supported LDAP Identity Store Types."](#)
- `dstLdifFile` specifies the relative or absolute path to the LDIF file created. Applies only when the destination is an LDAP-based Oracle Internet Directory store, such as the embedded LDAP. Notice that the LDIF file is not imported into the LDAP server and, typically, requires manual editing.

The contexts passed to `src` and `dst` must be defined in the passed configuration file and must have distinct names. From these two contexts, the script determines the locations of the source and the target repositories involved in the migration.

Important: The password of every user in the output LDIF file is not the real user password, but the fake string *weblogic*. In case the destination is an LDAP-based Oracle Internet Directory store, the fake string is *change*.

Therefore, before importing the LDIF file into the target LDAP store, the security administrator would typically edit this file and change the fake passwords for real ones.

21.4.9 Example of Configuration File `jps-config.xml`

The following sample shows a complete `jps-config.xml` file that illustrates the configuration of several services and properties; they apply to both Java EE and Java SE applications.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<jpsConfig xmlns="http://xmlns.oracle.com/oracleas/schema/11/jps-config-11_1.xsd">
  <property value="off" name="oracle.security.jps.jaas.mode"/>
  <propertySets>
    <propertySet name="saml.trusted.issuers.1">
      <property value="www.oracle.com" name="name"/>
    </propertySet>
  </propertySets>
```

```

    <serviceProviders>
      <serviceProvider
class="oracle.security.jps.internal.credstore.ssp.SspCredentialStoreProvider"
name="credstoressp" type="CREDENTIAL_STORE">
        <description>SecretStore-based CSF Provider</description>
      </serviceProvider>
      <serviceProvider
class="oracle.security.jps.internal.idstore.ldap.LdapIdentityStoreProvider"
name="idstore.ldap.provider" type="IDENTITY_STORE">
        <description>LDAP-based IdentityStore Provider</description>
      </serviceProvider>
      <serviceProvider
class="oracle.security.jps.internal.idstore.xml.XmlIdentityStoreProvider"
name="idstore.xml.provider" type="IDENTITY_STORE">
        <description>XML-based IdentityStore Provider</description>
      </serviceProvider>
      <serviceProvider
class="oracle.security.jps.internal.policystore.xml.XmlPolicyStoreProvider"
name="policystore.xml.provider" type="POLICY_STORE">
        <description>XML-based PolicyStore Provider</description>
      </serviceProvider>
      <serviceProvider
class="oracle.security.jps.internal.login.jaas.JaasLoginServiceProvider"
name="jaas.login.provider" type="LOGIN">
        <description>JaasLoginServiceProvider to conf loginMod servInsts</description>
      </serviceProvider>
      <serviceProvider class="oracle.security.jps.internal.keystore.KeyStoreProvider"
name="keystore.provider" type="KEY_STORE">
        <description>PKI Based Keystore Provider</description>
        <property value="owsm" name="provider.property.name"/>
      </serviceProvider>
      <serviceProvider class="oracle.security.jps.internal.audit.AuditProvider"
name="audit.provider" type="AUDIT">
        <description>Audit Service</description>
      </serviceProvider>
      <serviceProvider
class="oracle.security.jps.internal.credstore.ldap.LdapCredentialStoreProvider"
name="ldap.credentialstore.provider" type="CREDENTIAL_STORE"/>
      <serviceProvider
class="oracle.security.jps.internal.policystore.ldap.LdapPolicyStoreProvider"
name="ldap.policystore.provider" type="POLICY_STORE">
        <property value="OID" name="policystore.type"/>
      </serviceProvider>
    </serviceProviders>

    <serviceInstances>
      <serviceInstance location="." provider="credstoressp" name="credstore">
        <description>File Based Credential Store Service Instance</description>
      </serviceInstance>
      <serviceInstance provider="idstore.ldap.provider" name="idstore.ldap">
        <property
value="oracle.security.jps.wls.internal.idstore.WlsLdapIdStoreConfigProvider"
name="idstore.config.provider"/>
      </serviceInstance>
      <serviceInstance location="./system-jazn-data.xml"
provider="idstore.xml.provider" name="idstore.xml">
        <description>File Based Identity Store Service Instance</description>
        <property value="jazn.com" name="subscriber.name"/>
      </serviceInstance>

```

```

<serviceInstance location="./system-jazn-data.xml"
provider="policystore.xml.provider" name="policystore.xml">
  <description>File Based Policy Store Service Instance</description>
</serviceInstance>
<serviceInstance location="./default-keystore.jks" provider="keystore.provider"
name="keystore">
  <description>Default JPS Keystore Service</description>
  <property value="JKS" name="keystore.type"/>
  <property value="oracle.wsm.security" name="keystore.csf.map"/>
  <property value="keystore-csf-key" name="keystore.pass.csf.key"/>
  <property value="enc-csf-key" name="keystore.sig.csf.key"/>
  <property value="enc-csf-key" name="keystore.enc.csf.key"/>
</serviceInstance>
<serviceInstance provider="audit.provider" name="audit">
  <property value="None" name="audit.filterPreset"/>
  <property value="0" name="audit.maxDirSize"/>
  <property value="104857600" name="audit.maxFileSize"/>
  <property value="jdbc/AuditDB" name="audit.loader.jndi"/>
  <property value="15" name="audit.loader.interval"/>
  <property value="File" name="audit.loader.repositoryType"/>
</serviceInstance>
<serviceInstance provider="jaas.login.provider" name="saml.loginmodule">
  <description>SAML Login Module</description>
  <property
value="oracle.security.jps.internal.jaas.module.saml.JpsSAMLLoginModule"
name="loginModuleClassName"/>
  <property value="REQUIRED" name="jaas.login.controlFlag"/>
  <propertySetRef ref="saml.trusted.issuers.1"/>
</serviceInstance>
<serviceInstance provider="jaas.login.provider" name="krb5.loginmodule">
  <description>Kerberos Login Module</description>
  <property value="com.sun.security.auth.module.Krb5LoginModule"
name="loginModuleClassName"/>
  <property value="REQUIRED" name="jaas.login.controlFlag"/>
  <property value="true" name="storeKey"/>
  <property value="true" name="useKeyTab"/>
  <property value="true" name="doNotPrompt"/>
  <property value="./krb5.keytab" name="keyTab"/>
  <property value="HOST/localhost@EXAMPLE.COM" name="principal"/>
</serviceInstance>
<serviceInstance provider="jaas.login.provider"
name="digest.authenticator.loginmodule">
  <description>Digest Authenticator Login Module</description>
  <property
value="oracle.security.jps.internal.jaas.module.digest.DigestLoginModule"
name="loginModuleClassName"/>
  <property value="REQUIRED" name="jaas.login.controlFlag"/>
</serviceInstance>
<serviceInstance provider="jaas.login.provider"
name="certificate.authenticator.loginmodule">
  <description>X509 Certificate Login Module</description>
  <property value="oracle.security.jps.internal.jaas.module.x509.X509LoginModule"
name="loginModuleClassName"/>
  <property value="REQUIRED" name="jaas.login.controlFlag"/>
</serviceInstance>
<serviceInstance provider="jaas.login.provider" name="wss.digest.loginmodule">
  <description>WSS Digest Login Module</description>
  <property
value="oracle.security.jps.internal.jaas.module.digest.WSSDigestLoginModule"
name="loginModuleClassName"/>

```

```

    <property value="REQUIRED" name="jaas.login.controlFlag"/>
  </serviceInstance>
  <serviceInstance provider="jaas.login.provider"
name="user.authentication.loginmodule">
    <description>User Authentication Login Module</description>
    <property
value="oracle.security.jps.internal.jaas.module.authentication.JpsUserAuthenticati
onLoginModule" name="loginModuleClassName"/>
    <property value="REQUIRED" name="jaas.login.controlFlag"/>
  </serviceInstance>
  <serviceInstance provider="jaas.login.provider"
name="user.assertion.loginmodule">
    <description>User Assertion Login Module</description>
    <property
value="oracle.security.jps.internal.jaas.module.assertion.JpsUserAssertionLoginMod
ule" name="loginModuleClassName"/>
    <property value="REQUIRED" name="jaas.login.controlFlag"/>
  </serviceInstance>
  <serviceInstance provider="ldap.credentialstore.provider" name="credstore.ldap">
    <property value="bootstrap" name="bootstrap.security.principal.key"/>
    <property value="cn=wls-jrfServer" name="oracle.security.jps.farm.name"/>
    <property value="cn=jpsTestNode" name="oracle.security.jps.ldap.root.name"/>
    <property value="ldap://stadw12.us.oracle.com:3060" name="ldap.url"/>
  </serviceInstance>
  <serviceInstance location="./bootstrap" provider="credstoressp"
name="bootstrap.cred">
    <property value="./bootstrap" name="location"/>
  </serviceInstance>
  <serviceInstance provider="ldap.policystore.provider" name="policystore.ldap">
    <property value="OID" name="policystore.type"/>
    <property value="bootstrap" name="bootstrap.security.principal.key"/>
    <property value="cn=wls-jrfServer" name="oracle.security.jps.farm.name"/>
    <property value="cn=jpsTestNode" name="oracle.security.jps.ldap.root.name"/>
    <property value="ldap://stadw12.us.oracle.com:3060" name="ldap.url"/>
  </serviceInstance>
</serviceInstances>

<jpsContexts default="default">
  <jpsContext name="default">
    <serviceInstanceRef ref="keystore"/>
    <serviceInstanceRef ref="audit"/>
    <serviceInstanceRef ref="credstore.ldap"/>
    <serviceInstanceRef ref="policystore.ldap"/>
  </jpsContext>
  <jpsContext name="oracle.security.jps.fmw.authenticator.DigestAuthenticator">
    <serviceInstanceRef ref="digest.authenticator.loginmodule"/>
  </jpsContext>
  <jpsContext name="X509CertificateAuthentication">
    <serviceInstanceRef ref="certificate.authenticator.loginmodule"/>
  </jpsContext>
  <jpsContext name="SAML">
    <serviceInstanceRef ref="saml.loginmodule"/>
  </jpsContext>
  <jpsContext name="bootstrap_credstore_context">
    <serviceInstanceRef ref="bootstrap.cred"/>
  </jpsContext>
</jpsContexts>
</jpsConfig>

```

Authentication for Java SE Applications

The information in this chapter applies only to Java SE applications, and the audience are developers of Java SE applications. For details about authentication for Java EE applications, see any of the documents listed in [Links to Authentication Topics for Java EE Applications](#).

This chapter includes in the following topics:

- [Authentication for Java SE Applications](#)
- [Configuration Examples](#)

22.1 Links to Authentication Topics for Java EE Applications

The following documents are a good source of information for developing authentication in Java EE applications:

- For general information about authentication in the Oracle WebLogic Server, see section Authentication in chapter 3 in *Oracle Fusion Middleware Understanding Security for Oracle WebLogic Server*.
- *Oracle Fusion Middleware Programming Security for Oracle WebLogic Server*
 - Chapter 3, Securing Web Applications
 - Chapter 4, Using JAAS Authentication in Java Clients
 - Chapter 5, Using SSL Authentication in Java Clients
- *Oracle Fusion Middleware Developing Security Providers for Oracle WebLogic Server*
 - Chapter 4, Authentication Providers
 - Chapter 5, Identity Assertion Providers
 - Chapter 13, Servlet Authentication Filters
- Custom modules in Java EE applications required to be wrapped in an authenticator provider. For details, see section How to Develop a Custom Authentication Provider in *Oracle Fusion Middleware Developing Security Providers for Oracle WebLogic Server*.
- For login modules used in Java EE applications, see the following documentation:
 - Section Login Modules in chapter 4 in *Oracle Fusion Middleware Developing Security Providers for Oracle WebLogic Server*.
 - Section JAAS Authentication Development Environment in Chapter 4 in *Oracle Fusion Middleware Programming Security for Oracle WebLogic Server*
- For links to all OPSS API javadocs, see [Section H.1, "OPSS API References."](#)

22.2 Authentication for Java SE Applications

This section explains the identity store support for Java SE applications, and it includes the following sections:

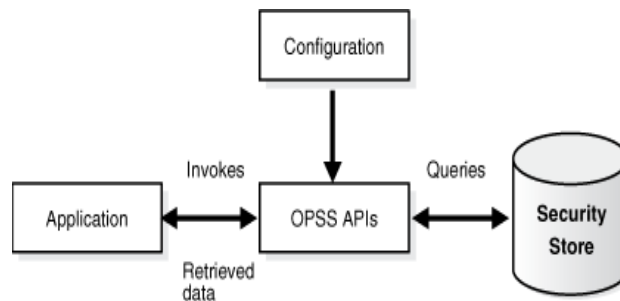
- [The Identity Store](#)
- [Configuring an LDAP Identity Store in Java SE Applications](#)
- [Supported Login Modules for Java SE Applications](#)
- [Using the OPSS API LoginService in Java SE Applications](#)

For details about authorization in Java SE applications, see [Section 23.1, "Configuring Policy and Credential Stores in Java SE Applications."](#)

22.2.1 The Identity Store

Authentication is the mechanism by which callers prove that they are acting on behalf of specific users or system. Using data, such as name-password combinations, authentication answers the question Who are you? The term identity store refers to the storage where identity data is kept, and authentication providers are ways to access an identity store.

An application obtains information from an OPSS security store (identity, policy, or credential store) and manages its contents using the OPSS APIs, as illustrated in the following graphic:



22.2.2 Configuring an LDAP Identity Store in Java SE Applications

A Java SE application can use an LDAP-based identity store configured in the file `jps-config-jse.xml` with the elements `<serviceProvider>`, `<serviceInstance>`, and `<jpsContext>`, as illustrated in the following snippet:

```

<serviceProviders>
  <serviceProvider type="IDENTITY_STORE" name="idstore.ldap.provider"
class="oracle.security.jps.internal.idstore.ldap.LdapIdentityStoreProvider">
    <description>Prototype LDAP-based ID store</description>
  </serviceProvider>
</serviceProviders>

<serviceInstances>
  <serviceInstance name="idstore.ldap" provider="idstore.ldap.provider">
    <property name="idstore.type" value="OID"/>
    <property name="security.principal.alias" value="MyCredentialMapName"/>
    <property name="security.principal.key" value="MyCredentialMapKey"/>
    <property name="ldap.url" value="{LDAP_URI}"/>
    <property name="max.search.filter.length" value="500"/>
    <extendedProperty>
  
```

```

    <name>user.search.bases</name>
    <values>
      <value>cn=users,dc=us,dc=oracle,dc=com</value>
    </values>
  </extendedProperty>
  <extendedProperty>
    <name>group.search.bases</name>
    <values>
      <value>cn=groups,dc=us,dc=oracle,dc=com</value>
    </values>
  </extendedProperty>
</serviceInstance>
</serviceInstances>

<jpsContexts default="ldap_idstore">
  <jpsContext name="ldap_idstore">
    <serviceInstanceRef ref="idstore.ldap"/>
  </jpsContext>

  <jpsContext name="bootstrap_credstore_context">
    <serviceInstanceRef ref="bootstrap.cred"/>
  </jpsContext>
</jpsContexts>

```

Note the following points:

- The name of the `<serviceInstance>` (`idstore.ldap` in the example above) can have any value, but it must match the instance referenced in element `<serviceInstanceRef>`.
- The name of the `<serviceProvider>` (`idstore.ldap.provider` in the example above) can have any value, but it must match the provider in element `<serviceInstance>`.
- To add properties to a provider instance with a prescribed script, see [Appendix E.1, "Configuring OPSS Service Provider Instances with a WLST Script."](#)
- The credentials to access the identity LDAP store are specified with the instance properties `security.principal.key` and `security.principal.alias` and stored in the bootstrap credential store.

22.2.3 Supported Login Modules for Java SE Applications

A login module is a component that authenticates users and populates a subject with principals. This process occurs in two distinct phases: during the first phase, the login module attempts to authenticate a user requesting, as necessary, a name and a password or some other credential data; only if this phase succeeds, the second phase is invoked. During the second phase, the login module assigns relevant principals to a subject, which is eventually used to perform some privileged action.

22.2.3.1 The Identity Store Login Module

A Java SE application can use a stack of login modules to authenticate its users; each module in the stack performs its own computations independently from the others in the stack. These and other services are specified in the file `jps-config-jse.xml`.

OPSS APIs includes the interface

`oracle.security.jps.service.login.LoginService` which allows a Java SE application to invoke not just all login modules in a stack, but a subset of them in a prescribed order.

The name of the jps context (defined in the configuration file `jps-config-jse.xml`) passed to the method `LoginContext` in the `LoginService` interface (which is) determines the stack of login modules that an application uses.

The standard JAAS API `LoginContext` can also be used to invoke the login modules defined in the default context.

The sequence in which a jps context lists the login modules in a stack is significant, since the authentication algorithm takes this order into account in addition to other data, such as the flag that identifies the module security level (required, sufficient, requisite, or optional).

Out-of-the-box, the identity store service is file-based, its contents being provisioned the file `system-jazn-data.xml`, but it can be reconfigured to be an LDAP-based identity store.

OPSS supports the Identity Store login module in Java SE applications, which can be used for authentication or identity assertion.

Identity Store Login Module

The class associated with this login module is the following:

```
oracle.security.jps.internal.jaas.module.idstore.IdStoreLoginModule
```

An instance of this module is configured in the file `jps-config-jse.xml` as illustrated in the following fragment:

```
<serviceInstance name="idstore.loginmodule" provider="jaas.login.provider">
  <description>Identity Store Login Module</description>
  <property name="loginModuleClassName"
value="oracle.security.jps.internal.jaas.module.idstore.IdStoreLoginModule"/>
  <property name="jaas.login.controlFlag" value="REQUIRED"/>
</serviceInstance>
```

Properties specific to this login module include the following:

```
remove.anonymous.role (defaults to true)
add.application.role (defaults to true)
```

22.2.3.2 Using the Identity Store Login Module for Authentication

This section illustrates the use of the Identity Store login module for basic username and password authentication.

Invoke `IdStoreLoginModule`

The following code fragment illustrates how to set a callback handler and a context:

```
import javax.security.auth.Subject;
import javax.security.auth.login.LoginContext;

Subject sub = new Subject();
CallbackHandler cbh = new YourCallbackHandler();
LoginContext context = new LoginContext(appName, subject, cbh);
context.login();
```

The callback handler must be able to handle `NameCallback` and `PasswordCallback`.

Configure jps-config-jse.xml

The following `jps-config-jse.xml` fragment illustrates the configuration of the context `appName`:

```
<jpsContext name="appName">
  <serviceInstanceRef ref="jaaslm.idstore1"/>
</jpsContext>

<serviceProvider type="JAAS_LM" name="jaaslm.idstore"
  class="oracle.security.jps.internal.jaas.module.idstore.IdStoreLoginModule">
  <description>Identity Store-based LoginModule
  </description>
</serviceProvider>

<serviceInstance name="jaaslm.idstore1" provider="jaaslm.idstore">
  <property name="jaas.login.controlFlag" value="REQUIRED"/>
  <property name="debug" value="true"/>
  <property name="addAllRoles" value="true"/>
</serviceInstance>
```

Write the Callback Handler

The following code snippet illustrates a callback handler able to handle name and password callback:

```
import javax.security.auth.callback.*;
import java.io.IOException;
public class SampleCallbackHandler implements CallbackHandler {
  //For name/password callbacks
  private String name = null;private char[] password = null;
  public SampleCallbackHandler(String name, char[] pwd) {
    if (name == null || name.length() == 0 )
      throw new IllegalArgumentException("Invalid name ");
    else
      this.name = name;
    if (pwd == null || pwd.length == 0)
      throw new IllegalArgumentException("Invalid password ");
    else
      this.password = pwd;
  }
  public String getName() {
    return name;
  } public char[] getPassword() {
    return password;
  }
  public void handle(Callback[] callbacks)
    throws IOException, UnsupportedCallbackException {
    if (callbacks != null && callbacks.length > 0) {
      for (Callback c : callbacks) {
        if (c instanceof NameCallback) {
          ((NameCallback) c).setName(name);
        }
        else
          if (c instanceof PasswordCallback) {
            ((PasswordCallback) c).setPassword(password);
          }
        else {
          throw new UnsupportedCallbackException(c);
        }
      }
    }
  }
}
```

```
}  
}
```

22.2.3.3 Using the Identity Login Module for Assertion

To use the Identity Store login module for assertion, a developer must:

- Provide the appropriate permission for the caller to execute the protected method `setIdentity`. This requires granting the permission `oracle.security.jps.JpsPermission` with the name `IdentityAssertion`.
- Implement a callback handler that uses the class `oracle.security.jps.callback.IdentityCallback` as shown in the code sample below.

The above two requirements are illustrated in the following configuration and code samples.

Provisioning the JpsPermission

The following configuration sample illustrates a grant allowing the code `MyApp` the required `JpsPermission` to execute protected methods in the assertion login module:

```
<grant>  
  <grantee>  
    <codesource>  
      <url>file:${soa.oracle.home}/application/myApp.ear</url>  
      <!--! soa.oracle.home is a system property set when  
           the server JVM is started -->  
    </codesource>  
  </grantee>  
  <permissions>  
    <permission>  
      <class>oracle.security.jps.JpsPermission</class>  
      <name>IdentityAssertion</name>  
    </permission>  
  </permissions>  
</grant>
```

The following configuration sample illustrates a grant allowing the principal `jdoe` the required `JpsPermission` to execute the assertion login module:

```
<grant>  
  <grantee>  
    <principals>  
      <principal>  
        <class>weblogic.security.principal.WLSUserImpl</class>  
        <name>jdoe</name>  
      </principal>  
    </principals>  
  </grantee>  
  <permissions>  
    <permission>  
      <class>oracle.security.jps.JpsPermission</class>  
      <name>IdentityAssertion</name>  
    </permission>  
  </permissions>  
</grant>
```

Implementing the CallbackHandler

The following code fragment illustrates an implementation of the callback handler:

```
import javax.security.auth.callback.Callback;
import javax.security.auth.callback.CallbackHandler;
import javax.security.auth.callback.NameCallback;
import javax.security.auth.callback.PasswordCallback;
import javax.security.auth.callback.UnsupportedCallbackException;

import oracle.security.jps.callback.IdentityCallback;

public class CustomCallbackHandler implements CallbackHandler {
    private String name = null;
    private char[] password;

    public CustomCallbackHandler(String name) {
        this.name = name;
    }

    public CustomCallbackHandler(String name, char[] password) {
        this.name = name;
        this.password = password;
    }

    public void handle(Callback[] callbacks) throws IOException,
        UnsupportedCallbackException {
        for (Callback callback : callbacks) {
            if (callback instanceof NameCallback) {
                NameCallback nc = (NameCallback) callback;
                nc.setName(name);
            }
            else if (callback instanceof PasswordCallback) {
                PasswordCallback pc = (PasswordCallback) callback;
                pc.setPassword(password);
            }
            else if (callback instanceof IdentityCallback) {
                IdentityCallback idcb = (IdentityCallback) callback;
                idcb.setIdentity(name);
                idcb.setIdentityAsserted(true);
                idcb.setAuthenticationType("CUSTOM");
            } else {
                //throw exception
                throw new UnsupportedCallbackException(callback);
            }
        }
    }
}
```

The following code fragment illustrates the implementation of a login module:

```
import javax.security.auth.callback.CallbackHandler;
import javax.security.auth.login.LoginContext;

import oracle.security.jps.service.JpsServiceLocator;
import oracle.security.jps.service.login.LoginService;

public class LoginModuleExample {
    private static final String CONTEXT_NAME = "JSE_UserAuthnAssertion";

    public LoginModuleExample() {
        super();
    }
}
```

```
    }

    public Subject assertUser(final String username) throws Exception {
        CallbackHandler cbh =
            AccessController.doPrivileged(new
PrivilegedExceptionAction<CallbackHandler>() {
            public CallbackHandler run() throws Exception {
                return new CustomCallbackHandler(username);
            }
        });
        Subject sub = new Subject();
        LoginService ls =
            JpsServiceLocator.getServiceLocator().lookup(LoginService.class);
        LoginContext context = ls.getLoginContext(sub, cbh);

        context.login();
        Subject s = context.getSubject();

        return s;
    }

    public Subject authenticate(final String username, final char[] password)
throws Exception {
        CallbackHandler cbh = new CustomCallbackHandler(username, password);
        Subject sub = new Subject();
        LoginService ls =
            JpsServiceLocator.getServiceLocator().lookup(LoginService.class);
        LoginContext context = ls.getLoginContext(sub, cbh);

        context.login();
        Subject s = context.getSubject();

        return s;
    }

    public static void main(String[] args) {
        LoginModuleExample loginModuleExample = new LoginModuleExample();
        try {
            System.out.println("authenticated user subject = " +
                loginModuleExample.authenticate("testUser",
"welcome1".toCharArray()));
            System.out.println("asserted user subject = " +
                loginModuleExample.assertUser("testUser"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

22.2.4 Using the OPSS API LoginService in Java SE Applications

To invoke a login module programmatically in Java SE applications, use the method `getLoginContext` of the interface `oracle.security.jps.service.login.LoginService`.

Similar to the method `LoginContext` in the standard JAAS API, `getLoginContext` returns an instance of a `LoginContext` object that can be used to authenticate a user, but, more generally, it also allows the use of any number of login modules in any order. Authentication is then performed on just those login modules and in the order they were passed.

The following code fragment illustrates user authentication against a subset of login modules in a prescribed order using `getLoginContext`:

```
import oracle.security.jps.service.ServiceLocator;
import oracle.security.jps.service.JpsServiceLocator;
import oracle.security.jps.service.login.LoginService;

//Obtain the login service
ServiceLocator locator = JpsServiceLocator.getServiceLocator();
LoginService loginService = locator.lookup(LoginService.class);

//Create the handler for given name and password
CallbackHandler cbh = new MyCallbackHandler("name", "password".toCharArray());

//Invoke login modules selectively in a given order
selectiveModules = new Sting[]{"lmName1", "lmName2", "lmName3"};
LoginContext ctx = loginService.getLoginContext(new Subject(), cbh,
selectiveModules);
ctx.login();
Subject s = ctx.getSubject();
```

`selectiveModules` is an array of (login module) names, and the authentication uses precisely those login modules named in the array in the order listed in the array. Each name in the array must be the name of a service instance listed in the default context of the file `jps-config-jse.xml`.

The following fragment illustrates the configuration of a stack of two login modules:

```
<serviceProvider type="LOGIN" name="jaas.login.provider"
class="oracle.security.jps.internal.login.jaas.JaasLoginServiceProvider">
  <description>Common definition for any login module instances</description>
</serviceProvider>

<serviceInstance name="auth.loginmodule" provider="jaas.login.provider">
  <description>User Authentication Login Module</description>
  <property name="loginModuleClassName"
value="oracle.security.jps.internal.jaas.module.authentication.JpsUserAuthenticati
onLoginModule"/>
  <property name="jaas.login.controlFlag" value="REQUIRED"/>
</serviceInstance>

<serviceInstance name="custom.loginmodule" provider="jaas.login.provider">
  <description>My Custom Login Module</description>
  <property name="loginModuleClassName" value="my.custom.MyLoginModuleClass"/>
  <property name="jaas.login.controlFlag" value="REQUIRED"/>
</serviceInstance>

<jpsContexts default="aJpsContext">
  <jpsContext name="aJpsContext">
    <serviceInstanceRef ref="auth.loginmodule"/>
    <serviceInstanceRef ref="custom.loginmodule"/>
  </jpsContext>
</jpsContexts>
```

22.3 Configuration Examples

This section illustrates the configuration of the following artifacts:

- XML policy and credential stores
- XML and LDAP identity stores

- Login Module Principals

XML Policy and Credential Stores Configuration

The following snippets illustrate the configuration of XML-based policy and credential stores. The contents of an XML-based policy store is specified in the file `system-jazn-data.xml`; the contents of an XML-based credential store is specified in the file `cwallet.sso`.

```
<serviceProviders>
  <serviceProvider
class="oracle.security.jps.internal.policystore.xml.XmlPolicyStoreProvider"
name="policystore.xml.provider" type="POLICY_STORE">
  <description>XML-based PolicyStore Provider</description>
</serviceProvider>

  <serviceProvider
class="oracle.security.jps.internal.credstore.ssp.SspCredentialStoreProvider"
name="credstoressp" type="CREDENTIAL_STORE">
  <description>SecretStore-based CSF Provider</description>
</serviceProvider>
</serviceProviders>

<serviceInstances>
  <serviceInstance location="." provider="credstoressp" name="credstore">
  <description>File-based Credential Store Service Instance</description>
</serviceInstance>

  <serviceInstance location="./system-jazn-data.xml"
provider="policystore.xml.provider" name="policystore.xml">
  <description>File-based Policy Store Service Instance</description>
</serviceInstance>
</serviceInstances>
```

XML Identity Store Configuration

The following snippets illustrate the configuration of an XML-based identity store. The contents of an XML-based identity store is specified in the file `system-jazn-data.xml`.

```
<serviceProvider
  class="oracle.security.jps.internal.idstore.xml.XmlIdentityStoreProvider"
  name="idstore.xml.provider"
  type="IDENTITY_STORE">
  <description>XML-based Identity Store Service Provider</description>
</serviceProvider>

<serviceInstance
  location="./system-jazn-data.xml" provider="idstore.xml.provider"
  name="idstore.xml">
  <description>File Based Identity Store Service Instance</description>
  <property value="jazn.com" name="subscriber.name"/>
</serviceInstance>
```

LDAP Identity Store Configuration

The snippets below illustrate the configuration of an LDAP-based identity store, which includes the required configuration of the bootstrap credentials to access the LDAP server. The service instance property `idstore.type` can have the following values, according to the LDAP used:

Table 22–1 Idstore Types

Supported LDAP	Idstore.type value
Oracle Internet Directory 10g and 11g	OID
Oracle Virtual Directory 10g and 11g	OVD
Sun Java System Directory Server 6.3	IPLANET
Active Directory 2003, 2008	ACTIVE_DIRECTORY
Novell eDirectory 8.8	EDIRECTORY
Oracle Directory Server Enterprise Edition 11gR1 (11.1.1.3+)	IPLANET
IBM Tivoli DS 6.2	OPEN_LDAP
OpenLDAP 2.2.	OPEN_LDAP

```

<serviceProvider
  class="oracle.security.jps.internal.idstore.ldap.LdapIdentityStoreProvider"
  name="idstore.ldap.provider" type="IDENTITY_STORE">
  <description>LDAP-based Identity Store Service Provider</description>
</serviceProvider>

<serviceProvider
  class="oracle.security.jps.internal.credstore.ssp.SspCredentialStoreProvider"
  name="credstoressp" type="CREDENTIAL_STORE">
  <description>SecretStore-based CSF Provider</description>
</serviceProvider>

<serviceInstance name="idstore.oid" provider="idstore.ldap.provider">
  <property name="subscriber.name" value="dc=us,dc=oracle,dc=com" />
  <property name="idstore.type" value="OID" />
  <property value=ldap://myOID.com:3555 name="ldap.url" />
  <extendedProperty>
    <name>user.search.bases</name>
    <values>
      <value>cn=users,dc=us,dc=oracle,dc=com</value>
    </values>
  </extendedProperty>
  <extendedProperty>
    <name>group.search.bases</name>
    <values>
      <value>cn=groups,dc=us,dc=oracle,dc=com</value>
    </values>
  </extendedProperty>
  <property name="username.attr" value="uid" />
  <property name="group.attr" value="cn" />
</serviceInstance>

<serviceInstance location="./bootstrap" provider="credstoressp"
  name="bootstrap.cred">
  <property value="./bootstrap" name="location" />
</serviceInstance>

```

Login Module Principals

The following properties are set in the out-of-the-box `jps-config-jse.xml`:

```

<property name="oracle.security.jps.enterprise.user.class"
  value="weblogic.security.principal.WLSUserImpl" />

```

```
<property name="oracle.security.jps.enterprise.role.class"  
          value="weblogic.security.principal.WLSGroupImpl"/>
```

The above properties must be used in any login module; this implies that the principals that represent users and groups in the identity store are the following:

```
weblogic.security.principal.WLSUserImpl  
weblogic.security.principal.WLSGroupImpl
```

Authorization for Java SE Applications

This chapter explains how to develop and configure authorization in Java SE applications and lists some unsupported methods in the following sections:

- [Configuring Policy and Credential Stores in Java SE Applications](#)
- [Unsupported Methods for File-Based Policy Stores](#)

For details about the policy model, see [Section 20.3, "The JAAS/OPSS Authorization Model."](#)

23.1 Configuring Policy and Credential Stores in Java SE Applications

The configuration of policy and credential stores in Java SE applications is explained in the following sections:

- [Configuring File-Based Policy and Credential Stores](#)
- [Configuring LDAP-Based Policy and Credential Stores](#)
- [Configuring DB-Based OPSS Security Stores](#)

For details about configuring authentication for Java SE applications, see [Section 22.2, "Authentication for Java SE Applications."](#)

System properties should be set, as appropriate, for authorization to work in Java SE applications. For a complete list of properties, see [Section F.1, "OPSS System Properties."](#)

A Java SE application can use file-, LDAP-, or DB-based store providers; these services are configured in the application file `jps-config-jse.xml`.

23.1.1 Configuring File-Based Policy and Credential Stores

A file-based policy store is specified in the file `system-jazn-data.xml`; a file-based credential store is specified in the file `cwallet.sso` (this wallet file should not be confused with the bootstrap file, also named `cwallet.sso`, which contains the credentials to access LDAP stores, when the application security is LDAP-based).

For details about wallets, see [Section 21.4.3, "Using a Wallet-Based Credential Store."](#)
For details about modifying or adding bootstrap credentials, see [Section 10.5.5, "modifyBootStrapCredential,"](#) and [Section 10.5.6, "addBootStrapCredential."](#)

The following fragments illustrate the configuration of file-based policy and credential stores, and the `jpsContext` that reference them:

```
<serviceProviders>
  <serviceProvider type="CREDENTIAL_STORE" name="credstoressp"
    class="oracle.security.jps.internal.credstore.ssp.SspCredentialStoreProvider">
```

```

        <description>SecretStore-based CSF Provider</description>
    </serviceProvider>
    <serviceProvider type="POLICY_STORE" name="policystore.xml.provider"
class="oracle.security.jps.internal.policystore.xml.XmlPolicyStoreProvider">
    <description>XML-based PolicyStore Provider</description>
    </serviceProvider>
</serviceProviders>

<serviceInstances>
    <serviceInstance name="credstore" provider="credstoressp" location="."/>
    <description>File-based Credential Store Service Instance</description>
    </serviceInstance>

    <serviceInstance name="policystore.xml" provider="policystore.xml.provider"
location="./system-jazn-data.xml">
    <description>File-based Policy Store Service Instance</description>
    <property name="oracle.security.jps.policy.principal.cache.key" value="false"/>
    </serviceInstance>
</serviceInstances>

<jpsContexts default="TestJSE">
    <jpsContext name="TestJSE">
    <serviceInstanceRef ref="credstore"/>
    <serviceInstanceRef ref="policystore.xml"/>
    ...
    </jpsContext>
    ...
</jpsContexts>

```

Note the required setting of the property `oracle.security.jps.policy.principal.cache.key` to `false` in the policy store instance.

23.1.2 Configuring LDAP-Based Policy and Credential Stores

This section assumes that an LDAP-based store has been set to be used as the policy and credential stores; for details about setting up nodes in an Oracle Internet Directory, see section [Section 8.2.2, "Prerequisites to Using an LDAP-Based Security Store."](#)

The following fragments illustrate the configurations of providers and instances for LDAP-based policy and credential stores for a Java SE application:

```

<serviceProviders
    <serviceProvider type="POLICY_STORE" name="ldap.policystore.provider"
class=oracle.security.jps.internal.policystore.ldap.LdapPolicyStoreProvider"/>

    <serviceProvider type="CREDENTIAL_STORE" name="ldap.credential.provider"
class=oracle.security.jps.internal.credstore.ldap.LdapCredentialStoreProvider"/>
</serviceProviders>

<serviceInstances>
    <serviceInstance provider="ldap.policystore.provider" name="policystore.ldap">
    <property value="OID" name="policystore.type"/>
    <property value="bootstrap" name="bootstrap.security.principal.key"/>
    <property value="cn=PSldomainRC3" name="oracle.security.jps.farm.name"/>
    <property value="cn=myTestNode" name="oracle.security.jps.ldap.root.name"/>
    <property value="ldap://myComp.com:1234" name="ldap.url"/>
    </serviceInstance>

    <serviceInstance provider="ldap.credential.provider" name="credstore.ldap">
    <property value="bootstrap" name="bootstrap.security.principal.key"/>

```

```

<property value="cn=PS1domainRC3" name="oracle.security.jps.farm.name"/>
<property value="cn=myTestNode" name="oracle.security.jps.ldap.root.name"/>
<property value="ldap://myComp.com:1234" name="ldap.url"/>
</serviceInstance>
</serviceInstances>

```

The following fragment illustrates the configuration of the bootstrap credentials file (`cwallet.sso`), which allows the program access to the LDAP server:

```

<serviceInstance location="./bootstrap" provider="credstoressp"
name="bootstrap.cred">
  <property value="./bootstrap" name="location"/>
</serviceInstance>

```

The following fragment illustrates the configuration of the necessary `jpsContexts` that reference the instances above:

```

<jpsContexts default="TestJSE">
  <jpsContext name="TestJSE">
    <serviceInstanceRef ref="policystore.ldap"/>
    <serviceInstanceRef ref="credstore.ldap"/>
  </jpsContext>
  <jpsContext name="bootstrap_credstore_context">
    <serviceInstanceRef ref="bootstrap.cred"/>
  </jpsContext>
</jpsContexts>

```

The following code fragment illustrates how to obtain programmatically a reference to the LDAP-based policy store configured above, and it assumes that the system property `oracle.security.jps.config` has been set to the location of the file `jps-config-jse.xml`:

```

String contextName="TestJSE"; ...
public static PolicyStore getPolicyStore(String contextName) {
    try-block
        JpsContextFactory ctxFact;
        ctxFact = JpsContextFactory.getContextFactory();
        JpsContext ctx = ctxFact.getContext(contextName);
        return ctx.getServiceInstance(PolicyStore.class);
    catch-block
    ...
}

```

23.1.3 Configuring DB-Based OPSS Security Stores

This section assumes that a DB-based store has been set to be used as the OPSS security store. For details about setting up nodes in a DB, see section [Section 8.3.1, "Prerequisites to Using a DB-Based Security Store."](#)

Note the following important points regarding the sample configuration below:

- The value of the configuration property `jdbc.url` should be identical to the name of the JDBC data source entered when the data source was created.
- The values of the bootstrap credentials (map and key) must match those passed to the WLST script `addBootStrapCredential` when the bootstrap credential was created.

The following fragment illustrates configuration of DB-based policy, credential, and key stores in the file `jps-config-jse.xml`:

```

<jpsConfig ...>
  <propertySets>

```

```

<propertySet name="props.db.1">
  <property value="cn=myDomain" name="oracle.security.jps.farm.name"/>
  <property value="DB_ORACLE" name="server.type"/>
  <property value="cn=myRoot" name="oracle.security.jps.ldap.root.name"/>
  <property name="jdbc.url" value="jdbc:oracle:thin:@myhost.com:1521/srv_name"/>
  <property name="jdbc.driver" value="oracle.jdbc.driver.OracleDriver"/>
  <property name="bootstrap.security.principal.key" value="myKeyName" />
  <property name="bootstrap.security.principal.map" value="myMapName" />
</propertySet>
</propertySets>
<serviceProviders>
  <serviceProvider
class="oracle.security.jps.internal.policystore.OPSSPolicyStoreProvider"
  type="POLICY_STORE" name="policy.rdbms">
    <description>DBMS based PolicyStore</description>
  </serviceProvider>

  <serviceProvider
class="oracle.security.jps.internal.credstore.rdbms.DbmsCredentialStoreProvider"
  type="CREDENTIAL_STORE" name="db.credentialstore.provider" >

  <serviceProvider class="oracle.security.jps.internal.keystore.KeyStoreProvider"
  type="KEY_STORE" name="keystore.provider" >
    <property name="provider.property.name" value="owsm"/>
  </serviceProvider>
</serviceProviders>

<serviceInstances>
  <serviceInstance name="policystore.rdbms" provider="db.policystore.provider">
    <propertySetRef ref = "props.db.1"/>
    <property name="policystore.type" value="DB_ORACLE"/>
  </serviceInstance>

  <serviceInstance name="credstore.rdbms" provider="db.credstore.provider">
    <propertySetRef ref = "props.db.1"/>
  </serviceInstance>

  <serviceInstance name="keystore.rdbms" provider="db.keystore.provider">
    <propertySetRef ref = "props.db.1"/>
    <property name="keystore.provider.type" value="db"/>
  </serviceInstance>
</serviceInstances>

<jpsContexts default="default">
  <jpsContext name="default">
    <serviceInstanceRef ref="policystore.rdbms"/>
    <serviceInstanceRef ref="credstore.rdbms"/>
    <serviceInstanceRef ref="keystore.rdbms"/>
  </jpsContext>
</jpsContexts>
</jpsConfig>

```

23.2 Unsupported Methods for File-Based Policy Stores

This release does not support, for file-based policy stores, methods involving the following features:

- Bulk authorization
- Complex queries

- Cascading deletions

Bulk authorization is encapsulated in the following method of the interface `oracle.security.jps.service.policystore`:

```
java.util.Set<ResourceActionsEntry>
checkBulkAuthorization(javax.security.auth.Subject subject,
                       java.util.Set<ResourceActionsEntry> requestedResources)
                       throws PolicyStoreException
```

Complex queries relates to any method that takes a query. When the policy store is file-based, the query must be simple; if such a method is passed a complex query and the policy store is file-based, the method will throw an exception.

A simple query is a query with just one search criterion; a complex query is a query with two or more search criteria; each call to `addQuery` adds a criterion to the query.

The following code fragment that illustrates the building of a simple query that returns of all permissions with a display name matching the string `MyDisplayName`:

```
PermissionSetSearchQuery query = new PermissionSetSearchQuery();
query.addQuery(PermissionSetSearchQuery.SEARCH_PROPERTY.DISPLAY_NAME,
               false,
               ComparatorType.EQUALITY,
               "MyDisplayName",
               BaseSearchQuery.MATCHER.EXACT);
getPermissionSets(query);
```

The following example illustrates the building of a complex query that returns all permission sets with a given resource type and a given resource instance name:

```
PermissionSetSearchQuery query = new PermissionSetSearchQuery();
query.addQuery(PermissionSetSearchQuery.SEARCH_PROPERTY.RESOURCE_TYPE,
               false,
               ComparatorType.EQUALITY,
               "MyResourceType",
               BaseSearchQuery.MATCHER.EXACT);

query.addQuery(PermissionSetSearchQuery.SEARCH_PROPERTY.RESOURCE_NAME,
               false,
               ComparatorType.EQUALITY,
               "MyResourceInstanceName",
               BaseSearchQuery.MATCHER.EXACT);

query.setANDMatch();
getPermissionSets(query);
```

Cascading deletions relates to any method that includes the Boolean argument `cascadeDelete`. The only value allowed for this argument in case the policy store is file-based is `FALSE`. Here is an example of such a method in the interface `ResourceTypeManager`:

```
void deleteResourceType(EntryReference rtRef, boolean cascadeDelete)
                       throws PolicyObjectNotFoundException,
                       PolicyStoreOperationNotAllowedException,
                       PolicyStoreException
```

Developing with the Credential Store Framework

This chapter describes how to work with the Credential Store Framework (CSF) APIs in the following sections:

- [About the Credential Store Framework API](#)
- [Overview of Application Development with CSF](#)
- [Setting the Java Security Policy Permissions](#)
- [Guidelines for the Map Name](#)
- [Configuring the Credential Store](#)
- [Steps for Using the API](#)
- [Examples](#)
- [Best Practices](#)

24.1 About the Credential Store Framework API

A credential store is used for secure storage of credentials. The credential store framework (CSF) API is used to access and perform operations on the credential store.

The Credential Store Framework:

- enables you to manage credentials securely
- provides an API for storage, retrieval, and maintenance of credentials in different back-end repositories
- supports file-based (Oracle wallet) and LDAP-based credential management

Critical (create, update, delete) functions provided by the CSF API include:

- verifying if a credential map, or a credential with a given key, exists in the store
- returning credentials associated with `<mapname, key>`
- assigning credentials to `<mapname, key>`
- deleting credentials associated with a given map name, or a given map name and key
- resetting credentials for a specified `<mapname, key>`

Operations on `CredentialStore` are secured by `CredentialAccessPermission`, which implements the fine-grained access control model utilized by CSF.

See Also:

- [Chapter 10, "Managing the Credential Store"](#)

24.2 Overview of Application Development with CSF

Knowledge of the following areas is helpful in getting your applications to work with the credential store framework:

- Determining appropriate map names and key names to use. This is critical in an environment with multiple applications storing credentials in the common credential store.
- Provisioning Java security policies.

Policy permissions are set in the policy store, which can be file-based (`system-jazn-data.xml`) or LDAP-based. Setting appropriate permissions to enable application usage without compromising the security of your data requires careful consideration of permission settings.

See Also: [Section 9.1, "Managing the Policy Store"](#).

- How to define the credential store instance in `jps-config.xml`.

You will need to define the service instance in `jps-config.xml` only if manually crafting the configuration file.

Note: The file-based provider is already configured by default, and can be changed to an LDAP-based provider. See [Section 8.6, "Migrating the OPSS Security Store"](#).

- Steps to take in setting up the environment.

The steps are different for stand-alone applications and those that operate in an Oracle WebLogic Server environment.

Subsequent sections provide details about each of these tasks.

24.3 Setting the Java Security Policy Permissions

The Oracle Platform Security Services policy provider is set when the server is started. When the provider is file-based, the policy data is stored in `system-jazn-data.xml`.

CSF supports securing credentials:

- at the map level, or
- with finer granularity for specific `<mapname, key>`

Notes:

- To properly access the CSF APIs, you need to grant Java permissions in the policy store.
 - The code invoking CSF APIs needs code source permission. The permissions are typically for specific code jars and not for the complete application.
-
-

24.3.1 Guidelines for Granting Permissions

The Credential Store Framework relies on Java permissions to grant permissions to credential store objects.

It is highly recommended that only the requisite permissions be granted, and no more.

WARNING: It is risky and inadvisable to grant unnecessary permissions, particularly permissions to all maps and/or keys.

24.3.2 Permissions Grant Example 1

Note: In the examples, the application jar file name is `AppName.jar`.

The `CredentialStore` maintains mappings between map names and credential maps. Each map name is mapped to a `CredentialMap`, which is a secure map of keys to `Credential` objects.

This example grants permissions for a specific map name and a specific key name of that map.

```
<jazn-policy>
  <grant>
    <grantee>
      <principals>...</principals>
      <!-- This is the location of the jar -->
      <!-- as loaded with the run-time -->
      <codesource>
        <url>file:${oracle.deployed.app.dir}/<MyApp>${oracle.deployed.app.ext}</url>
      </codesource>
    </grantee>
    <permissions>
      <permission>
        <class>oracle.security.jps.service.credstore.
          CredentialAccessPermission</class>
        <name>context=SYSTEM,mapName=myMap,keyName=myKey</name>
        <!-- All actions are granted -->
        <actions>*</actions>
      </permission>
    </permissions>
  </grant>
</jazn-policy>
```

where:

- `MapName` is the name of the map (typically the name of the application) for which you want to grant these permissions (read, write, update, and delete permissions denoted by the wildcarded actions).
- `KeyName` is the key name in use.

24.3.3 Permissions Grant Example 2

In this example permissions are granted for a specific map name and all its key names.

```
<jazn-policy>
  <grant>
```

```
<grantee>
  <principals>...</principals>
  <codesource>
<url>file:${oracle.deployed.app.dir}/<MyApp>${oracle.deployed.app.ext}</url>
  </codesource>
</grantee>
<permissions>
  <permission>
    <class>oracle.security.jps.service.credstore.
      CredentialAccessPermission</class>
    <name>context=SYSTEM,mapName=myMap,keyName=*</name>
    <!-- Certain actions are explicitly specified -->
    <!-- Compare to wild-card grant in previous example -->
    <actions>read,write,update,delete</actions>
  </permission>
</permissions>
</grant>
</jaza-policy>
```

24.4 Guidelines for the Map Name

When the domain-level credential store is used, name conflicts can arise with the various map names in the store for different applications. To avoid this, each application must have a unique map name in the store.

To achieve this, it is recommended that the map name you use uniquely identify the application.

Within a given map name, an application can store multiple credentials each of which is identifiable by a key. The map name and the key together constitute a primary key within a given credential store.

If there is a requirement that an application use more than one map name, then uniqueness continues to be maintained.

For example, consider three applications:

- a Repository Creation Utility (RCU) based application,
- a Oracle WebCenter application, and
- a Fusion Middleware Control application

For RCU, a map name of RCU is chosen and the keys for three credentials are (say) Key1, Key2, and Key3:

Note: The map names and key names used here are arbitrary and chosen for illustration only. Your application can use altogether different map names and/or keynames.

```
MapName -> RCU, Key -> Key1 and Credential -> PasswordCredential1
MapName -> RCU, Key -> Key2 and Credential -> PasswordCredential2
MapName -> RCU, Key -> Key3 and Credential -> GenericCredential1
```

For Oracle WebCenter, the map name is Web and the key for a single credential is Key1:

```
MapName -> Web, Key -> Key1 and Credential -> PasswordCredential3
```

For Fusion Middleware Control, the map name is denoted by EM and the keys for two credentials are Key1 and Key2 respectively:

```
MapName -> EM, Key -> Key1 and Credential -> PasswordCredential14
MapName -> EM, Key -> Key2 and Credential -> GenericCredential2
```

Note that the map name and key name are just two arbitrary strings and can have any valid string values in practice. However, implementing this way makes map names easier to manage.

24.5 Configuring the Credential Store

The administrator needs to define the credential store instance in a configuration file which contains information about the location of the credential store and the provider classes. Configuration files are located in:

```
DOMAIN_HOME/config/fmwconfig
```

and are named as follows:

- `jps-config.xml` for Oracle WebLogic Server
- `jps-config-jse.xml` for Java SE

For details, see [Chapter 10, "Managing the Credential Store"](#).

24.6 Steps for Using the API

You can use the credential store framework within Oracle WebLogic Server or in a standalone environment.

- [Using the CSF API in a Standalone Environment](#)
- [Using the CSF API in Oracle WebLogic Server](#)

24.6.1 Using the CSF API in a Standalone Environment

The steps for using the API in a standalone environment are:

1. Set up the classpath. Ensure that the `jps-manifest.jar` file is in your classpath. For details, see Required JAR in Classpath in [Section 1.5.3, "Scenario 3: Securing a Java SE Application"](#).
2. Set up the policy; to provide access to the CSF APIs, you need to configure the access permissions in the reference policy store. For examples, see [Section 24.3, "Setting the Java Security Policy Permissions"](#).
3. Run the application.

Command-line options include:

```
-Doracle.security.jps.config
specifies the full path to the configuration file

-Djava.security.policy
specifies the location of the OPSS/Oracle WebLogic Server policy file

-Djava.security.debug=all
is helpful for debugging purposes
```

24.6.2 Using the CSF API in Oracle WebLogic Server

The steps for using the API in an Oracle WebLogic Server environment are:

1. The credential store service provider section of the `jps-config.xml` file is configured out-of-the-box in the following directory:

```
$DOMAIN_HOME/config/fmwconfig
```

If needed, reassociate to an LDAP credential store.

2. Set up the policy; to provide access to the CSF APIs, you need to configure the access permissions in the reference policy store. For examples, see [Section 24.3, "Setting the Java Security Policy Permissions"](#).
3. Start Oracle WebLogic Server.
4. Deploy and test the application.

24.7 Examples

This section provides several examples of using the credential store framework APIs. It shows:

- a "utility" Java program which is called by all examples and performs the actual credential store operations
- the Java SE or Java EE code that calls the utility program,
- the policy store setup
- the configuration file

In each example, the test code is set up to show how the credential store operations are affected by the permissions. For each example the policy file, the test code, and the configuration file are provided to demonstrate how the provider information must be specified, and to enable you to compare the defined permissions on the map/key with the operation attempted in the code.

The section is structured as follows:

- [Code for CSF Operations](#)
- [Example 1: Java SE Application with Wallet Store](#)
- [Example 2: Java EE Application with Wallet Store](#)
- [Example 3: Java EE Application with LDAP Store](#)

24.7.1 Code for CSF Operations

The following common "utility" program performs the CSF API operations. It is called by the example programs.

```
package demo.util;

import java.security.AccessController;
import java.security.PrivilegedAction;

import oracle.security.jps.JpsException;
import oracle.security.jps.service.credstore.Credential;
import oracle.security.jps.service.credstore.CredentialAlreadyExistsException;
import oracle.security.jps.service.credstore.CredentialFactory;
import oracle.security.jps.service.credstore.CredentialStore;
```

```

import oracle.security.jps.service.credstore.PasswordCredential;

public class CsfUtil {
    final CredentialStore store;
    public CsfUtil(CredentialStore store) {
        super();
        this.store = store;
    }

    private void doOperation() {
        try {
            PasswordCredential pc = null;
            try {
                // this call requires read privilege
                pc = (PasswordCredential)store.getCredential("pc_map", "pc_key");
                if (pc == null) {
                    // key not found, create one
                    pc = CredentialFactory.newPasswordCredential("jdoe",
                        "password".toCharArray());
                    // this call requires write privilege
                    store.setCredential("pc_map", "pc_key", pc);
                    System.out.print("Created ");
                }
                else {
                    System.out.print("Found ");
                }

                System.out.println("password credential: Name=" + pc.getName() +
                    ", Password=" +
                    new String(pc.getPassword()));
            } catch (CredentialAlreadyExistsException e) {
                // ignore since credential already exists.
                System.out.println("Credential already exists for
                <pc_map, pc_key>: " + pc.getName() + ":" +
                    new String(pc.getPassword()));
            }

            try {
                // permission corresponding to
                // "context=SYSTEM,mapName=gc_map,keyName=gc_key"
                byte[] secret =
                    new byte[] { 0x7e, 0x7f, 0x3d, 0x4f, 0x10,
                        0x20, 0x30 };
                Credential gc =
                    CredentialFactory.newGenericCredential(secret);
                store.setCredential("gc_map", "gc_key", gc);
                System.out.println("Created generic credential");
            } catch (CredentialAlreadyExistsException e) {
                // ignore since credential already exists.
                System.out.println("Generic credential already exists
                for <gc_map,gc_key>");
            }

            try {
                //no permission for pc_map2 & pc_key2 to perform
                //operation on store
                Credential pc2 =
                    CredentialFactory.newPasswordCredential("pc_jode2",
                        "pc_password".toCharArray());
            }
        }
    }
}

```

```

        store.setCredential("pc_map2", "pc_key2", pc2);

    } catch (Exception expected) {
        //CredentialAccess Exception expected here. Not enough permission
        System.out.println("This is expected : " +
            expected.getLocalizedMessage());
    }

    } catch (JpsException e) {
        e.printStackTrace();
    }
}

/*
 * This method performs a non-privileged operation. Either all code
 * in the call stack must have CredentialAccessPermission
 * OR
 * the caller must have the CredentialAccessPermission only and
 * invoke this operation in doPrivileged block
 */
public void doCredOperation() {
    doOperation();
}

/*
 * Since this method performs a privileged operation, only current class or
 * jar containing this class needs CredentialAccessPermission
 */
public void doPrivilegedCredOperation() {
    AccessController.doPrivileged(new PrivilegedAction<String>() {
        public String run() {
            doOperation();
            return "done";
        }
    });
}
}
}

```

24.7.2 Example 1: Java SE Application with Wallet Store

This example shows a sample Java SE application using wallet credentials, that is, a file-based provider.

The example illustrates:

- how the permissions are set in an xml-based policy store (jazn-data.xml)
- how the configuration file is set up
- the Java SE code

jazn-data.xml File

For illustration, the example uses an xml-based policy store file which has the appropriate permissions needed to access the given credential from the store. The file defines the permissions for different combinations of map name (alias) and key. Other combinations, or attempts to access the store beyond the permissions defined here, will be disallowed.

Note: The default policy store to which this grant is added is
 \$DOMAIN_HOME/config/fmwconfig/system-jazn-data.xml.

Here the system property `projectsrc.home` is set to point to the directory containing the Java SE application, and `clientApp.jar` is the application jar file which is present in sub-directory `dist`.

The corresponding policy grant looks like this:

```
<grant>
  <grantee>
    <codesource>
      <url>file:${projectsrc.home}/dist/clientApp.jar</url>
    </codesource>
  </grantee>
  <permissions>
    <permission>
      <class>oracle.security.jps.service.credstore.CredentialAccessPermission
    </class>
    <name>context=SYSTEM,mapName=pc_map,keyName=*</name>
    <actions>read,write</actions>
  </permission>
  <permission>
    <class>oracle.security.jps.service.credstore.CredentialAccessPermission
    </class>
    <name>context=SYSTEM,mapName=gc_map,keyName=gc_key</name>
    <actions>write</actions>
  </permission>
  </permissions>
</grant>
```

Note that no permission has been granted to `mapName=pc_map2, keyName=pc_key2`, hence the `setCredential` call for this map and key combination in [Section 24.7.1, "Code for CSF Operations"](#) is expected to fail.

jps-config-jse.xml File

Note: For the complete configuration file see the default file shipped with the distribution at
 \$DOMAIN_HOME/config/fmwconfig/jps-config-jse.xml.

The location property of the credential store service shows the directory containing the wallet file:

```
<jpsConfig>
  ...
  <serviceInstances>
    <serviceInstance name="credstore_file_instance"
      provider="credstore_file_provider">
      <property name="location" value="store" />
    </serviceInstance>
  </serviceInstances>
  ...
</jpsConfig>
```

Note: The default value of location is "./", that is, the current directory relative to the location of `jps-config-jse.xml`. To use a different path, be sure to specify the full path.

The wallet name is always `cwallet.sso` which is the default file-based Oracle wallet.

Java Code

Here is the Java SE code that calls the utility program.

```
package demo;

import java.io.ByteArrayInputStream;

import java.security.AccessController;
import java.security.PrivilegedAction;

import oracle.security.jps.JpsContext;
import oracle.security.jps.JpsContextFactory;
import oracle.security.jps.JpsException;
import oracle.security.jps.internal.policystore.JavaPolicyProvider;
import oracle.security.jps.jaas.JavaPolicy;
import oracle.security.jps.service.credstore.Credential;
import oracle.security.jps.service.credstore.CredentialAlreadyExistsException;
import oracle.security.jps.service.credstore.CredentialFactory;
import oracle.security.jps.service.credstore.CredentialStore;
import oracle.security.jps.service.credstore.PasswordCredential;
import oracle.security.jps.service.policystore.PolicyStore;
import oracle.security.jps.service.policystore.PolicyStoreException;

import demo.util.CsfUtil;

public class CsfApp {

    // set the OPSS policy provider explicitly, as required in a Java SE application
    static {
        java.security.Policy.setPolicy(new
oracle.security.jps.internal.policystore.JavaProvider());
    }

    public CsfApp() {
        super();
    }

    public static void main(String[] a) {
        // perform operation as privileged code
        JpsContextFactory ctxFactory;
        try {
            ctxFactory = JpsContextFactory.getContextFactory();
            JpsContext ctx = ctxFactory.getContext();

            CredentialStore store =
                ctx.getServiceInstance(CredentialStore.class);
            CsfUtil csf = new CsfUtil(store);
            // #1 - this call is in a doPrivileged block
            // #1 - this should succeed.
            csf.doPrivilegedCredOperation();
        }
    }
}
```



```

        // #2 - this will also pass since granted all application
        // code necessary permission
        // NOTE: Since this call is not in a doPrivileged block,
        // this call would have failed if CredentialAccessPermission
        // wasn't granted to this class.
        /*
        csf.doCredOperation();
        */
    } catch (JpsException e) {
        e.printStackTrace();
    }
}
}
}

```

Notes:

- It is not necessary to replace the JDK-wide policy object. Since the example grant shown conforms to the OPSS XML policy store, it is reasonable to set the policy provider to the OPSS provider.
 - In a Java EE environment for a JRF install for a supported application server, the OPSS policy provider will have been initialized.
-
-

24.7.3 Example 2: Java EE Application with Wallet Store

This example shows a sample Java EE application using wallet credentials. A simple servlet calls the CSF API.

The jazn-data.xml File

The `jazn-data.xml` file for this example defines the appropriate permissions needed to access the given credential from the store. The file defines both the codesource permissions and the permissions for different combinations of map name (alias) and key. Other combinations, or attempts to access the store beyond the permissions defined here, will be disallowed.

A fragment of the policy file showing the corresponding policy grant looks like this:

```

<grant>
  <grantee>
    <codesource>
      <url>file:${oracle.deployed.app.dir}/<MyApp>${oracle.deployed.app.ext}</url>
    </codesource>
  </grantee>
  <permissions>
    <permission>
      <class>oracle.security.jps.service.credstore.CredentialAccessPermission
      </class>
      <name>context=SYSTEM,mapName=pc_map,keyName=*</name>
      <actions>read,write</actions>
    </permission>
    <permission>
      <class>oracle.security.jps.service.credstore.CredentialAccessPermission
      </class>
      <name>context=SYSTEM,mapName=gc_map,keyName=gc_key</name>
      <actions>write</actions>
    </permission>
  </permissions>
</grant>

```

```

    </permissions>
</grant>

```

Note that the first map and key permissions enable both read and write operations; the second enable write operations but not reads.

jps-config.xml File

A portion of the default configuration file `jps-config.xml` showing the credential store configuration is as follows:

```

<jpsConfig>
  <serviceProviders>
    <serviceProvider type="CREDENTIAL_STORE" name="credstoressp"
class="oracle.security.jps.internal.credstore.ssp.SspCredentialStoreProvider">
      <description>SecretStore-based CSF provider</description>
    </serviceProvider>
  </serviceProviders>

  <serviceInstances>
    <serviceInstance name="credstore" provider="credstoressp">
      <property name="location" value="." />
    </serviceInstance>
  </serviceInstances>

  <jpsContexts default="default">
    <jpsContext name="default">
      ...
      <serviceInstanceRef ref="credstore" />
      ...
    </jpsContext>
  </jpsContexts>
</jpsConfig>

```

The `location` property specifies the wallet location; this specification is essentially the same as in Example 1, except that in this example the wallet is located inside the configuration directory. The wallet name is always `cwallet.sso`.

Java Code

```

package demo;

import demo.util.CsfUtil;

import java.io.IOException;
import java.io.PrintWriter;

import java.net.URL;

import java.util.Date;

import javax.servlet.*;
import javax.servlet.http.*;

import oracle.security.jps.JpsException;
import oracle.security.jps.service.JpsServiceLocator;
import oracle.security.jps.service.credstore.CredentialStore;

public class CsfDemoServlet extends HttpServlet {
    private static final String CONTENT_TYPE = "text/html; charset=windows-1252";

```

```

public void init(ServletConfig config) throws ServletException {
    super.init(config);
}

public void doGet(HttpServletRequest request,
                  HttpServletResponse response) throws ServletException,
                  IOException {
    response.setContentType(CONTENT_TYPE);
    PrintWriter out = response.getWriter();
    //ServletOutputStream out = response.getOutputStream();
    try {
        response.setContentType("text/html");
        out.println("<html><body bgcolor=\\"#FFFFFF\\">");
        out.println("<b>Current Time: </b>" + new Date().toString() +
                    "<br><br>");

        //This is to get hold of app-level CSF service store
        //Outside app context, this call returns domain-level CSF store
        //This call also works in Java SE env
        final CredentialStore store =
            JpsServiceLocator.getServiceLocator().lookup(CredentialStore.class);
        CsfUtil csf = new CsfUtil(store);

        csf.doPrivilegedCredOperation();
        out.println("Credential operations completed using privileged code.");
    } catch (JpsException e) {
        e.printStackTrace(out);
    }
}
}

```

The credential create operation is conducted using privileged code. The success of the operation can be verified by using the WLST `listCred` command:

```
listCred(map="pc_map", key="pc_key")
```

Note About Java SE Environment

In the Java SE environment, the following calls are equivalent:

```
CredentialStore store =
JpsServiceLocator.getServiceLocator().lookup(CredentialStore.class);
```

and:

```
CredentialStore store =
JpsContextFactory.getContextFactory().getContext().getServiceInstance(CredentialStore.class);
```

The latter call is shown in [Section 24.7.2, "Example 1: Java SE Application with Wallet Store"](#).

24.7.4 Example 3: Java EE Application with LDAP Store

This example uses the same Java EE application used earlier in Example 2. The only difference is that the credential store is LDAP-based and not file (wallet) based.

You need to configure the following properties in the domain-level `jps-config.xml` file:

- root name

```
<property name="oracle.security.jps.ldap.root.name"
value="cn=OracleJpsContainer" />
```

- farm name

```
<property name="oracle.security.jps.farm.name" value="cn=OracleFarmContainer"
/>
```

The configuration of the LDAP store in `jps-config.xml` is as follows:

```
<jpsConfig>
  <serviceProviders>
    <serviceProvider name="credstore_ldap_provider"
class="oracle.security.jps.internal.credstore.ldap.LdapCredentialStoreProvider">
      <description>Prototype LDAP-based CSF provider</description>
    </serviceProvider>
  </serviceProviders>

  <serviceInstances>
    <serviceInstance provider="ldap.credentialstore.provider"
name="credstore.ldap">
      <property value="bootstrap"
name="bootstrap.security.principal.key" />
      <property value="cn=wls-jrfServer"
name="oracle.security.jps.farm.name" />
      <property value="cn=jpsTestNode"
name="oracle.security.jps.ldap.root.name" />
      <property value="ldap://mynode.us.mycorp.com:1234"
name="ldap.url" />
    </serviceInstance>
  </serviceInstances>

  <jpsContexts default="appdefault">
    <jpsContext name="appdefault">
      <serviceInstanceRef ref="credstore_ldap_instance" />
    </jpsContext>
  </jpsContexts>
</jpsConfig>
```

The highlighted lines define the LDAP parameters necessary to locate the credentials.

24.8 Best Practices

In a clustered environment, use the Credential Store Mbean API over the Credential Store Framework API to create, retrieve, update, and delete credentials for an application.

If you are simply reading credentials, however, either API can be used.

Developing with the User and Role API

This chapter contains these topics:

- [Introduction to the User and Role API Framework](#)
- [Summary of Roles and Classes](#)
- [Working with Service Providers](#)
- [Searching the Repository](#)
- [User Authentication](#)
- [Creating and Modifying Entries in the Identity Store](#)
- [SSL Configuration for LDAP-based User and Role API Providers](#)
- [The User and Role API Reference](#)
- [Developing Custom User and Role Providers](#)
- [The User and Role SPI Reference](#)

Note: The User and Role API is deprecated and may be withdrawn in a future release. Your new applications should be developed on the Identity Governance Framework. Plan to migrate existing applications to the Identity Governance Framework in a future release.

For details, see the *Oracle Fusion Middleware Identity Governance Framework ArisID API Developer's Guide*.

25.1 Introduction to the User and Role API Framework

The User and Role API framework allows applications to access identity information (users and roles) in a uniform and portable manner regardless of the particular underlying identity repository. The repository could be an LDAP directory server such as Oracle Internet Directory, Active Directory (from Microsoft), or Oracle Directory Server Enterprise Edition, or could be a database, flat file, or some other custom repository.

This API framework provides a convenient way to access repositories programmatically in a portable way, freeing the application developer from the potentially difficult task of accounting for the intricacies of particular identity sources. The framework allows an application to work against different repositories seamlessly. An application can switch between various identity repositories without any code changes being required.

Supported operations include creating, updating, or deleting users and roles, or searching users and roles for attributes or information of interest. For example, you may want to search for the e-mail addresses of all users in a certain role.

Note: These APIs are not meant for authentication or authorization functions, but for maintaining identity information.

You can use a basic usage model (without container integration) or a usage model with container integration that allows your code to be portable.

When the application is intended to run in the context of an Oracle WebLogic Server container, the principal class should be cast to `weblogic.security.principal.WLSUserImpl`.

Note: The following are required to invoke the User and Role API:

- The identity store is LDAP-based
 - The domain administration server is up and running
-
-

A Note about Using the User and Role API

As a general rule of thumb, authentication should only be performed by authentication providers, not through the User and Role API.

Additionally, it is recommended that authentication providers be configured with the connect DN of a user that does not have write privileges.

25.1.1 User and Role API and the Oracle WebLogic Server Authenticators

The User and Role API is automatically configured to use the first Oracle WebLogic Server authenticator and does not require any special configuration. F

Note, however, that configuration is required if the User and Role API is going against other authenticators.

The API can access data only from the first LDAP authenticator listed in an Oracle WebLogic Server domain. When more than one authenticator is present, the precedence is determined by their control flag priority. If both have the same priority, the first one is picked. Any LDAP authenticators below the first one on the list are not accessed.

About Concurrent Use of WebLogic APIs

Your application should not try to use both the User and Role API and the WebLogic LDAPAuthenticator API (such as `EmbeddedLDAPAuthenticator`, `OracleInternetDirectoryAuthenticator`, `OracleVirtualDirectoryAuthenticator`) to work on entries in the same LDAP server concurrently. To understand why, consider two LDAP clients, both with caching enabled, that access the same LDAP server; one is deleting entries, and the other tries to use the deleted entries.

The conflict caused by the two clients cannot be resolved unless caching capability is disabled, and the LDAP operations are coordinated among the clients.

25.2 Summary of Roles and Classes

[Table 25–1](#) lists the classes and interfaces of the User and Role API.

Table 25–1 *Classes and Interfaces in the User and Role API*

Name	Type	Description
AuthenticationException	Class	This exception is thrown when an authentication error occurs while accessing the identity store. An authentication error can happen, for example, when the credentials supplied by the user program is invalid or otherwise fails to authenticate the user to the identity store.
AuthenticationWarningException	Class	This class extends IMException (see below).
ComplexSearchFilter	Interface	A complex search filter represents a complex logical expression that can be used to filter results from underlying identity repository. Complex search filter combines multiple SearchFilter instances together with a single logical operator (AND/OR). Each of these component SearchFilter can itself be a complex filter, enabling you to form a complex nested search filter. See the Javadoc (Section 25.9, "The User and Role API Reference") for an example of creating a complex search filter.
ConfigurationException	Class	This exception is thrown when there is a configuration problem. This can arise when configuration information required to access the service provider is malformed or missing.
Identity	Interface	This interface represents a basic identity in the identity repository.
IdentityStore	Interface	IdentityStore represents a handle to actual identity repository. This handle can be used to search, create, drop, and modify identities in the repository.
IdentityStoreFactory	Interface	IdentityStoreFactory is a programmatic representation of underlying identity repository. Actual handle to the identity repository can be obtained by calling <code>getIdentityStoreInstance(Hashtable)</code> on this object.
IdentityStoreFactoryBuilder	Class	This class builds the identity store factory.
IMException	Class	This exception is the superclass of all the exceptions thrown by ADF identity management APIs. The nature of failure is described by the name of the subclass. See the Javadoc (Section 25.9, "The User and Role API Reference") for a list of the direct known subclasses.
ModProperty	Class	This class represents the modification of a property object. ModProperty is called with property name, modified value(s) and type of modification. Modification type can be one of ADD, REMOVE, or REPLACE.
NoPermissionException	Class	This exception is thrown when attempting to perform an operation for which the API caller has no permission. The access control/permission model is dictated by the underlying identity store.
ObjectExistsException	Class	This exception is thrown when an identity with given name is already present in the underlying identity store. For example this exception is thrown when create user API call tries to create a user with the name of an existing user.
ObjectNotFoundException	Class	This exception is thrown when a specified identity does not exist in the identity store.
OperationFailureException	Class	This exception is thrown when an operation fails during execution in the underlying identity store.

Table 25–1 (Cont.) Classes and Interfaces in the User and Role API

Name	Type	Description
OperationNotSupportedException	Class	This exception is thrown by a service provider if it does not support an operation. For example this can be thrown by the service provider, in <code>IdentityStore.getUserManager()</code> call, if it does not provide support for <code>UserManager</code> .
PasswordPolicyException	Class	This class extends <code>IMException</code> (see above).
Property	Class	Property contains name-value information.
PropertySet	Class	A collection of property name and value pairs. Property class is used to represent the property name and value(s) pair. <code>PropertySet</code> guarantees that no two properties have same name.
Role	Interface	This interface represents a role in the identity store.
RoleManager	Interface	This interface represents a role manager that manages execution of various operations, involving roles, in the identity repository.
RoleProfile	Interface	This interface represents the detailed profile of a role.
SearchFilter	Interface	This interface represents a search filter to be used in searching the identity repository.
SearchParameters	Class	This class represents search parameters that need to be specified while performing searches on the identity store. These search parameters are: <ul style="list-style-type: none"> ▪ Search filter, ▪ Search identity type, ▪ page size, ▪ time limit, and ▪ count limit.
SearchResponse	Interface	This interface represents search results obtained after searching the identity store. Its implementation is service provider-specific.
SimpleSearchFilter	Interface	This interface represents a simple search filter to be used while searching the identity repository. Each simple search filter is a logical expression consisting of a search attribute/property, evaluation operator and value. This logical expression will be applied to the underlying identity repository while searching and matching results will be filtered out. See the Javadoc (Section 25.9, "The User and Role API Reference") for an example of a simple search filter.
StoreConfiguration	Interface	<code>StoreConfiguration</code> holds the configuration properties for a given <code>IdentityStore</code> instance. The behavior of this <code>IdentityStore</code> instance can be controlled by changing the properties in this configuration object. The actual configuration properties and their values are specific to the service provider. Some service providers may not support any configuration property at all.
SubjectParser	Interface	This interface provides utility methods for extracting out the user and role principals from the given <code>Subject</code> . Service provider needs to provide the implementation for this interface.
User	Interface	This interface represents a user in the identity store.
UserManager	Interface	This interface represents a user manager that manages execution of various operations, involving users, in the identity repository.

Table 25–1 (Cont.) Classes and Interfaces in the User and Role API

Name	Type	Description
UserProfile	Interface	<p>This interface represents the detailed profile of a user. It allows for user properties to be accessed in a generic manner.</p> <p>You can read or modify any property of user with these APIs:</p> <ul style="list-style-type: none"> ▪ <code>getProperty(java.lang.String)</code> ▪ <code>getProperties(java.lang.String[])</code> ▪ <code>setProperty(oracle.security.idm.ModProperty)</code> ▪ <code>setProperties(oracle.security.idm.ModProperty[])</code>

25.3 Working with Service Providers

In this section we describe basic provider concepts and life cycle, and explain how to set up, configure, and use the provider to work with user repositories in an Oracle Platform Security Services environment.

After ensuring the environment is properly set up, implementing the provider involves:

- identifying the underlying repository and selecting the provider factory class appropriate to that repository
- creating instances of the provider factory and the identity store
- configuring the provider

This section contains these topics:

- [Understanding Service Providers](#)
- [Setting Up the Environment](#)
- [Selecting the Provider](#)
- [Properties for Provider Configuration](#)
- [Programming Considerations](#)
- [Provider Life cycle](#)

25.3.1 Understanding Service Providers

Although the User and Role API is called for user and role management, the API does not directly interact with the underlying identity repository. Instead, security applications make use of *providers* which carry out the actual communication with the underlying repository. This offers flexibility since the same code can be used with various underlying repositories simply by modifying the provider/connection information.

25.3.2 Setting Up the Environment

Jar Configuration

Several jars must be present in your environment:

- the provider jar file, which implements the desired underlying identity repository
- the User and Role API jars

- other component jars which the provider may need, including Toplink, jdbc, xdb, and so on

Ensure that your application classpath includes the relevant jars.

User Classes in jps-config.xml (Oracle Virtual Directory only)

Note: Make this change only for the Oracle Virtual Directory authenticator.

For efficiency when fetching user attributes, add the following entry in `jps-config.xml` to specify the user object classes for the search:

```
.
.
  <serviceInstance name="idstore.ldap" provider="idstore.ldap.provider">
    <property name="idstore.config.provider"
value="oracle.security.jps.wls.internal.idstore.WlsLdapIdStoreConfigProvider" /
>
      <property name="CONNECTION_POOL_CLASS"
value="oracle.security.idm.providers.stdlldap.JNDIPool" />
      <extendedProperty>
        <name>user.object.classes</name>
        <values>
          <value>top</value>
          <value>person</value>
          <value>inetorgperson</value>
          <value>organizationalperson</value>
          <value>otherActiveDirectorySpecificClasses</value>
          ...
        </values>
      </extendedProperty>
.
```

25.3.3 Selecting the Provider

Oracle Platform Security Services support a range of user repositories, including the following LDAP directories:

- Microsoft Active Directory
- Novell eDirectory
- Oracle Directory Server Enterprise Edition
- Oracle Internet Directory
- Oracle Virtual Directory
- OpenLDAP
- Oracle WebLogic Server Embedded LDAP Directory
- Microsoft ADAM
- IBM Tivoli

The choice of identity repository dictates the provider class to use with the provider. The provider class must implement the interface specified by the User and Role API framework. [Table 25–2](#) shows the available provider classes:

Table 25–2 LDAP Identity Provider Classes

Provider	Factory Name
Microsoft Active Directory	oracle.security.idm.providers.ad.ADIIdentityStoreFactory
Novell eDirectory	oracle.security.idm.providers.edir.EDIIdentityStoreFactory
Oracle Directory Server Enterprise Edition	oracle.security.idm.providers.iplanet.IPIIdentityStoreFactory
Oracle Internet Directory	oracle.security.idm.providers.oid.OIDIdentityStoreFactory
OpenLDAP	oracle.security.idm.providers.openldap.OLdapIdentityStoreFactory
Oracle WebLogic Server Embedded LDAP Directory	oracle.security.idm.providers.wlsldap.WLSLDAPIdentityStoreFactory
Oracle Virtual Directory	oracle.security.idm.providers.ovd.OVDIdentityStoreFactory
Microsoft ADAM	oracle.security.idm.providers.ad.ADIIdentityStoreFactory
IBM Tivoli	oracle.security.idm.providers.openldap.OLdapIdentityStoreFactory

25.3.4 Creating the Provider Instance

Once the provider's class name is identified, take these steps to create the provider:

1. Use the `getIdentityStoreFactory` method of the `IdentityStoreFactoryBuilder` class to build a factory instance. The builder class API accepts:
 - the provider class name
 - the necessary environment properties from a hash table
2. Use the `getIdentityStoreInstance` method of the `IdentityStoreFactory` class to create a store instance

The following example creates a factory instance for the Oracle Internet Directory store:

```
IdentityStoreFactoryBuilder builder = new
    IdentityStoreFactoryBuilder ();

IdentityStoreFactory oidFactory = builder.getIdentityStoreFactory(
    "oracle.security.idm.providers.oid.OIDIdentityStoreFactory", factEnv);
```

Now obtain the store reference, which is the actual handle to the identity store:

```
oidStore = oidFactory.getIdentityStoreInstance(storeEnv);
```

Note that two hash-table objects are supplied in these examples:

- the `factEnv` hash table provides the factory instance environment
- the `storeEnv` hash table provides the store instance environment

25.3.5 Properties for Provider Configuration

Configuration is dependent on the identity store provider being used.

You can fine-tune the behavior of all types of LDAP-based identity store providers by configuring a number of properties for the factory instance and the store instance. The following properties are relevant for LDAP-based providers only:

- URL
- the port at which the repository runs
- the user and password to use in accessing the repository

For a list of supported LDAP-based providers, see [Section 25.3.3, "Selecting the Provider"](#).

This section explains the following provider configuration topics:

- [Start-time and Run-time Configuration](#)
- [ECID Propagation](#)
- [When to Pass Configuration Values](#)

25.3.5.1 Start-time and Run-time Configuration

The properties that can be configured fall into two categories:

- Start-time configuration - the naming convention uses property names starting with ST_.
- Run-time configuration - the naming convention uses property names starting with RT_.

Start-time Configuration Properties

Start-time configuration is performed only once, and once set, the configuration settings persist for the duration of the provider's lifetime.

With the exception of ST_SUBSCRIBER_NAME, the start-time properties are specified when creating the provider factory instance; ST_SUBSCRIBER_NAME is set when creating the store instance.

[Table 25-3](#) lists the start-time configuration properties:

Table 25-3 Start-time Identity Provider Configuration Properties

Property Name	Description
ST_BINARY_ATTRIBUTES	An array of Array of String objects containing the names of binary attributes stored in the underlying LDAP server. The provider will treat these attributes as binary while sending data to and receiving it from the LDAP server.
ST_CONNECTION_POOL	External connection pool, an instance of class oracle.idm.connection.ConnectionPool. If set, the provider uses this pool to acquire connections to the LDAP server, and the properties ST_SECURITY_PRINCIPAL, ST_SECURITY_CREDENTIALS, and ST_LDAP_URL are ignored.
ST_USER_NAME_ATTR	The attribute used to determine the username of the user in the identity repository.
ST_GROUP_NAME_ATTR	The attribute used to determine the role name in the identity repository.
ST_USER_LOGIN_ATTR	The attribute used to determine the login ID of the user in the identity repository.
ST_SECURITY_PRINCIPAL	The user (principal).

Table 25–3 (Cont.) Start-time Identity Provider Configuration Properties

Property Name	Description
ST_SECURITY_CREDENTIALS	The credentials necessary to log in to the identity repository.
ST_LDAP_URL	The URL of the identity repository.
ST_MAX_SEARCHFILTER_LENGTH	The maximum length of the search filter allowed by the LDAP server.
ST_LOGGER	The logger object that is to be used by the API.
ST_SUBSCRIBER_NAME	The base DN of operations in the LDAP server. This property is specified while creating the IdentityStore instance and is used to determine default values for remaining properties. This property must be specified while creating the IdentityStore instance; however, subsequent changes to its value have no effect on IdentityStore behavior.
ST_CONNECTION_POOL_CLASS	The fully-qualified Connection Pool implementation class name.
ST_INITIAL_CONTEXT_FACTORY	The fully-qualified class name of the initial context factory that will create the initial context.

Run-time Configuration Properties

Properties set at runtime affect all subsequent operations executed by the provider and control the behavior of the IdentityStore instance of the provider.

Runtime properties are configured by specifying the appropriate parameters and values for the StoreConfiguration object obtained from the IdentityStore instance. All runtime properties have default values when the IdentityStore instance is created, and can be subsequently changed.

Table 25–3 lists the run-time configuration properties:

Table 25–4 Runtime Identity Provider Configuration Properties

Property Name	Description
RT_USER_OBJECT_CLASSES	array of object classes required to create a user in the LDAP server
RT_USER_MANDATORY_ATTRS	attribute names that must be specified while creating a user
RT_USER_CREATE_BASES	Base DNs in the LDAP server where a new user can be created
RT_USER_SEARCH_BASES	the base DNs in the LDAP server that can be searched for users
RT_USER_FILTER_OBJECT_CLASSES	array of object classes to use when searching for a user in the LDAP server
RT_GROUP_OBJECT_CLASSES	array of object classes required to create a role in the LDAP server
RT_GROUP_MANDATORY_ATTRS	attribute names that must be specified when creating a role
RT_GROUP_CREATE_BASES	the base DNs in the LDAP server where a new role can be created
RT_GROUP_SEARCH_BASES	the base DNs in the LDAP server that can be searched for a role

Table 25–4 (Cont.) Runtime Identity Provider Configuration Properties

Property Name	Description
RT_GROUP_MEMBER_ATTRS	An array of member attribute(s) in a role. All members of a role have value(s) for the attribute(s).
RT_GROUP_FILTER_OBJECT_CLASSES	an array of object classes to use when searching for a role in the LDAP server
RT_USER_SELECTED_CREATE_BASE	The currently selected user create base. The user will be created in this base DN upon execution of the createUser() call. If the selected create base is null and the ST_SUBSCRIBER_NAME is not specified, the first supplied value of the RT_USER_CREATE_BASE is used. If the ST_SUBSCRIBER_NAME is specified, the default value is relative to the subscriber name based on the identity store type.
RT_GROUP_SELECTED_CREATE_BASE	The currently selected role create base. This role will be created in this base DN upon execution of the createRole() call. If the selected create base is null and the ST_SUBSCRIBER_NAME is not specified, the first supplied value of the RT_GROUP_CREATE_BASE is used. If the ST_SUBSCRIBER_NAME is specified, the default value is relative to the subscriber name based on the identity store type.
RT_GROUP_GENERIC_SEARCH_BASE	A generic role search base to use in searching the roles related to a given identity. For example while searching all granted roles for a user, or all managed roles for a user, we need a search base under which all the required groups would reside; this helps in optimizing the searches. This search base is usually a common parent. By default, in all LDAP providers this value is set to the subscriber name if provider, else it uses the first group search base.
RT_SEARCH_TYPE	determines whether a search on the LDAP server should be of type SIMPLE, PAGED, or VIRTUAL_LIST_VIEW

25.3.5.2 ECID Propagation

By default, ECID support is disabled in the User and Role API.

When initializing the API, set the `ST_ECID_ENABLED` property to true for ECID support, as illustrated in the following example:

```
factEnv.put(OVDIdentityStoreFactory.ST_ECID_ENABLED, "true");
```

Note: This action is necessary only if either Oracle Internet Directory or Oracle Virtual Directory is used as the back-end identity store. It is not necessary if using other repositories such as Microsoft Active Directory or Novell eDirectory.

25.3.5.3 When to Pass Configuration Values

You can specify configuration data:

- when creating a factory instance

See Also: [Section 25.3.6, "Configuring the Provider when Creating a Factory Instance"](#)

- when creating a store instance

See Also: [Section 25.3.7, "Configuring the Provider when Creating a Store Instance"](#)

- at runtime, through a store configuration object

See Also: [Section 25.3.8, "Runtime Configuration"](#)

25.3.6 Configuring the Provider when Creating a Factory Instance

This section contains topics related to configuring the provider during factory instance creation.

Configuration at this stage affects the entire factory object as well as objects created using this specific factory instance. Many start-time properties are set at this time, including these common properties:

- `ST_LDAP_URL` - the URL of the LDAP repository
- `ST_SECURITY_PRINCIPAL` - the user name
- `ST_SECURITY_CREDENTIAL` - the user credentials required to connect to the repository

25.3.6.1 Oracle Internet Directory Provider

In this example, the provider is configured when setting up an Oracle Internet Directory (OID) factory:

```
IdentityStoreFactoryBuilder builder = new
    IdentityStoreFactoryBuilder();
IdentityStoreFactory oidFactory = null;
Hashtable factEnv = new Hashtable();

// Creating the factory instance
factEnv.put(OIDIdentityStoreFactory.ST_SECURITY_PRINCIPAL,
    "User DN");
factEnv.put(OIDIdentityStoreFactory.ST_SECURITY_CREDENTIALS,
    "User password");
factEnv.put(OIDIdentityStoreFactory.ST_LDAP_URL,
    "ldap://ldaphost:port");
oidFactory = builder.getIdentityStoreFactory(
    "oracle.security.idm.providers.oid.
    OIDIdentityStoreFactory", factEnv);
```

Note: The values in italics must be replaced with appropriate values prior to execution.

25.3.6.2 Using Existing Logger Objects

You can supply named logger objects to the User and Role API. The API uses the specified logger to log messages. You must supply the external logger's name as an environment variable during the factory creation.

Here is an example:

```
Logger mylogr = Logger.getLogger("mylogger.abc.com");
FileHandler fh = new FileHandler("userroleapi.log");
mylogr.addHandler(fh);
```

...

```
factEnv.put(OIDIdentityStoreFactory.ST_LOGGER_NAME,
"mylogger.abc.com");
oidFactory = builder.getIdentityStoreFactory(
    "oracle.security.idm.providers.oid.
    OIDIdentityStoreFactory", factEnv);
```

This code directs that all the log messages should be redirected to the log file named `userroleapi.log`.

25.3.6.3 Supplying Constant Values

You can overwrite constants or pre-supply values for missing constants by supplying the map in the `ST_PROPERTY_ATTRIBUTE_MAPPING` property during factory creation.

This example code sets the mapping of `RoleProfile.OWNER` to the "myowner" attribute. In this way, all operations related to the owner, such as `getOwners()`, `getOwnedRoles()`, and so on, are performed using this attribute.

```
factEnv.put
    (IPIIdentityStoreFactory.ST_SECURITY_PRINCIPAL, "<User DN>");
factEnv.put
    (IPIIdentityStoreFactory.ST_SECURITY_CREDENTIALS, "<User password>");
factEnv.put(IPIIdentityStoreFactory.ST_LDAP_URL,
    "ldap://ldaphost:port/");
```

```
Map m = new Hashtable();
m.put(RoleProfile.OWNER, "myowner");
```

```
factEnv.put
    (IPIIdentityStoreFactory.ST_PROPERTY_ATTRIBUTE_MAPPING, m);

ipFactory = builder.getIdentityStoreFactory(
    "oracle.security.idm.providers.ipplanet.IPIIdentityStoreFactory",
    factEnv);
```

25.3.6.4 Configuring Connection Parameters

You can configure the connection pool parameters for minimum/maximum connections using `ST_CONNECTION_POOL_MIN_CONNECTIONS` and `ST_CONNECTION_POOL_MAX_CONNECTIONS` respectively. By default, the values for these parameters are "0" and "10" respectively. There is an additional restriction that:

$$(ST_CONNECTION_POOL_MAX_CONNECTIONS - ST_CONNECTION_POOL_MIN_CONNECTIONS) \geq 10$$

Here is an example:

```
factEnv.put
    (LDIdentityStoreFactory.ST_CONNECTION_POOL_MIN_CONNECTIONS, "3");

factEnv.put
    (LDIdentityStoreFactory.ST_CONNECTION_POOL_MAX_CONNECTIONS, "16");
```


25.3.6.5 Configuring a Custom Connection Pool Class

To use a custom connection pool, you must provide the fully qualified class name of the custom connection pool class, as follows:

```
factEnv.put(OIDIdentityStoreFactory.ST_CONNECTION_POOL_CLASS,
"oracle.security.idm.providers.stldap.JNDIPool");
```

For related information, see [Section L.6, "Failure to Connect to the Embedded LDAP Authenticator."](#)

25.3.7 Configuring the Provider when Creating a Store Instance

The IdentityStore configuration affects the store object and all objects that are created using this store instance. A configuration parameter commonly used with the store is ST_SUBSCRIBER_NAME, which is the only start-time property accepted here. (All the runtime properties can be supplied during identity store creation.)

Continuing with the earlier example in [Section 25.3.6, "Configuring the Provider when Creating a Factory Instance"](#) which created a factory instance, this code creates a handle instance to the store.

```
IdentityStore oidStore = null;
Hashtable storeEnv = new Hashtable();

// Creating the store instance
storeEnv.put(OIDIdentityStoreFactory.ST_SUBSCRIBER_NAME,
"dc=us,dc=oracle,dc=com");
oidStore = oidFactory.getIdentityStoreInstance(storeEnv);
```

Note: Directories require that you supply a valid subscriber name. For Oracle Internet Directory, you can supply the STsubscriber name as either a proper DN or as the nickname of the realm.

25.3.8 Runtime Configuration

Earlier, in [Section 25.3.6, "Configuring the Provider when Creating a Factory Instance"](#) and [Section 25.3.7, "Configuring the Provider when Creating a Store Instance"](#), we demonstrated how to perform configuration when creating an instance. To facilitate adding and modifying properties at runtime, the User and Role APIs also provide a Configuration class.

The Configuration instance can be obtained from the store instance using the IdentityStore.getStoreConfiguration() API call. Properties can be modified using the configuration object.

Only *runtime* properties can be modified using this approach, and the effect is visible only at runtime.

This example sets the RT_USER_SEARCH_BASES property:

```
StoreConfiguration conf = oidStore.getStoreConfiguration();
conf.setProperty("RT_USER_SEARCH_BASES", "dc=us,dc=oracle,dc=com");
```

25.3.9 Programming Considerations

This section contains tips for working with providers and provider artifacts.

25.3.9.1 Provider Portability Considerations

To ensure that your application is portable when switching providers (say, from OpenLDAP provider to Oracle Internet Directory provider or the converse), follow these guidelines when working with the User and Role API:

1. Use only the corresponding `oracle.security.idm.UserProfile` constants to refer to user properties. Avoid using native names which are not portable across identity repositories. For example, if the application needs to obtain a user's login name, fetch it using the `UserProfile.USER_NAME` constant:

```
Property prop = usrprofile.getProperty(UserProfile.USER_NAME);
```

2. For obvious reasons, `UserProfile` constants are provided for most standard user properties but not for all possible properties. If the application needs to obtain all the properties of a user generically, use the following code:

```
UserProfile upf = null;

// Obtain the user profile from user object. User object can be obtained using
search

// get the properties supported for given user in currently configured
repository
List proplst = store.getUserPropertyNames();

String[] proparr = (String[]) proplst.toArray(new String[proplst.size()]);

// get all properties of the user
PropertySet pset = upf.getProperties(proparr);
```

3. When creating search filters, do not use native wild card characters directly in your search filter string. Instead, use the `SimpleSearchFilter.getWildcardChar()` method. This will fetch the correct wild character based upon the underlying provider. For example, the API will return `%` for say a database provider and return `*` for the Oracle Internet Directory provider.

```
SimpleSearchFilter sf = m_identityStore.getSimpleSearchFilter(
    attrName, SimpleSearchFilter.TYPE_EQUAL, null);

sf.setValue( filterStringWithoutWildcard+sf.getWildcardChar());
```

4. If your application accepts user-supplied filter strings with a predefined wild card character, apply the following conversion on the filter while generating the User and Role API filter:

```
//User supplied filter which assumes "%" as the wildcard character

String userDefinedFilter = .....

SimpleSearchFilter sf = m_identityStore.getSimpleSearchFilter(
    attrName, SimpleSearchFilter.TYPE_EQUAL, null);

userDefinedFilter =
    userDefinedFilter.replaceAll("%", sf.getWildcardChar());

sf.setValue(userDefinedFilter);
```

The line in bold converts the user-supplied filter to the generic User and Role API filter format.

25.3.9.2 Considerations when Using IdentityStore Objects

Keep the following considerations in mind when coding your applications.

Thread Safety

The current IdentityStore implementations are not thread-safe. The User and Role API assumes that the store instances are not generally shared among threads. If the store instance is shared among threads, the application code must take care to handle any required thread safety issues.

There are trade-offs between thread safety and performance. Use cases that need to implement thread safety must be willing to consider the performance implications of doing so.

One Store Instance per Session

In applications such as Delegated Administration Server, each session (corresponding to one logged-in user) can change its own create/search bases and various other runtime settings; these are defined as runtime properties in the User and Role API. The IdentityStore object encapsulates all these settings and changes its runtime behavior accordingly. For this reason, the rule of one IdentityStore instance per session is enforced.

25.3.10 Provider Life cycle

A given provider exists for the lifetime of the factory instance created for that provider. The life of a factory instance ends whenever the close() API is called on that instance. When the provider instance ends, all the objects that were created using that instance become invalid, and subsequent API calls on those objects return unanticipated output.

Similar considerations apply to IdentityStore instances.

Note:

- Factory instances are thread-safe while this is not the case with IdentityStore instances.
 - It is best practice to cascade close server connections and explicitly delete objects and instances no longer in use.
-
-

25.4 Searching the Repository

The User and Role API provides two types of query functions:

- functions that return a single identity
- functions that return a list of identities

This section describes searches and related tasks you can accomplish with the API, and provides details on specifying search parameters:

- [Searching for a Specific Identity](#)
- [Searching for Multiple Identities](#)
- [Specifying Search Parameters](#)
- [Using Search Filters](#)
- [Searching by GUID](#)

25.4.1 Searching for a Specific Identity

You can query the identity store directly for a specific user or role using the `searchUser` and `searchRole` APIs:

```
IdentityStore.searchUser(String name);

IdentityStore.searchUser(Principal principal);

IdentityStore.searchUser(int searchType, String name);
```

where `searchType` can be:

- `SEARCH_BY_NAME`
- `SEARCH_BY_UNIQUE_NAME`

```
IdentityStore.searchRole(int searchType, Sting value);
```

These functions facilitate simple queries where a particular user/role identity is known to exist in the store, and you simply need the object reference to that identity. The functions are minimal in that:

- they accept only string values
- they do not support regular expressions

The functions raise an exception if multiple entities with the same value exist in the store.

25.4.2 Searching for Multiple Identities

The User and Role APIs contain several functions that can perform searches to return multiple identities:

```
IdentityStore.search(SearchParams params);
IdentityStore.searchUsers(SearchParams params);
IdentityStore.searchRoles(int searchType, SearchParams params);
IdentityStore.searchProfiles(SearchParams params);
```

Each function accepts a search object and returns a search response object.

25.4.3 Specifying Search Parameters

The SearchParams Object

The `SearchParams` object contains the following information:

- Search Filter - this is discussed in [Section 25.4.4, "Using Search Filters"](#)
- Search Identity of type - you can search identities of type `Roles` or `Users`
- Page Size - By default the paging option is turned off. If the query needs paging, set the page size to a relevant positive value.
- Timeout limit – timeout is specified in seconds
- Count Limit – limits the number of results returned by the query

The SearchResponse Object

`SearchResponse` is a data structure used when retrieving multiple identities. Your code can iterate through the identities contained in this structure using these functions:

- `hasNext()` - returns `true` if more elements are present, `false` otherwise
- `next()` - returns the next element if it is available, an exception otherwise

25.4.4 Using Search Filters

The User and Role API includes different types of search filters to facilitate a variety of search operations. This section explains key facts about the use of search filters:

- [Operators in Search Filters](#)
- [Handling Special Characters when Using Search Filters](#)
- [Search Filter for Logged-In User](#)
- [Examples of Using Search Filters](#)

25.4.4.1 Operators in Search Filters

Observe these rules when using search filter operators.

Supported Operators

The standard LDAP store accepts only "=" (equals operator), "<" (less-than operator), ">" (greater-than operator), "&" (AND operator), "|" (OR operator) and "!" (NOT operator). IdentityStore provides two more operators to simplify usage, namely "<=" (less than or equal to) and ">=" (greater than or equal to).

The operators "=", "<", ">", "<=" and ">=" are used to create simple search filters while the "&" and "|" operators are used to combine two or more search strings to make a complex search filter.

NOT Operator

You can use the NOT operator in both the simple search filter and complex search filters. This operator is used to negate the state of the filter, that is, the state of the filter is changed to accept the entities which were earlier rejected by the filter, or to reject entities that were earlier accepted.

The NOT operator is accessible using the following `SearchFilter` API:

- `void negate();`
- `boolean isNegated();`

25.4.4.2 Handling Special Characters when Using Search Filters

According to RFC-2254 (String Representation of LDAP Search Filters), "*", "(", ")", "\" and NULL characters are to be handled separately. The User and Role API handles "(", ")", "\" and "\" operators but does not handle the "*" operator, which is also a wild-card character for LDAP stores. The API user is not required to separately handle these characters as the User and Role API framework handles these characters.

25.4.4.3 Search Filter for Logged-In User

Applications commonly need to retrieve the identity of the logged-in user and the user's group name.

The Oracle WebLogic Server authenticator uses two attributes related to users: `user.login.attr` and `groupname.attr`. Upon login, the authenticator uses `user.login.attr` to store the user and `groupname.attr` for the group.

Your application should use `UserProfile.getUserName()` (which maps to `user.login.attr`) to obtain the identity of the logged-in user. To obtain the role (group) name, it should use `RoleProfile.getProperty(RoleProfile.NAME)` (which maps to `groupname.attr`).

Sample calls showing how to obtain the logged-in user and role are shown in [Example 25-6](#) and [Example 25-7](#), respectively.

25.4.4.4 Examples of Using Search Filters

Several usage examples are presented in this section.

Example 25-1 Simple Filter to Retrieve Users by Name

The implementation of the simple search filter depends on the underlying store; you can obtain an instance of the search filter through the store instance.

In this example, the filter allows all entries with a non-null value for the "name" field:

```
SimpleSearchFilter sf =
    oidStore.getSimpleSearchFilter(UserProfile.NAME,
        SimpleSearchFilter.TYPE_EQUAL, null);
sf.setValue(sf.getWildcardChar());
```

Example 25-2 Simple Filter to Find Users by Language Preference

This example retrieves users whose preferred language is not English:

```
SimpleSearchFilter sf =
    oidStore.getSimpleSearchFilter(
        UserProfile.PREFERRED_LANGUAGE,
        SimpleSearchFilter.TYPE_EQUAL,
        "english");
sf.negate();
```

Example 25-3 Complex Filter for Names by Starting Letter

This complex filter combines multiple search filters with operators "&" or "|". It searches for users whose name starts with a letter between "a" and "j":

```
SimpleSearchFilter sf1 =
    oidStore.getSimpleSearchFilter(
        UserProfile.NAME,
        SimpleSearchFilter.TYPE_GREATER,
        null);

sf1.setValue("a"+sf1.getWildcardChar());
SimpleSearchFilter sf2 =
    oidStore.getSimpleSearchFilter(UserProfile.NAME,
        SimpleSearchFilter.TYPE_LESS, null);
sf2.setValue("j"+sf2.getWildcardChar());
SimpleSearchFilter sfArray[] = new SimpleSearchFilter[] {sf1, sf2};
ComplexSearchFilter cf1 =
    store.getComplexSearchFilter(sfArray, ComplexSearchFilter.TYPE_AND);
```

Example 25-4 Complex Filter with Restrictions on Starting Letter

In this example, complex filters are nested to enable a search for users whose name starts with a letter between "a" and "j" but not with the letter "i":

[continue from Example 3]

```
SimpleSearchFilter sf3 =
```

```

oidStore.getSimpleSearchFilter(
    UserProfile.NAME,
    SimpleSearchFilter.TYPE_EQUAL,
    null);

sf3.setValue("i"+sf3.getWildCardChar());
sf3.negate();

SearchFilter sfArray2[] = new SearchFilter[] {cf1, sf3};
ComplexSearchFilter cf2 =
    store.getComplexSearchFilter(sfArray2, ComplexSearchFilter.TYPE_AND);

```

Example 25-5 Complete Search with Output

This example filters names starting with the letter "a" and outputs the return values:

```

// search filter (cn=a*)
SimpleSearchFilter sf = oidStore.getSimpleSearchFilter(
    UserProfile.NAME,
    SimpleSearchFilter.TYPE_EQUAL,
    null);
sf.setValue("a"+sf.getWildCardChar());

SearchParameters params = new SearchParameters();
params.setFilter(sf);

// Searching for users
SearchResponse resp = oidStore.searchUsers(params);
System.out.println("Searched users are:");
while (resp.hasNext()) {
    Identity idy = resp.next();
    System.out.println("Unique name: "+idy.getUniqueName());
}

```

Example 25-6 Obtaining the Identity of the Logged-in User

This example shows how to retrieve the logged-in user:

```

SimpleSearchFilter sf = oidStore.getSimpleSearchFilter(
    UserProfile.USER_NAME, SimpleSearchFilter.TYPE_EQUAL, "sampleUserName");
SearchParameters ssp = new SearchParameters(sf, SearchParameters.SEARCH_USERS_
ONLY);

// Searching for users
SearchResponse resp = oidStore.searchUsers(params);
System.out.println("Searched users are:");
while (resp.hasNext()) {
    Identity idy = resp.next();
    String foundUserName = ((User)idy).getUserProfile().getUserName();
    System.out.println("Found user name: "+ foundUserName );
}

```

Note: The name returned by `((User) idy).getName` is derived from the RDN, which might be different from the login name.

Example 25–7 Obtaining the Role/Group Name

This example shows how to retrieve the role (group) name:

```
Role aRole = idStore.searchRole(IdentityStore.SEARCH_BY_NAME, "sampleRoleName");
Property prop = aRole.getRoleProfile().getProperty(RoleProfile.NAME);

List roleList = prop.getValues();
Iterator itr = roleList.iterator();
System.out.println("Searched roles are:");
while (itr.hasNext()) {
    String foundRoleName = (String)itr.next();
    System.out.println("Found role name: "+ foundRoleName );
}
```

25.4.5 Searching by GUID

In this example, GUID values obtained from the User and Role API can be directly used in the search:

```
// up = user.getUserProfile();
String guid = up.getGUID();
SimpleSearchFilter sfl = oidStore.getSimpleSearchFilter(
    UserProfile.GUID,
    SimpleSearchFilter.TYPE_EQUAL, guid);
SearchParameters params = new SearchParameters();
params.setFilter(sfl);
SearchResponse resp = oidStore.search(params);
while (resp.hasNext())
    System.out.println("user for guid : " + guid + ", "+ resp.next());
```

25.5 User Authentication

For verification purposes, you can use the User and Role API for password-based authentication of users. (As mentioned earlier, the API is not meant for authentication and authorization.)

The `authenticateUser` API accepts a user login name and attempts to authenticate the user with the specified password. If authentication is successful, it returns the user object.

Here is an example of password-based authentication:

```
store.getUserManager().authenticateUser("testuser", "password");
```

25.6 Creating and Modifying Entries in the Identity Store

The User and Role API facilitates adding new identities to the identity store and modifying identities in the store. The `UserManager` and `RoleManager` classes address the user- and role-specific data creation, modification and deletion operations.

`UserManager` and `RoleManager` instances can be obtained from the store instance as follows:

```
UserManager um = oidStore.getUserManager();
RoleManager rm = oidStore.getRoleManager();
```

Topics in this section include:

- [Handling Special Characters when Creating Identities](#)
- [Creating an Identity](#)
- [Modifying an Identity](#)
- [Deleting an Identity](#)

25.6.1 Handling Special Characters when Creating Identities

RFC-2253 defines the string representation of Distinguished Names for LDAP v3. This means that all the characters specified in the RFC are handled. The User and Role API user does not need to escape/de-escape those special characters; attempting to do so will cause erroneous results.

There could be a problem when creating identities with empty properties. In this case, the "RDN name" is used to fill in the values of various mandatory attributes. Some of these attributes could have stricter validation rules. In this case, the creation of the identity fails and an exception is raised.

25.6.2 Creating an Identity

Two functions in the `UserManager` class facilitate creating a user:

```
createUser(java.lang.String name, char[] password)
```

creates a user with the specified name and password in the underlying repository.

When the identity store designates that some attributes are mandatory, all such fields will be populated with the "name" value.

```
createUser(java.lang.String name, char[] password, PropertySet suppliedProps)
```

Properties are set using the supplied property values. If any mandatory attribute values are not supplied, the missing attributes will use the "name" value as the default.

Likewise, `RoleManager` APIs are used to create roles.

Roles are organized into two categories:

- application scope
- enterprise scope

When you invoke `RoleManager` to create a role, by default the role is created in the enterprise scope unless you specify otherwise.

`RoleManager` APIs supporting role creation are:

```
createRole(String roleName);
createRole(String roleName, int roleScope);
```

The procedure for creating a role is similar to that for creating a user, and all mandatory attributes must be supplied with `roleName`.

25.6.3 Modifying an Identity

To modify an identity, you need a reference to the identity. The `User`, `UserProfile`, `Role`, and `RoleProfile` classes provide the following APIs to facilitate modifying identities:

```
user.setProperty(ModProperty prop);
```

```
user.setProperty (ModProperty [] props);
```

ModProperty structure consists of:

- the field name
- its new value(s)
- the modifying operator

Valid operators are:

```
ModProperty.ADD  
ModProperty.REMOVE  
ModProperty.REPLACE
```

In this example, a display name is replaced:

```
UserProfile usrprofile = usr.getUserProfile();  
  
ModProperty mprop = new ModProperty (UserProfile.DISPLAY_NAME,  
    "modified display name",  
    ModProperty.REPLACE);  
  
usrprofile.setProperty (mprop);
```

Modifying a particular value in a multi-valued attribute is a two-step process; first remove the value, then add the new value.

25.6.4 Deleting an Identity

You drop an identity with the `dropUser` and `dropRole` APIs.

You need both user and role references in your code when dropping an identity. Here is an example:

```
User usr;  
Role role;  
...  
...  
usrmanager.dropUser (usr);  
rolemanager.dropRole (role);
```

25.7 Examples of User and Role API Usage

This section contains some examples illustrating practical applications of the User and Role API:

- [Example 1: Searching for Users](#)
- [Example 2: User Management in an Oracle Internet Directory Store](#)
- [Example 3: User Management in a Microsoft Active Directory Store](#)

25.7.1 Example 1: Searching for Users

In this example the identity store is Oracle Internet Directory, and a simple search filter is set up to search for users:

```
import oracle.security.idm.*;  
import oracle.security.idm.providers.oid.*;  
import java.util.*;
```

```

import java.io.*;

public class SearchUsersOID
{
    public static void main(String args[])
    {
        IdentityStoreFactoryBuilder builder = new
IdentityStoreFactoryBuilder();
        IdentityStoreFactory oidFactory = null;
        IdentityStore oidStore = null;

        try
        {

            Hashtable factEnv = new Hashtable();
            Hashtable storeEnv = new Hashtable();

            // creating the factory instance
            factEnv.put(OIDIdentityStoreFactory.ST_SECURITY_PRINCIPAL,
                "<User DN>");
            factEnv.put(OIDIdentityStoreFactory.ST_SECURITY_CREDENTIALS,
                "<User password>");
            factEnv.put(OIDIdentityStoreFactory.ST_LDAP_URL,
                "ldap://ldaphost:port/");
            oidFactory = builder.getIdentityStoreFactory(
                "oracle.security.idm.providers.oid.OIDIdentityStoreFactory",
                factEnv);

            // creating the store instance
            storeEnv.put(OIDIdentityStoreFactory.RT_SUBSCRIBER_NAME,
                "<Subscriber name>");
            oidStore = oidFactory.getIdentityStoreInstance(storeEnv);

            // search filter (cn=a*)
            SimpleSearchFilter sf = oidStore.getSimpleSearchFilter(
                UserProfile.NAME, SimpleSearchFilter.TYPE_EQUAL, null);
            sf.setValue("a"+sf.getWildCardChar());
            // sf2 search filter (!(cn=*a))
            SimpleSearchFilter sf2 = oidStore.getSimpleSearchFilter(
                UserProfile.NAME, SimpleSearchFilter.TYPE_EQUAL, null);
            sf2.setValue(sf.getWildCardChar()+"a");
            sf2.negate();

            SimpleSearchFilter sfArray[] = new SimpleSearchFilter[] {sf,sf2};
            ComplexSearchFilter cf1 = oidStore.getComplexSearchFilter(sfArray,
                ComplexSearchFilter.TYPE_AND);

            SearchParameters params = new SearchParameters();
            params.setFilter(cf1);

            // Searching for users
            SearchResponse resp = oidStore.searchUsers(params);
            System.out.println("Searched users are:");
            while (resp.hasNext()) {
                Identity idy = resp.next();
                System.out.println("Unique name: "+idy.getUniqueName());
            }
        }catch (IMException e)
        {
            e.printStackTrace();
        }
    }
}

```

```
    }  
  }  
}
```

Searching for Users and Searching for Groups

When searching for users, you invoke `UserProfile`, as in the above example with `SimpleSearchFilter`. When searching for groups, however, you use `RoleProfile` instead.

25.7.2 Example 2: User Management in an Oracle Internet Directory Store

In this example several user management tasks such as creating, modifying, and dropping an identity are performed in an Oracle Internet Directory store:

- creating a user
- modifying the user's display name
- dropping the user

```
public class CreateModifyDeleteUser  
{  
    public static void main(String args[])  
    {  
        IdentityStoreFactoryBuilder builder = new  
IdentityStoreFactoryBuilder();  
        IdentityStoreFactory oidFactory = null;  
        IdentityStore oidStore = null;  
  
        try  
        {  
  
            Hashtable factEnv = new Hashtable();  
            Hashtable storeEnv = new Hashtable();  
  
            // creating the factory instance  
            factEnv.put(OIDIdentityStoreFactory.ST_SECURITY_PRINCIPAL,  
                "<User DN>");  
            factEnv.put(OIDIdentityStoreFactory.ST_SECURITY_CREDENTIALS,  
                "<User password>");  
            factEnv.put(OIDIdentityStoreFactory.ST_LDAP_URL,  
                "ldap://ldaphost:port/");  
            oidFactory = builder.getIdentityStoreFactory(  
                "oracle.security.idm.providers.oid.  
OIDIdentityStoreFactory",  
                factEnv);  
  
            // creating the store instance  
            storeEnv.put(OIDIdentityStoreFactory.RT_SUBSCRIBER_NAME,  
                "dc=us,dc=oracle,dc=com");  
            oidStore = oidFactory.getIdentityStoreInstance(storeEnv);  
  
            //get UserManager  
            UserManager usrmanager = oidStore.getUserManager();  
  
            // create user  
            String username = "testuser";  
            // delete user if already exists  
            try  
            {
```

```

        User usr = oidStore.searchUser(usrname);
        usrmanager.dropUser(usr);
    }catch(IMException ime){}

    System.out.println("creating user "+usrname);
    User usr =
usrmanager.createUser(usrname,"passwd1".toCharArray());
    System.out.println("user (" +usr.getUniqueName() + ") created");

    // modifying user properties
    System.out.println("modifying property
UserProfile.DISPLAY_NAME");
    UserProfile usrprofile = usr.getUserProfile();
    ModProperty mprop = new ModProperty(
UserProfile.DISPLAY_NAME,
        "modified display name",
        ModProperty.REPLACE);

    usrprofile.setProperty(mprop);

    System.out.println("get property values
UserProfile.DISPLAY_NAME");
    Property prop = usrprofile.getProperty(UserProfile.DISPLAY_NAME);
    List values = prop.getValues();
    Iterator itr = values.iterator();
    while(itr.hasNext()) {
        System.out.println(UserProfile.DISPLAY_NAME+": "+ itr.next());
    }
    System.out.println();

    // drop user
    System.out.println("Now dropping user "+usrname);
    usrmanager.dropUser(usr);
    System.out.println("user dropped");

}catch (IMException e)
{
    e.printStackTrace();
}
}
}

```

25.7.3 Example 3: User Management in a Microsoft Active Directory Store

In this example several user management tasks such as creating, modifying, and dropping an identity are performed in a Microsoft Active Directory store:

- creating a user
- modifying the user's display name
- dropping the user

```

package oracle.security.idm.samples;

import oracle.security.idm.*;
import oracle.security.idm.providers.ad.*;
import java.util.*;
import java.io.*;

```

```
public class CreateModifyDeleteUserAD
{
    public static void main(String args[])
    {
        IdentityStoreFactoryBuilder builder = new IdentityStoreFactoryBuilder();
        IdentityStoreFactory adFactory = null;
        IdentityStore adStore = null;

        try
        {

            Hashtable factEnv = new Hashtable();
            Hashtable storeEnv = new Hashtable();

            String keystore = "/home/bhusingh/client_keystore.jks";
            System.setProperty("javax.net.ssl.trustStore", keystore);
            System.setProperty("javax.net.ssl.trustStorePassword", "welcome1");

            // creating the factory instance
            factEnv.put(ADIdentityStoreFactory.ST_SECURITY_PRINCIPAL,
                "sramaset@upad.us.oracle.com");
            factEnv.put(ADIdentityStoreFactory.ST_SECURITY_CREDENTIALS,
                "ntrtntrt");
            factEnv.put(ADIdentityStoreFactory.ST_LDAP_URL,
                "ldaps://mynode.us.mycorp.com:123/");
            factEnv.put("java.naming.security.protocol", "SSL");

            adFactory = builder.getIdentityStoreFactory(
                "oracle.security.idm.providers.ad.ADIdentityStoreFactory",
                factEnv);

            // creating the store instance
            storeEnv.put(ADIdentityStoreFactory.ST_SUBSCRIBER_NAME,
                "dc=upad,dc=us,dc=oracle,dc=com");
            adStore = adFactory.getIdentityStoreInstance(storeEnv);

            //get UserManager
            UserManager usrmanager = adStore.getUserManager();

            // create user
            String username = "amyd";
            // delete user if already exists
            try
            {
                User usr = adStore.searchUser(username);
                usrmanager.dropUser(usr);
            } catch (IMException ime) {}

            System.out.println("creating user "+username);
            char[] password = {'w', 'e', 'l', 'c', 'o', 'm', 'e', '3'};
            User usr = usrmanager.createUser(username, password);
            System.out.println("user (" +usr.getUniqueName() + ") created with
guid="+usr.getGUID());
            System.out.println("user name = "+usr.getName() );

            // modifying user properties
            System.out.println("DISPLAY_NAME="+usr.getDisplayName());
            System.out.println("modifying property UserProfile.DISPLAY_NAME");
            UserProfile usrprofile = usr.getUserProfile();
```

```

ModProperty mprop = new ModProperty(UserProfile.DISPLAY_NAME,
                                     "modified display name",
                                     ModProperty.REPLACE);

usrprofile.setProperty(mprop);

System.out.println("get property values UserProfile.DISPLAY_NAME");
Property prop = usrprofile.getProperty(UserProfile.DISPLAY_NAME);
List values = prop.getValues();
Iterator itr = values.iterator();
while(itr.hasNext())
{
    System.out.println(UserProfile.DISPLAY_NAME+": "+ itr.next());
}
System.out.println();

System.out.println("now verifying the password");
boolean pass = false;
try
{
    usrmanager.authenticateUser(usrname, password);
    pass= true;
}catch (oracle.security.idm.AuthenticationException e)
{
    System.out.println(e);
    e.printStackTrace();
}
if (pass)
    System.out.println("password verification SUCCESS !!");
else
    System.out.println("password verification FAILED !!");

SimpleSearchFilter sf = adStore.getSimpleSearchFilter(
    UserProfile.NAME, SimpleSearchFilter.TYPE_EQUAL, usrname);

SearchParameters params = new SearchParameters();
params.setFilter(sf);

// Searching for users
SearchResponse resp = adStore.searchUsers(params);
System.out.println("Searched users are:");
while (resp.hasNext())
{
    Identity idy = resp.next();
    System.out.println("name: "+idy.getName()+"\tUnique name:
"+idy.getUniqueName());
}

// drop user
System.out.println("Now dropping user "+usrname);
usrmanager.dropUser(usr);
System.out.println("user dropped");

}catch (Exception e)
{
    e.printStackTrace();
}
}
}

```

25.8 SSL Configuration for LDAP-based User and Role API Providers

This section describes SSL support for the User and Role API. It contains these topics:

- [Out-of-the-box Support for SSL](#)
- [Customizing SSL Support for the User and Role API](#)

25.8.1 Out-of-the-box Support for SSL

LDAP-based providers for the User and Role API rely on the Sun Java Secure Sockets Extension (JSSE) to provide secure SSL communication with LDAP-based identity stores. JSSE is part of JDK 1.4 and higher.

These LDAP providers are:

- Microsoft Active Directory
- Novell eDirectory
- Oracle Directory Server Enterprise Edition
- Oracle Internet Directory
- OpenLDAP
- Oracle WebLogic Server Embedded LDAP Directory

25.8.1.1 System Properties

To support SSL you must provide the following information in the form of system properties:

```
javax.net.ssl.keyStore
```

```
javax.net.ssl.keyStorePassword
```

```
javax.net.ssl.trustStore
```

```
javax.net.ssl.trustStorePassword
```

Refer to Sun Microsystems' documentation on JSSE for details.

25.8.1.2 SSL configuration

You need to provide SSL configuration details during User and Role API configuration.

Provide your keystore location and password as system properties to the JVM:

```
String keystore = "<key store location>";  
String keypasswd = "<key store password>";  
System.setProperty("javax.net.ssl.trustStore", keystore);  
System.setProperty("javax.net.ssl.trustStorePassword", keypasswd);
```

Specify following properties in the environment when creating the IdentityStoreFactory instance:

1. Set the SSL URL of the LDAP server, as in this example:

```
factEnv.put(ADIdentityStoreFactory.ST_LDAP_URL,  
            "ldaps://ldaphost:sslport/");
```

2. Set the security protocol to SSL:


```
factEnv.put("java.naming.security.protocol", "SSL");
```

25.8.2 Customizing SSL Support for the User and Role API

You can customize SSL support by providing a customized `SSLConnectionFactory` to the User and Role API provider.

25.8.2.1 SSL configuration

Specify the following properties when creating the `IdentityStoreFactory` instance:

1. Specify the custom SSL socket factory name:

```
factEnv.put("java.naming.ldap.factory.socket",  
"fully qualified custom socket factory name");
```

2. Set the SSL URL of the LDAP server, as in this example:

```
factEnv.put(ADIdentityStoreFactory.ST_LDAP_URL,  
"ldaps://ldaphost:sslport/");
```

3. Set the security protocol to SSL:

```
factEnv.put("java.naming.security.protocol", "SSL");
```

25.9 The User and Role API Reference

The User and Role API reference (Javadoc) is available at:

Oracle Fusion Middleware User and Role Java API Reference for Oracle Platform Security Services

25.10 Developing Custom User and Role Providers

This section explains how to develop custom providers that security developers can use to manage identities (users and roles). It contains these topics:

- [SPI Overview](#)
- [Types of User and Role Providers](#)
- [Developing a Read-Only Provider](#)
- [Developing a Full-Featured Provider](#)
- [Development Guidelines](#)
- [Testing and Verification](#)
- [Example: Implementing an Identity Provider](#)

25.10.1 SPI Overview

The User and Role API is accompanied by a service provider interface (SPI) that makes it possible to develop custom user/role providers. You can use the service provider interface to develop a custom provider for any identity data repository.

The SPI is bundled as the `oracle.security.idm.spi` package, which is a set of abstract classes. Custom User and Role providers are created by extending this SPI to fit your requirements.

See Also: ["The User and Role SPI Reference"](#)

25.10.2 Types of User and Role Providers

The User and Role API offers functions for both search and Create/Read/Update/Delete (CRUD) operations. A User and Role provider based on read-only functions supports only search operations. A full-featured provider supports both search operations and CRUD operations. In other words, the full-featured provider is a superset of a read-only provider.

As a developer you have the choice of creating either read-only or full-functionality providers depending upon the requirements.

It is reasonable to develop a read-only provider in the following situations:

- if the underlying identity repository operates in read-only mode
- if applications consuming the User and Role API do not make any CRUD API calls

For example, it makes sense to develop a read-only provider for use with the SOA identity service.

25.10.3 Developing a Read-Only Provider

This section describes the classes used to implement a provider. Topics include:

- [SPI Classes Requiring Extension](#)
- [oracle.security.idm.spi.AbstractIdentityStoreFactory](#)
- [oracle.security.idm.spi.AbstractIdentityStore](#)
- [oracle.security.idm.spi.AbstractRoleManager](#)
- [oracle.security.idm.spi.AbstractUserManager](#)
- [oracle.security.idm.spi.AbstractRoleProfile](#)
- [oracle.security.idm.spi.AbstractUserProfile](#)
- [oracle.security.idm.spi.AbstractSimpleSearchFilter](#)
- [oracle.security.idm.spi.AbstractComplexSearchFilter](#)
- [oracle.security.idm.spi.AbstractSearchResponse](#)

25.10.3.1 SPI Classes Requiring Extension

[Table 25–5](#) shows that SPI classes that must be extended to implement a read-only provider:

Note: All abstract methods must be implemented.

Table 25–5 *SPI Classes to Extend for Custom Provider*

Class	Usage Notes
oracle.security.idm.spi.AbstractIdentityStoreFactory	The extending class must include a default constructor and a constructor accepting a <code>java.util.Hashtable</code> object.
oracle.security.idm.spi.AbstractIdentityStore	
oracle.security.idm.spi.AbstractRoleManager	
oracle.security.idm.spi.AbstractUserManager	
oracle.security.idm.spi.AbstractRoleProfile	
oracle.security.idm.spi.AbstractUserProfile	
oracle.security.idm.spi.AbstractSimpleSearchFilter	The constructor of the extending class must call the constructor of the abstract (super) class.
oracle.security.idm.spi.AbstractComplexSearchFilter	The constructor of the extending class must call the constructor of the abstract (super) class.
oracle.security.idm.spi.AbstractSearchResponse	

Additional requirements and notes for each class are provided below.

25.10.3.2 oracle.security.idm.spi.AbstractIdentityStoreFactory

The class extending this SPI class must have following constructors:

1. The default constructor (one which has no arguments).
2. A constructor that accepts a `java.util.Hashtable` object as an argument. You can use the hash table to accept any configuration properties required by the provider.

The configuration properties are passed to this constructor during the user and role configuration phase. The properties are key-value pairs passed in the `Hashtable` argument:

- The key must be `java.lang.String`.
- The value can be `java.lang.Object`.

It is recommended that the value be of type `String`. This guarantees that the property can be specified in `jps-config.xml`, which is a text file.

See Also: "[The User and Role SPI Reference](#)" for details about the methods that need to be implemented in this class. All listed methods *must* be implemented.

25.10.3.3 oracle.security.idm.spi.AbstractIdentityStore

"[The User and Role SPI Reference](#)" provides details about the methods that need to be implemented in this class. Note that:

- Method `getStoreConfiguration()` is optional and can throw `OperationNotSupportedException`.
- Method `getSubjectParser()` can return `null`.

When there are no search results to be returned, all search APIs should throw:

`oracle.security.idm.ObjectNotFoundException`

Never return an empty `SearchResponse`.

25.10.3.4 `oracle.security.idm.spi.AbstractRoleManager`

"[The User and Role SPI Reference](#)" provides details about the methods that need to be implemented in this class. Note that only the following methods need concrete/actual implementations:

- `getGrantedRoles()`
- `getOwnedRoles()`
- `getManagedRoles()`
- `isGranted()`
- `isManagedBy()`
- `isOwnedBy()`
- `isDropRoleSupported()` – should always return `false`
- `isCreateRoleSupported()` – should always return `false`
- `isModifyRoleSupported()` – should always return `false`

The remaining methods must throw the following in their respective implementations:

`oracle.security.idm.OperationNotSupportedException`

25.10.3.5 `oracle.security.idm.spi.AbstractUserManager`

"[The User and Role SPI Reference](#)" provides details about the methods that need to be implemented in this class. Only the following methods need concrete/actual implementations:

- `authenticateUser(User, char[])`
- `authenticateUser(String, char[])`
- `isDropUserSupported()` – should always return `false`
- `isCreateUserSupported()` – should always return `false`
- `isModifyUserSupported()` – should always return `false`

The remaining methods must throw the following in their respective implementations:

`oracle.security.idm.OperationNotSupportedException`

25.10.3.6 `oracle.security.idm.spi.AbstractRoleProfile`

`oracle.security.idm.spi.AbstractRoleProfile` is an abstract class that can be used to return a detailed role profile.

"[The User and Role SPI Reference](#)" provides details about the methods that need to be implemented in this class. Only the following methods need concrete/actual implementations:

- `getDisplayName()`
- `getGUID()`
- `getName()`

- `getUniqueName()`
- `getPrincipal()`
- `getDescription()`
- `getGrantees()`
- `getManagers()`
- `getOwners()`
- `getProperty()` - If requested property is not set/valid for corresponding role then null should be returned as value.
- `isApplicationRole()` - must always return false
- `isEnterpriseRole()` - must always return false
- `isSeeded()` - must always return false
- `getRoleProfile()` – should return reference to current object.

The remaining methods must throw the following in their respective implementations:

`oracle.security.idm.OperationNotSupportedException`

25.10.3.7 oracle.security.idm.spi.AbstractUserProfile

`oracle.security.idm.spi.AbstractUserProfile` is an abstract class that can be used to return a detailed user profile.

"[The User and Role SPI Reference](#)" provides details about the methods that need to be implemented in this class. Only the following methods need concrete/actual implementations:

- `getDisplayName()`
- `getGUID()`
- `getName()`
- `getUniqueName()`
- `getPrincipal()`
- `getProperty()` - If the requested property is not set/valid for corresponding role then a null value must be returned.
- `getProperties()` – If the requested property is not set/valid for the corresponding user, then a null value must be returned.
- `getAllUserProperties()` – Only the properties set for the corresponding user should be returned.
- `getReportees()`
- `getManagementChain()`
- `getUserProfile()` – must return reference to current object.

These two methods:

- `setProperty()`
- `setProperties()`

must throw the following in their implementation:

```
oracle.security.idm.OperationNotSupportedException
```

25.10.3.8 oracle.security.idm.spi.AbstractSimpleSearchFilter

`oracle.security.idm.spi.AbstractSimpleSearchFilter` is an abstract class that can be extended to implement a simple search filter.

The implementing class must have a constructor that calls the constructor of the abstract class:

```
AbstractSimpleSearchFilter (  
    String attrname, int type, Object value)
```

"[The User and Role SPI Reference](#)" provides details about the methods that need to be implemented in this class. Only the following methods need concrete/actual implementations:

- `getNativeRepresentation()` – convert filter into the native representation to be used with the underlying identity repository.
- `getWildcardChar()` – wild card character, for example "*", to be used in searches. The specific character depends on the underlying identity repository.

25.10.3.9 oracle.security.idm.spi.AbstractComplexSearchFilter

`oracle.security.idm.spi.AbstractComplexSearchFilter` is an abstract class that can be extended to implement a search filter of any complexity.

The implementing class must have a constructor that calls the constructor of the abstract class:

```
AbstractComplexSearchFilter (  
    oracle.security.idm.SearchFilter[] filters, int oper_type)
```

"[The User and Role SPI Reference](#)" provides details about the methods that need to be implemented in this class. Only the following methods need concrete/actual implementations:

- `getNativeRepresentation()` – convert the filter into the native representation to be used with the underlying identity repository.

25.10.3.10 oracle.security.idm.spi.AbstractSearchResponse

The `SearchResponse` object contains search results being returned from a repository. Each result entry corresponds to one user or role in the underlying identity repository, represented by the corresponding `UserProfile/RoleProfile` class implementation.

The `SearchResponse` object must return one or more results. This means that the `hasNext()` method must return `TRUE` at least once.

Do not use if there are zero results to return. When no results are to be returned, the corresponding search API should throw the following exception:

```
oracle.security.idm.ObjectNotFoundException
```

25.10.4 Developing a Full-Featured Provider

The full-featured provider implements all the functionality supported by a read-only provider, and additionally supports CRUD operations. This requires that the CRUD APIs be implemented in the SPI implementation classes.

In the read-only provider, these APIs were implemented simply by throwing an `OperationNotSupportedException` (see the class descriptions in [Section 25.10.3, "Developing a Read-Only Provider"](#)).

For a full-featured provider, this needs to be replaced by concrete/actual implementation of the corresponding CRUD operations.

25.10.5 Development Guidelines

This section provides some guidelines for developing providers.

Mapping of Names

Be aware of the usage of naming constants such as `UserProfile.NAME`, `UNIQUE_NAME`, `UserProfile.USER_NAME`, `UserProfile.USER_ID`.

- `NAME` – name of the user or role in the underlying repository.
- `UNIQUE_NAME` – Complete name with which the user or role is represented in the underlying repository.
- `USER_NAME` – login ID of the user in the underlying repository.
- `USER_ID` – always same as `USER_NAME` constant mapping.

Depending on the identity repository, these constants might map to the same underlying identity repository attribute or they might map to different attributes. If the underlying repository is an LDAP v3 server, the mappings are as follows:

- `NAME` – mapped to naming attribute of user/group entry, for example "cn"
- `UNIQUE_NAME` - mapped to "DN" of user/group entry
- `USER_NAME/USER_ID` – mapped to login attribute, for example "uid" or "mail"

Thread Safety

The following objects are likely to be shared among multiple threads:

- `IdentityStoreFactory`,
- `IdentityStore`,
- `UserManager`,
- `RoleManager`

You should ensure that there are no thread safety-related issues in the corresponding implementation classes of your provider.

25.10.6 Testing and Verification

The User and Role API ships with a test suite to enable you to test the basic operations of providers that you develop.

The test suite can be used to test both read-only and full-featured providers.

Usage

```
java oracle.security.idm.tests.SPITest propertiesfile
```

where *propertiesfile* contains the provider class name and any configuration data for the provider. It also contains information about the tests to be run.

You need to edit this file and update it with correct information before running the tests; the file contents are self-explanatory.

One such file (`ffprovider.properties`) is available with the sample provider discussed in [Section 25.10.7.1, "About the Sample Provider"](#).

Results

The test will produce the results on-screen. All providers that you develop must pass the "Lookup tests", "Role membership tests" and "Profile tests" in the test suite. Full-featured providers must pass all the tests in the suite including Create/Drop tests.

The log of test results will be output to the file `results.out` in current working directory.

25.10.7 Example: Implementing an Identity Provider

The distribution includes a sample identity provider that you can use to understand how custom providers are built.

This section describes how to access the sample provider, and explains the steps needed to implement a custom provider. The steps rely on the sample for illustration.

- [About the Sample Provider](#)
- [Overview of Implementation](#)
- [Configure `jps-config.xml` to use the Sample Identity Provider](#)
- [Configure Oracle WebLogic Server](#)

25.10.7.1 About the Sample Provider

The sample provider is bundled in `sampleprovider.zip`. Unzip the file. It should generate the following structure:

```
sampleprovider/  
  build.xml - ant build file  
  ffprovider.properties - properties file required for testing  
  jlib - provider jar file location  
  out - location of generated class files  
  samples - Folder for samples  
  src - provider source code
```

Run `ant help` for instructions on building and testing this provider.

The provider relies on an ad-hoc identity repository for fetching identity information and has been tested with Oracle SOA Suite. It is not intended for production use without appropriate testing for your environment.

25.10.7.2 Overview of Implementation

The sample identity provider used in this example is a custom Identity/Authentication provider that uses an RDBMS as the underlying store. It can be used as both an identity provider and an authentication provider.

Note: The sample provider is intended solely for demonstration purposes, and it is not advisable to use this provider in production without exhaustive testing.

These steps are required to set up the sample provider:

1. Implement the User and Role APIs to access the database repository serving as the identity store. This involves:
 - a. Building the sample provider. Run `ant help` for instructions.
 - b. Creating the identity store schema in the database.
2. Configure the sample provider as the identity store, as shown in [Section 25.10.7.3, "Configure jps-config.xml to use the Sample Identity Provider"](#).
3. Set up Weblogic Authenticator to use this provider as `SQLAuthenticator`, as explained in [Section 25.10.7.4, "Configure Oracle WebLogic Server"](#).

25.10.7.3 Configure jps-config.xml to use the Sample Identity Provider

Configure `jps-config.xml` as follows to enable the sample identity provider to be used as the identity store:

1. Add a new provider in the service providers list:

```
<serviceProviders>
    .....
    <serviceProvider type="IDENTITY_STORE" name="custom.provider"
class="oracle.security.jps.internal.idstore.generic.GenericIdentityStoreProvide
r">
        <description>Custom IdStore Provider</description>
    </serviceProvider>
</serviceProviders>
```

2. Add the service instance:

```
<serviceInstances>
    .....
    <serviceInstance name="idstore.custom" provider="custom.provider"
        location="dumb">
        <description>Custom Identity Store Service Instance</description>
        <property name="idstore.type" value="CUSTOM"/>
        <property name="ADF_IM_FACTORY_CLASS"
            value="custom_provider_identityStoreFactoryClassName"/>
        <property name="DB_SERVER_NAME" value="db_server_name"/>
        <property name="DB_SERVER_PORT" value="db_port"/>
        <property name="DB_DATABASE_NAME" value="db_service_name"/>
        <property name="ST_SECURITY_PRINCIPAL" value="user_name"/>
        <property name="ST_SECURITY_CREDENTIALS" value="password"/>
    </serviceInstance>
    .....
</serviceInstances>
```

Note: `custom_provider_identityStoreFactoryClassName` for the sample provider is `org.sample.providers.db.DBIdentityStoreFactory`

3. Ensure that the default `jpsContext` points to the identity store service instance added in Step 2 above:

```
<jpsContext name="default">
  <serviceInstanceRef ref="credstore"/>
  <serviceInstanceRef ref="keystore"/>
  <serviceInstanceRef ref="policystore.xml"/>
  <serviceInstanceRef ref="audit"/>
  <serviceInstanceRef ref="idstore.custom"/>
</jpsContext>
```

4. Add the path of the custom provider jar to the classpath.
5. Restart the server.

25.10.7.4 Configure Oracle WebLogic Server

The final task is to configure Oracle WebLogic Server to use `SQLAuthenticator`. The steps are as follows:

1. Log in to the Oracle WebLogic Server console. Select **Security Realms**, then **myrealm**, then **Providers**. Click **New** to add a new provider.
2. Enter a name for the provider and select `SQLAuthenticator` as the authenticator type.
3. On the **Providers** page, click on the newly created authenticator.
4. Set the Control Flag to `SUFFICIENT`. Click **Save**.
5. Set the control flag to sufficient for all authenticators in the list.
6. Click on the "Provider Specific" tab to enter the details for the authenticator server. Enter the `DataSource` name that was used to create the schema for the provider. Click **Save**.
7. Return to the Providers tab and reorder the providers so that `SQLAuthenticator` is at the top of the list.

The User and Role SPI Reference

This section contains the User and Role SPI reference (Javadoc), describing each abstract class in the SPI with package name `oracle.security.idm.spi`. The classes are:

- `oracle.security.idm.spi.AbstractUserProfile`
- `oracle.security.idm.spi.AbstractUserManager`
- `oracle.security.idm.spi.AbstractUser`
- `oracle.security.idm.spi.AbstractSubjectParser`
- `oracle.security.idm.spi.AbstractStoreConfiguration`
- `oracle.security.idm.spi.AbstractSimpleSearchFilter`
- `oracle.security.idm.spi.AbstractSearchResponse`
- `oracle.security.idm.spi.AbstractRoleProfile`
- `oracle.security.idm.spi.AbstractRoleManager`
- `oracle.security.idm.spi.AbstractRole`
- `oracle.security.idm.spi.AbstractIdentityStoreFactory`
- `oracle.security.idm.spi.AbstractIdentityStore`
- `oracle.security.idm.spi.AbstractComplexSearchFilter`

oracle.security.idm.spi.AbstractUserProfile

This class represents a detailed user profile and enables you to set or obtain attributes of the user profile.

Constructors

```
public AbstractUserProfile()
```

Methods

```
public void setPassword(char[] oldPasswd, char[] newPasswd)
public byte[] getUserCertificate()
public void setUserCertificate(byte[] cert)
public java.lang.String getEmployeeNumber()
public void setEmployeeNumber(String employeeNumber)
public java.lang.String getBusinessPostalAddr()
public void setBusinessPostalAddr(String addr)
public java.lang.String getBusinessPOBox()
public void setBusinessPOBox(String pobox)
public byte[] getJPEGPhoto()
public void setJPEGPhoto(String imgpath)
public java.lang.String getTimeZone()
public void setTimeZone(String zone)
public java.lang.String getDescription()
public void setDescription(String desc)
public java.lang.String getDepartmentNumber()
public void setDepartmentNumber(String departmentnumber)
public java.lang.String getGivenName()
public void setGivenName(String givenname)
public java.lang.String getBusinessEmail()
public void setBusinessEmail(String email)
public java.lang.String getBusinessPager()
public void setBusinessPager(String pager)
public java.lang.String getOrganization()
public void setOrganization(String org)
public void setName(String name)
public java.lang.String getBusinessCity()
public void setBusinessCity(String city)
public java.lang.String getMaidenName()
public void setMaidenName(String maidenname)
public java.lang.String getDepartment()
public void setDepartment(String dept)
public java.lang.String getBusinessFax()
public void setBusinessFax(String fax)
public java.lang.String getUsername()
public void setUsername(String uname)
public java.lang.String getBusinessMobile()
public void setBusinessMobile(String mobile)
public java.lang.String getDateofHire()
public void setDateofHire(String hiredate)
public java.lang.String getTitle()
public void setTitle(String title)
public java.lang.String getNameSuffix()
public void setNameSuffix(String suffix)
public java.lang.String getMiddleName()
public void setMiddleName(String middlename)
public java.lang.String getHomePhone()
```

```
public void setHomePhone(String homephone)
public void setDisplayName(String dispname)
public java.lang.String getEmployeeType()
public void setEmployeeType(String emptytype)
public java.lang.String getLastName()
public void setLastName(String lastname)
public java.lang.String getDateofBirth()
public void setDateofBirth(String dob)
public java.lang.String getManager()
public void setManager(String manager)
public java.lang.String getBusinessState()
public void setBusinessState(String state)
public java.lang.String getHomeAddress()
public void setHomeAddress(String homeaddr)
public java.lang.String getBusinessStreet()
public void setBusinessStreet(String street)
public java.lang.String getBusinessPostalCode()
public void setBusinessPostalCode(String postalcode)
public java.lang.String getInitials()
public void setInitials(String initials)
public java.lang.String getUserID()
public void setUserID(String userid)
public java.lang.String getFirstName()
public void setFirstName(String firstname)
public java.lang.String getDefaultGroup()
public void setDefaultGroup(String defgroup)
public java.lang.String getOrganizationalUnit()
public void setOrganizationalUnit(String ouUnit)
public java.lang.String getWirelessAcctNumber()
public void setWirelessAcctNumber(String wireles sacct)
public java.lang.String getBusinessPhone()
public void setBusinessPhone(String phone)
public java.lang.String getBusinessCountry()
public void setBusinessCountry(String country)
public java.lang.String getPreferredLanguage()
public void setPreferredLanguage(String language)
public java.lang.String getUIAccessMode()
public void setUIAccessMode(String accessMode)
public java.lang.Object getPropertyVal(String prop)
public oracle.security.idm.SearchResponse getReportees(boolean direct)
public java.util.List getManagementChain(int max, String upToManagerName, String
upToTitle)
public oracle.security.idm.PropertySet getAllUserProperties()
```

oracle.security.idm.spi.AbstractUserManager

This class represents a user manager and includes basic authentication methods.

Constructors

```
public AbstractUserManager()
```

Methods

```
public oracle.security.idm.User authenticateUser(  
    String user_id, String authProperty, char[] passwd)
```

```
public oracle.security.idm.User authenticateUser(  
    User user, char[] passwd)
```

oracle.security.idm.spi.AbstractUser

This class represents a user.

Constructors

```
public AbstractUser()
```

Methods

None.

oracle.security.idm.spi.AbstractSubjectParser

This abstract class provides a constructor for a subject parser.

Constructors

```
public AbstractSubjectParser()
```

Methods

None

oracle.security.idm.spi.AbstractStoreConfiguration

This abstract class provides a constructor for identity store configuration.

Constructors

```
public AbstractStoreConfiguration()
```

Methods

None

oracle.security.idm.spi. AbstractSimpleSearchFilter

This abstract class represents a simple search filter that can be used to search the identity store. Each simple filter consists of a search attribute, matching operator type, and value. Search results are filtered based on this condition.

This class is abstract as its actual underlying representation (provided by method `@link #getNativeRepresentation()`) is implementation-specific. A service provider can extend this class by setting up a specific implementation of that method.

Constructors

```
public AbstractSimpleSearchFilter(
    String attrname, int type, Object value)
```

Methods

[Table 25–6](#) lists the methods of `AbstractSimpleSearchFilter`.

Table 25–6 *Methods of AbstractSimpleSearchFilter*

Method	Description
public void setAttribute(String name)	Set attribute name. .
public void setType(int type)	Set filter type.
public void setValue(Object value)	Set attribute value.
public java.lang.String getAttributeName()	Retrieve attribute name.
public java.lang.Object getValue()	Retrieve attribute value.
public int getType()	Retrieve filter type.
public void setNegate()	Negate the current NOT state of the search filter. Behaves like a toggle switch.
public void negate()	Negate the current NOT state of the search filter. Behaves like a toggle switch.
public boolean isNegated()	Return the current NOT state of the search filter. Returns <code>true</code> if the NOT operator is set; <code>false</code> otherwise.

oracle.security.idm.spi.AbstractSearchResponse

This is an abstract class that represents search response results.

Constructors

```
public AbstractSearchResponse()
```

Methods

None.

oracle.security.idm.spi.AbstractRoleProfile

This class represents the detailed profile of a role.

Constructors

```
public AbstractRoleProfile()
```

Methods

```
public oracle.security.idm.SearchResponse getManagers(  
    SearchFilter filter, boolean direct)
```

```
public oracle.security.idm.SearchResponse getManagers(  
    SearchFilter filter)
```

```
public oracle.security.idm.SearchResponse getOwners(  
    SearchFilter filter, boolean direct)
```

```
public oracle.security.idm.SearchResponse getOwners(  
    SearchFilter filter)
```

```
public void addManager(  
    Principal principal)
```

```
public void removeManager(  
    Principal principal)
```

```
public void addOwner(  
    Principal principal)
```

```
public void removeOwner(  
    Principal principal)
```

```
public boolean isOwnedBy(  
    Principal principal)
```

```
public boolean isManagedBy(  
    Principal principal)
```

```
public void addOwner(  
    User user)
```

```
public void removeOwner(  
    User user)
```

```
public void setDisplayName(  
    String displayName)
```

```
public void setDescription(  
    String discription)
```

```
public java.lang.String getDescription()
```

```
public oracle.security.idm.Property getProperty(  
    String propName)
```

oracle.security.idm.spi.AbstractRoleManager

This class is an abstract representation of a role manager.

Constructors

```
public AbstractRoleManager()
```

Methods

```
public boolean isOwnedBy(  
    Role role, Principal principal)
```

```
public boolean isManagedBy(  
    Role role, Principal principal)
```

```
public oracle.security.idm.SearchResponse getOwnedRoles(  
    Principal principal, boolean direct)
```

```
public oracle.security.idm.SearchResponse getManagedRoles(  
    Principal principal, boolean direct)
```

oracle.security.idm.spi.AbstractRole

This class provides a constructor for a role.

Constructors

```
public AbstractRole()
```

Methods

None

oracle.security.idm.spi.AbstractIdentityStoreFactory

This class represents an identity store factory.

See Also: "IdentityStoreFactory" in [Table 25-1](#).

Constructors

```
public AbstractIdentityStoreFactory()
```

Methods

```
public oracle.security.idm.IdentityStore getIdentityStoreInstance()
```

oracle.security.idm.spi.AbstractIdentityStore

This abstract class represents an identity store.

Constructors

```
public AbstractIdentityStore()
```

Methods

```
public oracle.security.idm.RoleManager getRoleManager() public  
oracle.security.idm.UserManager getUserManager() public java.util.List  
getMandatoryUserPropertyNames() public java.util.List getUserPropertySchema()
```


oracle.security.idm.spi.AbstractComplexSearchFilter

This class represents a complex search filter. This type of search filter is used to combine multiple SearchFilter instances with a single boolean AND or OR operator. Each of the component search filters can itself be a complex filter, enabling you to form nested search filters with a high degree of complexity.

This class is abstract in that its actual underlying representation, provided by the @link #getNativeRepresentation() method, is implementation-specific.

A service provider can extend this class by creating a specific implementation of this method.

See Also: ["oracle.security.idm.spi. AbstractSimpleSearchFilter"](#)

Constructors

```
public AbstractComplexSearchFilter(SearchFilter[] filters, int oper_type)
```

Methods

Table 25–7 *Methods of Complex Search Filter*

Method	Description
public void addFilterComponent(SearchFilter filter)	Add the SearchFilter component to this complex filter's list.
public void setNegate()	Negate the current NOT state of the search filter. Behaves like a toggle switch.
public void negate()	Negate the current NOT state of the search filter. Behaves like a toggle switch.
public boolean isNegated()	Return the current NOT state of the search filter. Returns <code>true</code> if the NOT operator is set; <code>false</code> otherwise.
public int getOperatorType()	Logical operator type which binds together the SearchFilter components.

Developing with the Identity Directory API

This chapter explains how to access and work with identity stores using the Identity Directory API.

This chapter contains these topics:

- [About the Identity Directory API](#)
- [Summary of Classes](#)
- [Identity Directory Configuration](#)
- [Working with the Identity Directory API](#)
- [Examples of Identity Directory API](#)
- [SSL Configuration](#)

26.1 About the Identity Directory API

The Identity Directory API allows applications to access identity information (users and other entities) in a uniform and portable manner regardless of the particular underlying identity repository.

The Identity Directory API:

- is flexible
- is fully configurable by clients supporting a variety of identity stores having standard and specific schemas
- supports retrieving and managing users, groups, and organizations
- is extensible, supporting new entity types with relationships defined between those entities
- is robust, with high-availability/fail-over support.

The Identity Directory API uses the Identity Governance framework, providing all the benefits of the framework to enable you to control how identity related information, including Personally Identifiable Information (PII), access entitlements, attributes, and other entities are used, stored, and propagated between organizations.

26.1.1 Feature Overview

This section explains the features supported by the Identity Directory API.

Features for User Entities

The following features are supported for users:

- Perform Create/Update/Delete (CRUD) operations on users
- Perform the following actions:
 - get and set user attributes
 - authenticate the user with the identity store’s native authentication mechanism
 - get the group to which the user belongs (optionally, including nested groups)
 - make user a member of a group
- Change user password
- Force user password change
- Get and set user state (enable/disable, lockout, password must change)

Features for Group Entities

The following features are supported for groups:

- Perform CRUD operations on groups
- Perform the following actions:
 - get and set attributes
 - get and search for members of a group
 - get the groups to which a groups belongs (optionally, including nested groups)
 - determine if the group is a member of another group
- Support multi-valued attributes
- Support static and dynamic groups

26.2 Summary of Classes

Table 26–1 lists the classes in the Identity Directory API:

Table 26–1 *Classes in the Identity Directory API*

Class	Description
Capabilities	Contains an entity’s capabilities.
CreateOptions	Contains options for entity creation operations.
DeleteOptions	Contains options for entity deletion operations.
Entity	Generic entity class holding the list of attributes of the entity fetched using search or read methods.
EntityAlreadyExistsException	Returned following an attempt to create an existing entity.
EntityCapabilities	
EntityManager	Handles operations like read, create and search of generic entity.
EntityNotFoundException	Returned when requested entity is not found.
EntityNotUniqueException	Returned when the entity is not uniquely defined.
EntityRelationManager	Handles entity relationship operations like read, create, delete, search relationship.
Group	A generic entity class holding the list of members of the group. It also provides methods to modify group membership.

Table 26–1 (Cont.) Classes in the Identity Directory API

Class	Description
GroupManager	Handles operations like creating, deleting, and searching for groups.
IDSEException	Handles exceptions.
IDSPrincipal	Contains the principal related to the exception.
IdentityDirectory	Represents a handle to IdentityDirectory for creation of IdentityDirectory instance. The instance provides handles to User, Group, and generic Entity Manager so that operations on the corresponding entities can be performed.
IdentityDirectoryFactory	A factory class for creating IdentityDirectoryService.
IdentityDirectoryInfo	
InvalidAttributesException	Used for exceptions related to invalid entity attributes.
InvalidFilterException	Used for exceptions generated within Identity Beans
ModAttribute	
ModifyOptions	Extends OperationOptions containing options for entity modify operation
OperationNotSupportedException	Used for exceptions generated within Identity Beans
ReadOptions	Extends OperationOptions containing options for entity read operation. Read options include Locale and Requested Attributes settings.
ResultSet	An interface for the object returned by search interaction with paged results.
SearchFilter	Used to construct simple or complex nested search filters for searching the entities
SearchOptions	Extends ReadOptions containing options for entity search operation.
User	Generic class for User entities.
UserCapabilities	Contains user capability attributes.
UserManager	Contains methods for creating, deleting, and searching for users by various criteria.

26.3 Identity Directory Configuration

The identity directory configuration is a combination of the logical entity configuration and the physical identity store configuration.

The identity directory with logical entity configuration is stored in:

```
DOMAIN_HOME/config/fmwconfig/ids-config.xml
```

The physical identity store configuration for the default identity directory is located at:

```
DOMAIN_HOME/config/fmwconfig/ovd/default
```

The default identity directory uses the same identity store properties (namely host, port, credentials, search base, create base, name attribute, and so on) configured in OPSS (weblogic authenticator or in `jps-config.xml`). For more information, see [Section F.2.3, "LDAP Identity Store Properties"](#).

26.4 Working with the Identity Directory API

This section explains how applications can use the Identity Directory API to view and manage identity store data. It contains these sections:

- [Getting an Identity Directory API Instance](#)
- [Performing CRUD Operations on Users and Groups](#)

26.4.1 Getting an Identity Directory API Instance

You can obtain the identity directory handle from the jps-context and get a directory instance as follows:

```
JpsContextFactory ctxFactory = JpsContextFactory.getContextFactory();
JpsContext ctx = ctxFactory.getContext();

//find the JPS IdentityStore service instance
IdentityStoreService idstoreService =
ctx.getServiceInstance(IdentityStoreService.class)

//get the Identity Directory instance
oracle.igf.ids.IdentityDirectory ids = idstoreService.getIdentityStore();
```

26.4.2 Performing CRUD Operations on Users and Groups

You can perform Create, Retrieve, Update, and Delete (CRUD) operations on users and groups.

- [User Operations](#)
- [Group Operations](#)

26.4.2.1 User Operations

Basic CRUD operations on users are as follows:

Create User

```
Principal UserManager.createUser(List<Attribute> attrVals, CreateOptions opts)
```

Get User

```
User UserManager.getUser(Principal principal, ReadOptions opts)
```

Search for User

```
User UserManager.searchUser(String id, ReadOptions opts)
```

Delete User

```
void UserManager.deleteUser(String id, DeleteOptions opts)
```

Update User

```
void UserManager.modify(List<ModAttribute> attrVals, ModifyOptions opts)
```

Retrieve List of Users

```
ResultSet UserManager.searchUsers(SearchFilter filter, SearchOptions opts)
```

26.4.2.2 Group Operations

Basic CRUD operations on groups are as follows:

Create Group

```
Principal GroupManager.createGroup(List<Attribute> attrVals, CreateOptions opts)
```

Get Group

```
User GroupManager.getGroup(Principal principal, ReadOptions opts)
```

Search for Group

```
User GroupManager.searchGroup(String id, ReadOptions opts)
```

Delete Group

```
void GroupManager.deleteGroup(String id, DeleteOptions opts)
```

Modify Group Attributes

```
void GroupManager.modify(List<ModAttribute> attrVals, ModifyOptions opts)
```

Retrieve List of Groups

```
ResultSet GroupManager.searchGroups(SearchFilter filter, SearchOptions opts)
```

26.5 Examples of Identity Directory API

This section contains the following examples of using the Identity Directory API:

- [Initialize and Obtain Identity Directory Handle](#)
- [Create a User](#)
- [Get a User](#)
- [Modify a User](#)
- [Simple Search for a User](#)
- [Complex Search for Users](#)
- [Create a Group](#)
- [Get a Group](#)
- [Get Group Using a Search Filter](#)
- [Delete a Group](#)
- [Add a Member to a Group](#)
- [Delete a Member from a Group](#)

26.5.1 Initialize and Obtain Identity Directory Handle

This sample code initializes and obtains a handle to the identity directory:

```
/**
 * This is a sample program for initializing Identity Directory Service with the
 * configuration
 * that is already persisted in IDS config location.
 * Basic User and Group CRUDS are performed using this IDS instance
 */

import java.util.ArrayList;
import java.util.List;
import java.util.Iterator;
import java.util.Map;
```

```
import java.security.Principal;

import oracle.igf.ids.Entity;
import oracle.igf.ids.User;
import oracle.igf.ids.UserManager;
import oracle.igf.ids.Group;
import oracle.igf.ids.GroupManager;
import oracle.igf.ids.config.OperationalConfig;
import oracle.igf.ids.IdentityDirectoryFactory;
import oracle.igf.ids.IdentityDirectoryInfo;
import oracle.igf.ids.IdentityDirectory;
import oracle.igf.ids.IDSEException;
import oracle.igf.ids.ReadOptions;
import oracle.igf.ids.CreateOptions;
import oracle.igf.ids.ModifyOptions;
import oracle.igf.ids.DeleteOptions;
import oracle.igf.ids.SearchOptions;
import oracle.igf.ids.SearchFilter;
import oracle.igf.ids.ResultSet;
import oracle.igf.ids.Attribute;
import oracle.igf.ids.ModAttribute;

import oracle.dms.context.ExecutionContext;

public class Ids1Test {

    private IdentityDirectory ids;
    private UserManager uMgr;
    private GroupManager gMgr;

    /**
     * Get Identity Store Service
     */
    public Ids1Test() throws IDSEException {

        // Set Operational Config
        OperationalConfig opConfig = new OperationalConfig();

        // Set the application credentials: this overrides the credentials
        // set in physical ID store configuration
        //opConfig.setApplicationUser("cn=venkat_medam,l=amer,dc=oracle,dc=com");
        //opConfig.setApplicationPassword("welcome123".toCharArray());

        // Set search/create base, name, objclass, etc. config.
        // This overrides default operational configuration in IDS
        opConfig.setEntityProperty("User", opConfig.SEARCH_BASE,
            "l=amer,dc=oracle,dc=com");
        opConfig.setEntityProperty("User", opConfig.CREATE_BASE,
            "l=amer,dc=oracle,dc=com");
        opConfig.setEntityProperty("User", opConfig.FILTER_OBJCLASSES, "person");
        opConfig.setEntityProperty("User", opConfig.CREATE_OBJCLASSES,
            "inetorgperson");
        opConfig.setEntityProperty("Group", opConfig.SEARCH_BASE,
            "cn=dlcontainerOCS,dc=oracle,dc=com");
        opConfig.setEntityProperty("Group", opConfig.CREATE_BASE,
            "cn=dlcontainerOCS,dc=oracle,dc=com");
        opConfig.setEntityProperty("Group", opConfig.FILTER_OBJCLASSES,
```



```

    "groupofuniquenames");
opConfig.setEntityProperty("Group", opConfig.CREATE_OBJCLASSES,
    "groupofuniquenames,orclgroup");

// Get IdentityDirectoryService "userrole" configured in IDS config
IdentityDirectoryFactory factory = new IdentityDirectoryFactory();
//ids = factory.getDefaultIdentityDirectory(opConfig);
ids = factory.getIdentityDirectory("userrole", opConfig);

// Get UserManager and GroupManager handles
uMgr = ids.getUserManager();
gMgr = ids.getGroupManager();
}

```

26.5.2 Create a User

This sample code creates a user in the identity store:

```

public Principal createUser() {
    Principal principal = null;

    List<Attribute> attrs = new ArrayList<Attribute>();
    attrs.add(new Attribute("commonname", "test1_user1"));
    attrs.add(new Attribute("password", "welcome123".toCharArray()));
    attrs.add(new Attribute("firstname", "test1"));
    attrs.add(new Attribute("lastname", "user1"));
    attrs.add(new Attribute("mail", "test1.user1@oracle.com"));
    attrs.add(new Attribute("telephone", "1 650 123 0001"));
    attrs.add(new Attribute("title", "Senior Director"));
    attrs.add(new Attribute("uid", "tuser1"));
    // Adding locale specific value
    attrs.add(new Attribute("description", "created test user 1",
        new java.util.Locale("us", "en")));
    try {
        CreateOptions createOpts = new CreateOptions();
        createOpts.setCreateBase("l=apac,dc=oracle,dc=com");

        principal = uMgr.createUser(attrs, createOpts);

        System.out.println("Created user " + principal.getName());
    } catch (Exception e) {
        System.out.println(e.getMessage());
        e.printStackTrace();
    }

    return principal;
}

```

26.5.3 Get a User

This sample code obtains a user from the identity store:

```

public User getUser(Principal principal) {
    User user = null;

    try {
        ReadOptions readOpts = new ReadOptions();
        // Getting specific locale values
    }
}

```

```
        readOpts.setLocale("us-en");

        user = uMgr.getUser(principal, readOpts);

        printEntity(user);

    } catch (Exception e) {
        System.out.println(e.getMessage());
        e.printStackTrace();
    }

    return user;
}
```

26.5.4 Modify a User

This sample code modifies an existing user by adding a new user attribute:

```
public void modifyUser(User user) {

    try {
        ModifyOptions modifyOpts = new ModifyOptions();

        List<ModAttribute> attrs = new ArrayList<ModAttribute>();
        attrs.add(new ModAttribute("description", "modified test user 1"));
        //attrs.add(new ModAttribute("uid", "venkatmedam"));

        user.modify(attrs, modifyOpts);

        System.out.println("Modified user " + user.getName());

    } catch (Exception e) {
        System.out.println(e.getMessage());
        e.printStackTrace();
    }
}
```

26.5.5 Simple Search for a User

This sample code performs a basic user search:

```
try {

    ReadOptions readOpts = new ReadOptions();
    readOpts.setSearchBase("l=apac");

    User user = uMgr.searchUser("tuser1", readOpts);

    printEntity(user);

} catch (Exception e) {
    System.out.println(e.getMessage());
    e.printStackTrace();
}
}
```

26.5.6 Complex Search for Users

This sample code uses a complex search filter to return matching users:

```

public void searchUsers() {

    try {
        // Complex search filter with nested AND and OR conditions
        SearchFilter filter = new SearchFilter(
            SearchFilter.LogicalOp.OR,
            new SearchFilter(SearchFilter.LogicalOp.AND,
                new SearchFilter("firstname", SearchFilter.Operator.BEGINS_WITH, "ve"),
                new SearchFilter("telephone", SearchFilter.Operator.CONTAINS, "506")),
            new SearchFilter(SearchFilter.LogicalOp.AND,
                new SearchFilter("firstname", SearchFilter.Operator.BEGINS_WITH, "ra"),
                new SearchFilter(SearchFilter.LogicalOp.OR,
                    new SearchFilter("orgunit", SearchFilter.Operator.BEGINS_WITH, "ldap"),
                    new SearchFilter("orgunit", SearchFilter.Operator.BEGINS_WITH, "sun"),
                    new SearchFilter("orgunit", SearchFilter.Operator.BEGINS_WITH,
                        "access")),
                new SearchFilter("telephone", SearchFilter.Operator.CONTAINS, "506"));

        // Requesting attributes
        List<String> reqAttrs = new ArrayList<String>();
        reqAttrs.add("jpegphoto");

        SearchOptions searchOpts = new SearchOptions();
        searchOpts.setPageSize(3);
        searchOpts.setRequestedPage(1);
        searchOpts.setRequestedAttrs(reqAttrs);
        searchOpts.setSearchBase("l=amer");

        ResultSet<User> sr = uMgr.searchUsers(filter, searchOpts);
        while (sr.hasMore()) {
            User user = sr.getNext();
            //printEntity(user);
            //System.out.println(" ");
            System.out.println(user.getSubjectName());
            System.out.println("    " + user.getAttributeValue("commonname"));
        }

    } catch (Exception e) {
        System.out.println(e.getMessage());
        e.printStackTrace();
    }
}

```

26.5.7 Create a Group

This sample code creates a group:

```

public Principal createGroup() {
    Principal principal = null;

    List<Attribute> attrs = new ArrayList<Attribute>();
    attrs.add(new Attribute("name", "test1_group1"));
    attrs.add(new Attribute("description", "created test group 1"));
    attrs.add(new Attribute("displayname", "test1_group1"));
    try {
        CreateOptions createOpts = new CreateOptions();

        principal = gMgr.createGroup(attrs, createOpts);
    }
}

```

```
        System.out.println("Created group " + principal.getName());

    } catch (Exception e) {
        System.out.println(e.getMessage());
        e.printStackTrace();
    }

    return principal;
}
```

26.5.8 Get a Group

This sample code returns a specific group:

```
public Group getGroup(Principal principal) {
    Group group = null;

    try {
        ReadOptions readOpts = new ReadOptions();

        group = gMgr.getGroup(principal, readOpts);

        printEntity(group);
    } catch (Exception e) {
        System.out.println(e.getMessage());
        e.printStackTrace();
    }

    return group;
}
```

26.5.9 Get Group Using a Search Filter

This sample code uses a search filter to return groups:

```
public void searchGroups() {

    try {
        SearchFilter filter = new SearchFilter("name",
            SearchFilter.Operator.BEGINS_WITH, "test");

        SearchOptions searchOpts = new SearchOptions();
        searchOpts.setPageSize(10);

        ResultSet<Group> sr = gMgr.searchGroups(filter, searchOpts);
        while (sr.hasMore()) {
            Group group = sr.getNext();
            System.out.println(group.getSubjectName());
        }
    } catch (Exception e) {
        System.out.println(e.getMessage());
        e.printStackTrace();
    }
}
```

26.5.10 Delete a Group

This sample code deletes a group from the store:

```
public void deleteGroup(Principal principal) {

    try {
        DeleteOptions deleteOpts = new DeleteOptions();

        gMgr.deleteGroup(principal, deleteOpts);

        System.out.println("Deleted group " + principal.getName());

    } catch (Exception e) {
        System.out.println(e.getMessage());
        e.printStackTrace();
    }
}
```

26.5.11 Add a Member to a Group

This sample code adds a member to a group:

```
public void addMember() {
    try {
        ReadOptions readOpts = new ReadOptions();
        User user = uMgr.searchUser("amsharma", readOpts);
        Group group = gMgr.searchGroup("test1_group1", readOpts);

        ModifyOptions modOpts = new ModifyOptions();
        user.addMemberOf(group, modOpts);

        System.out.println("added amsharma as member of test1_group1");

    } catch (Exception e) {
        System.out.println(e.getMessage());
        e.printStackTrace();
    }
}
```

26.5.12 Delete a Member from a Group

This sample code deletes a member from a group:

```
public void deleteMember() {
    try {
        ReadOptions readOpts = new ReadOptions();
        User user = uMgr.searchUser("amsharma", readOpts);
        Group group = gMgr.searchGroup("test1_group1", readOpts);

        ModifyOptions modOpts = new ModifyOptions();
        group.deleteMember(user, modOpts);

        System.out.println("deleted amsharma from the group test1_group1");

    } catch (Exception e) {
        System.out.println(e.getMessage());
        e.printStackTrace();
    }
}
```

}

26.6 SSL Configuration

For details about SSL configuration when using the Identity Directory API, see [Section 7.5, "SSL for the Identity Store Service"](#).

Developing with the Keystore Service

This chapter explains how to utilize the Keystore Service when developing applications.

- [About the Keystore Service API](#)
- [Overview of Application Development with the Keystore Service](#)
- [Setting the Java Security Policy Permission](#)
- [Configuring the Keystore Service](#)
- [Steps for Using the Keystore Service API](#)
- [Example of Keystore Service API Usage](#)
- [Best Practices](#)

27.1 About the Keystore Service API

A keystore is used for secure storage of and access to keys and certificates. The Keystore Service API is used to access and perform operations on the keystores.

The Keystore Service:

- enables you to manage keys and certificates securely
- provides an API for storage, retrieval, and maintenance of keys and certificates in different back-end repositories
- supports file, database, LDAP-based keystore management

Critical (create, update, delete) functions provided by the Keystore Service API include:

- creating keystores
- deleting keystores
- obtaining a handle to the domain trust store
- obtaining a handle to a keystore
- obtaining the configured properties for a keystore
- obtaining a list of the keystores within an application stripe

Operations on a `KeyStore` are secured by `KeyStoreAccessPermission`, which implements the fine-grained access control model utilized by the Keystore Service.

See Also:

- [Chapter 11, "Managing Keys and Certificates with the Keystore Service"](#).

27.2 Overview of Application Development with the Keystore Service

Knowledge of the following areas is helpful in getting your applications to work with the Keystore Service:

- Determining appropriate application stripe and keystore names to use.
- Provisioning Java security policies.

Policy permissions are set in the policy store, which can be file-based (`system-jazn-data.xml`) or LDAP-based. Setting appropriate permissions to enable application usage without compromising the security of your data requires careful consideration of permission settings.

See Also: [Section 9.1, "Managing the Policy Store"](#).

- Defining the Keystore Service instance in `jps-config.xml`.

You will need to define the service instance in `jps-config.xml` only if manually crafting the configuration file.

Note: The file-based provider is already configured by default, and can be changed to an LDAP-based provider. See [Section 8.6, "Migrating the OPSS Security Store"](#).

- Steps to take in setting up the environment.

The steps are different for stand-alone applications and those that operate in an Oracle WebLogic Server environment.

27.3 Setting the Java Security Policy Permission

The Oracle Platform Security Services keystore provider is set when the server is started. When the provider is file-based, the data is stored in `system-jazn-data.xml`.

Keystore Service supports securing keys:

- at the application stripe level,
- at the keystore level, or
- with finer granularity for specific `<application stripe, keystore, key>`

Notes:

- To properly access the Keystore Service APIs, you need to grant Java permissions in the policy store.
- The code invoking Keystore Service APIs needs code source permission. The permissions are typically for specific code jars and not for the complete application.

This section provides guidelines for permission grants to keystore objects, along with several examples:

- [Guidelines for Granting Permissions](#)
- [Permissions Grant Example 1](#)
- [Permissions Grant Example 2](#)
- [Permissions Grant Example 3](#)

Note: In the examples, the application jar file name is `AppName.jar`.

27.3.1 Guidelines for Granting Permissions

The Keystore Service relies on Java permissions to grant permissions to keystore or key objects. It is highly recommended that only the requisite permissions be granted, and no more.

WARNING: It is risky and inadvisable to grant unnecessary permissions, particularly permissions to all application stripes and/or keystores.

27.3.2 Permissions Grant Example 1

The Keystore Service stores objects in a hierarchy:

application stripe -> keystore(s) -> key(s)/certificate(s)

See Also: [Section 11.1.1](#) for details about the object hierarchy in the Keystore Service.

This example grants permissions for a specific application stripe and a specific keystore name within that stripe.

```
<jazn-policy>
  <grant>
    <grantee>
      <principals>...</principals>
      <!-- This is the location of the jar -->
      <!-- as loaded with the run-time -->
      <codesource>
        <url>file:${oracle.deployed.app.dir}/<MyApp>${oracle.deployed.app.ext}</url>
      </codesource>
    </grantee>
    <permissions>
      <permission>
        <class>oracle.security.jps.service.keystore.
```

```

        KeyStoreAccessPermission</class>
        <name>stripeName=keystoreapp,keystoreName=ks1,alias=*</name>
        <!-- All actions are granted -->
        <actions>*</actions>
    </permission>
</permissions>
</grant>
</jazn-policy>

```

where:

- stripeName is the name of the application stripe (typically the name of the application) for which you want to grant these permissions (read, write, update, and delete permissions denoted by the wildcarded actions).
- keystoreName is the key store name in use.
- alias indicates the key alias within the key store.

Note: The wildcard indicates the application is granted permission for all aliases.

27.3.3 Permissions Grant Example 2

In this example, permissions are granted for a specific application stripe name and all its keystores.

```

<jazn-policy>
  <grant>
    <grantee>
      <principals>...</principals>
      <codesource>
        <url>file:${oracle.deployed.app.dir}/<MyApp>${oracle.deployed.app.ext}</url>
      </codesource>
    </grantee>
    <permissions>
      <permission>
        <class>oracle.security.jps.service.keystore.
          KeyStoreAccessPermission</class>
        <name>stripeName=keystoreapp,keystoreName=*,alias=*</name>
        <!-- Certain actions are explicitly specified -->
        <!-- Compare to wild-card grant in previous example -->
        <actions>read,write,update,delete</actions>
      </permission>
    </permissions>
  </grant>
</jazn-policy>

```

27.3.4 Permissions Grant Example 3

In this example, read permissions are granted for a specific key alias within an application stripe name and a keystore.

```

<jazn-policy>
  <grant>
    <grantee>
      <principals>...</principals>
      <codesource>
        <url>file:${oracle.deployed.app.dir}/<MyApp>${oracle.deployed.app.ext}</url>

```

```

        </codesource>
    </grantee>
    <permissions>
        <permission>
            <class>oracle.security.jps.service.keystore.
                KeyStoreAccessPermission</class>
            <name>stripeName=keystoreapp,keystoreName=ks1,alias=orakey</name>
            <actions>read</actions>
        </permission>
    </permissions>
</grant>
</jazzn-policy>

```

27.4 Configuring the Keystore Service

You need to define the Keystore Service instance in a configuration file which contains information about the location of the keystore and the provider classes. Configuration files are located in:

```
$DOMAIN_HOME/config/fmwconfig
```

and are named as follows:

- jps-config.xml for Oracle WebLogic Server
- jps-config-jse.xml for Java SE

27.5 Steps for Using the Keystore Service API

You can use the Keystore Service within Oracle WebLogic Server or in a standalone environment.

- [Using the Keystore Service API in a Standalone Environment](#)
- [Using the Keystore Service API in Oracle WebLogic Server](#)

27.5.1 Using the Keystore Service API in a Standalone Environment

The steps for using the API in a standalone environment are:

1. Set up the classpath. Ensure that the `jps-manifest.jar` file is in your classpath. For details, see Required JAR in Classpath in [Section 1.5.3, "Scenario 3: Securing a Java SE Application"](#).
2. Set up the policy; to provide access to the Keystore Service APIs, you need to configure the access permissions in the reference policy store. For examples, see [Section 27.3, "Setting the Java Security Policy Permission"](#).
3. Run the application.

Command-line options include:

```
-Doracle.security.jps.config
specifies the full path to the configuration file
```

```
-Djava.security.policy
specifies the location of the OPSS/Oracle WebLogic Server policy file
```

```
-Djava.security.debug=all
is helpful for debugging purposes
```

27.5.2 Using the Keystore Service API in Oracle WebLogic Server

Take these steps to use the API in an Oracle WebLogic Server environment:

1. Out-of-the-box, the keystore service provider section of the `jps-config.xml` file is configured in the following directory:

```
$DOMAIN_HOME/config/fmwconfig
```

If needed, reassociate to an LDAP or database store.

2. Set up the policy. To provide access to the Keystore Service APIs, you need to configure the access permissions in the reference policy store. For examples, see [Section 27.3, "Setting the Java Security Policy Permission"](#).
3. Start Oracle WebLogic Server.
4. Deploy and test the application.

27.6 Example of Keystore Service API Usage

This section provides an example of using the key store service APIs. It contains these topics:

- [Java Program for Keystore Service Operations](#)
- [Policy Store Setup](#)
- [Configuration File](#)
- [About Using the Keystore Service in the Java SE Environment](#)

27.6.1 Java Program for Keystore Service Operations

The following Java code demonstrates common Keystore Service operations:

```
import oracle.security.jps.JpsContext;
import oracle.security.jps.JpsContextFactory;
import oracle.security.jps.JpsException;
import oracle.security.jps.internal.policystore.JavaPolicyProvider;
import oracle.security.jps.service.keystore.KeyStoreProperties;
import oracle.security.jps.service.keystore.KeyStoreService;
import oracle.security.jps.service.keystore.KeyStoreServiceException;

import java.security.AccessController;
import java.security.PrivilegedAction;

public class KeyStoreTest {

    static {
        java.security.Policy.setPolicy(new JavaPolicyProvider());
    }

    private static KeyStoreService ks = null;

    public KeyStoreTest() {
        super();
    }

    /*
     * This method performs a non-privileged operation. Either all code
     * in the call stack must have KeyStoreAccessPermission
     */
}
```

```

* OR
* the caller must have the KeyStoreAccessPermission only and
* invoke this operation in doPrivileged block
*/
public static void doKeyStoreOperation() {
    doOperation();
}

/*
* Since this method performs a privileged operation, only current class or
* jar containing this class needs KeyStoreAccessPermission
*/
public static void doPrivilegedKeyStoreOperation() {
    AccessController.doPrivileged(new PrivilegedAction<String>() {
        public String run() {
            doOperation();
            return "done";
        }
    });
}

private static void doOperation() {
    try {
        ks.deleteKeyStore("keystoreapp", "ks1", null);
    } catch(KeyStoreServiceException e) {
        e.printStackTrace();
    }
}

/*
* Since this method performs a privileged operation, only current class or
* jar containing this class needs KeyStoreAccessPermission
*/
public static void doPrivilegedKeyStoreOperation() {
    AccessController.doPrivileged(new PrivilegedAction<String>() {
        public String run() {
            doOperation();
            return "done";
        }
    });
}

private static void doOperation() {
    try {
        ks.deleteKeyStore("keystoreapp", "ks1", null);
    } catch(KeyStoreServiceException e) {
        e.printStackTrace();
    }
}

public static void main(String args[]) throws Exception {

    try {

        JpsContext ctx = JpsContextFactory.getContextFactory().getContext();
        ks = ctx.getServiceInstance(KeyStoreService.class);

        // #1 - this call is in a doPrivileged block
        // #1 - this should succeed.
        doPrivilegedKeyStoreOperation();

        // #2 - this will also pass since granted all application

```

```

        // code necessary permission
        // NOTE: Since this call is not in a doPrivileged block,
        // this call would have failed if KeyStoreAccessPermission
        // wasn't granted to this class.

        /*
        doKeyStoreOperation();
        */

    } catch (JpsException je) {
        je.printStackTrace();
    }
}
}
}

```

27.6.2 Policy Store Setup

For illustration, the example uses an xml-based policy store file (`system-jazn-data.xml`) which has the appropriate permissions needed to access the given key store from the store. The file defines the permissions for different combinations of application stripe and key store name. Other combinations, or attempts to access the store beyond the permissions defined here, will be disallowed.

Note: The default policy store to which this grant is added is `$DOMAIN_HOME/config/fmwconfig/system-jazn-data.xml`.

Here the system property `projectsrc.home` is set to point to the directory containing the Java SE application, and `clientApp.jar` is the application jar file which is present in sub-directory `dist`.

The corresponding policy grant looks like this:

```

<grant>
  <grantee>
    <codesource>
      <url>file:${projectsrc.home}/dist/clientApp.jar</url>
    </codesource>
  </grantee>
  <permissions>
    <permission>
      <class>oracle.security.jps.service.keystore.KeyStoreAccessPermission
    </class>
    <name>stripeName=keystoreapp,keystoreName=ks1,alias=*</name>
    <actions>*</actions>
  </permission>
</permissions>
</grant>

```

27.6.3 Configuration File

Here is a sample configuration file (`jps-config-jse.xml`). The `keystore.file.path` property of the keystore service shows the directory containing the `kestores.xml` file:

Note: For the complete configuration file see the default file shipped with the distribution at `$DOMAIN_HOME/config/fmwconfig/jps-config-jse.xml`.

```
<jpsConfig>
...
  <serviceInstances>
    <serviceInstance name="keystore_file_instance"
      provider="keystore_file_provider">
      <property name="keystore.file.path" value="store" />
      <property name="keystore.provider.type" value="file" />
    </serviceInstance>
  </serviceInstances>
...
</jpsConfig>
```

27.6.4 About Using the Keystore Service in the Java SE Environment

In the Java SE environment, the following calls are equivalent:

```
KeyStoreService store =
JpsServiceLocator.getServiceLocator().lookup(KeyStoreService.class);
```

and:

```
KeyStoreService store =
JpsContextFactory.getContextFactory().getContext().getServiceInstance
(KeyStoreService.class);
```

27.7 Best Practices

In a clustered environment, use the Keystore Service Mbean API over the Keystore Service API to create, retrieve, update, and delete keys for an application.

If you are simply reading keys, however, either API can be used.

Developing with the Audit Service

This chapter explains how applications (also known as audit clients) can use the Oracle Fusion Middleware Audit Framework to provide auditing capabilities. Release 5.1.2 Patch Set 5 introduces an audit service that enables you to integrate with the audit framework programmatically to log audit events and generate compliance reports using the same capabilities available to Oracle components.

Using the audit service, applications can:

- create event definitions without the use of custom tables
- register with the audit service when you deploy the application
- change event definitions when redeploying the application
- change audit configuration settings at run-time

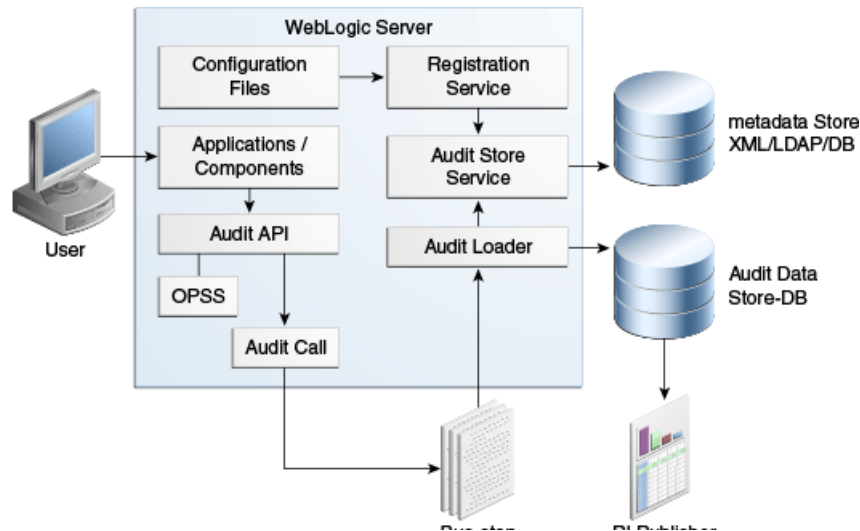
This chapter contains these topics:

- [Application Integration with Audit Flow](#)
- [Audit Metadata Model](#)
- [The Audit Metadata Store](#)
- [Integrating the Application with the Audit Framework](#)
- [Create Audit Definition Files](#)
- [Register Application with the Registration Service](#)
- [Add Application Code to Log Audit Events](#)
- [Integrate with Oracle Business Intelligence Publisher](#)
- [Update and Maintain Audit Definitions](#)

28.1 Application Integration with Audit Flow

As [Figure 28–1](#) shows, Java EE applications running on Oracle WebLogic Server can integrate with and leverage the audit framework seamlessly:

Figure 28–1 Integrating Applications with the Audit Framework



During application deployment or audit service start-up, a client such as a Java EE application or Oracle component registers with the audit service. The registration service updates the metadata store with the latest audit definitions contained in component_events.xml and related files.

See Also: [Section 12.3](#) for details about the audit flow.

The rest of this chapter explains the metadata model, and how you can integrate your applications with the audit flow to log audit events and create audit reports.

28.2 Audit Metadata Model

The audit framework supports a metadata model which enables applications to specify their audit artifacts in a flexible manner. Applications can dynamically define attribute groups, categories, and events.

28.2.1 Attribute Groups

Attribute groups provide broad classification of audit attributes and consist of three types:

- The common attribute group contains all the system attributes common to all applications, such as component type, system IP address, hostname, and others. The IAU_COMMON database table contains attributes in this group.
- Generic attribute groups contain attributes for audit application areas such as authentication and user provisioning.
- Custom attribute groups are those defined by an application to meet its specific needs. Attributes in a custom group are limited to that component scope.

28.2.1.1 Audit Attribute Data Types

[Table 28–1](#) shows the supported attribute data types and the corresponding Java object types:

Table 28–1 Audit Attribute Data Types

Attribute Data Type	Java Object Type	Notes
Integer	Integer	
Long	Long	
Float	Float	
Double	Double	
Boolean	Boolean	
DateTime	java.util.Date	
String	String	Maximum length 2048 bytes
LongString	String	Unlimited length
Binary	byte[]	

28.2.1.2 Common Attribute Groups

The common attribute group is stored in the IAU_COMMON database table.

28.2.1.3 Generic Attribute Groups

A generic attribute group is defined with a namespace, a version number, and one or more attributes. This example defines an attribute group with namespace authorization and version 1.0:

```
<AuditConfig xmlns="http://xmlns.oracle.com/ias/audit/audit-2.0.xsd" >
  <Attributes ns="authorization" version="1.0">
    <Attribute displayName="CodeSource" maxLength="2048" name="CodeSource"
type="string"/>
    <Attribute displayName="Principals" maxLength="1024" name="Principals"
type="string"/>
    <Attribute displayName="InitiatorGUID" maxLength="1024"
name="InitiatorGUID" type="string"/>
    <Attribute displayName="Subject" maxLength="1024" name="Subject"
type="string">
      <HelpText>Used for subject in authorization</HelpText>
    </Attribute>
  </Attributes>
.....
```

Your application can reference the CodeSource attribute like this:

```
<Attribute name="CodeSource" ns="authorization" version="1.0" />
```

Each generic attribute group is stored in a dedicated database table. The naming conventions are:

- IAU_GENERIC_ATTRIBUTE_GROUP_NAME for table names
- IAU_ATTRIBUTE_NAME for table columns

For example, the attribute group authorization is stored in database table IAU_AUTHORIZATION with these columns:

- IAU_CODESOURCE as string
- IAU_PRINCIPALS as string
- IAU_INITIATORGUID as string

28.2.1.4 Custom Attribute Groups

A custom attribute group is defined with a namespace, a version number, and one or more attributes.

Attributes consist of:

- attribute name
- data type
- attribute-database column mapping order, which specifies the order in which an attribute is mapped to a database column of a specific data type in the custom attribute table
- help text (optional)
- maximum length
- display name

This example defines attribute group `Accounting` with namespace `accounting` and version `1.0`:

```
<Attributes ns="accounting" version="1.0">
    <Attribute name="TransactionType" displayName="Transaction Type"
type="string" order="1"/>
    <Attribute name="AccountNumber" displayName="Account Number"
type="int" order="2">
        <HelpText>Account number.</HelpText>
    </Attribute>
    .....
</Attributes>
```

Custom attribute groups and attributes are stored in the `IAU_CUSTOM` table.

28.2.2 Event Categories and Events

An audit event category contains related events in a functional area. For example, a session category could contain login and logout events that are significant in a user session's life cycle.

An event category does not itself define attributes. Instead, it references attributes in component and system attribute groups.

There are two types of event categories:

- System Categories
- Component and Application Categories

28.2.2.1 System Categories and Events

A system category references common and generic attribute groups and contains audit events. System categories are the base set of component event categories and events. Applications can reference them directly, log audit events, and set filter preset definitions.

The following example shows several elements of the metadata model:

- common attribute group
- generic attribute groups `identity` and `authorization`

- system category `UserSession` with an attribute referencing to a common attribute `AuthenticationMethod`
- audit events such as `UserLogin` and `UserLogout`

```
<SystemComponent major="1" minor="0">
+<Attributes ns="common" version="1.0"></Attributes>
+<Attributes ns="identity" version="1.0"></Attributes>
+<Attributes ns="authorization" version="1.0"></Attributes>
-<Events>
  -<Category name="UserSession" displayName="User Sessions">
    -<Attributes>
      <Attribute name="AuthenticationMethod" ns="common" version="1.0" />
    </Attributes>
    -<HelpText></HelpText>
    -<Event name="UserLogin" displayName="User Logins" shortName="uLogin"></Event>
    -<Event name="UserLogout" displayName="User Logouts" shortName="uLogout"
      xdasName="terminateSession"></Event>
    -<Event name="Authentication" displayName="Authentication"></Event>
    -<Event name="InternalLogin" displayName="Internal Login" shortName="iLogin"
      xdasName="CreateSession"></Event>
    -<Event name="InternalLogout" displayName="Internal Logout" shortName="iLogout"
      xdasName="terminateSession"></Event>
    -<Event name="QuerySession" displayName="Query Session"
      shortName="qSession"></Event>
    -<Event name="ModifySession" displayName="Modify Session"
      shortName="mSession"></Event>
  </Category>
+<Category displayName="Authorization" name="Authorization"></Category>
+<Category displayName="ServiceManagement" name="ServiceManagement"></Category>
</Events>
</SystemComponent>
```

28.2.2.2 Component/Application Categories

A component or application can define extend system categories or define new component event categories. In this example, a transaction category references attributes `AccountNumber`, `Date`, and `Amount` from the accounting attribute group, and includes events 'purchase' and 'deposit':

```
<Category displayName="Transaction" name="Transaction">
  <Attributes>
    <Attribute name="AccountNumber" ns="accounting" version="1.0"/>
    <Attribute name="Date" ns="accounting" version="1.0" />
    <Attribute name="Amount" ns="accounting" version="1.0" />
  </Attributes>

  <Event displayName="purchase" name="purchase"/>
  <Event displayName="deposit" name="deposit">
    <HelpText>depositing funds.</HelpText>
  </Event>
.....
</Category>
```

You extend system categories by creating category references in your application audit definitions. List the system events that the system category includes, and add new attribute references and events to it.

In this example, a new category references a system category `ServiceManagement` with a new attribute reference `ServiceTime`, and a new event `restartService`:

```
<CategoryRef name="ServiceManagement" componentType="SystemComponent">
  <Attributes>
    <Attribute name="ServiceTime" ns="accounting" version="1.0" />
  </Attributes>

  <EventRef name="startService"/>
  <EventRef name="stopService"/>

  <Event displayName="restartService" name="restartService">
    <HelpText>restart service</HelpText>
  </Event>
</CategoryRef>
```

28.3 The Audit Metadata Store

The audit metadata store provides the repository for the metadata model and contains component audit definitions, NLS translation entries, runtime policies, and database mapping tables.

Note: The metadata store is separate from the audit data store.

The audit metadata store supports several critical auditing functions:

- The audit registration service creates, modifies, and deletes event definition entries.
- The audit runtime service retrieves event definitions and runtime policies.
- The audit data loader creates attribute database mappings to store audit data.
- Audit MBean commands look up and modify component audit definitions and runtime policies.

The audit framework supports three types of metadata store:

- XML file-based
- database
- LDAP

When a new application registers to the audit service, the following audit artifacts are stored in the audit store:

- audit event definitions including custom attribute group, categories, events, and filter preset definitions
- localized translation entries
- custom attribute-database column mapping table
- run-time audit policies

28.4 Integrating the Application with the Audit Framework

Take these steps to integrate your application with the audit framework:

1. Create an audit definition file, `component_events.xml`.
2. Package the `component_events.xml` file in the application EAR file.

3. Add the audit event API to the application code to enable it to log audit events.
4. Integrate with Oracle Business Intelligence Publisher for reporting.
5. Update the audit event definition and redeploy as needed.

The following sections provide more details on these tasks:

- [Create Audit Definition Files](#)
- [Register Application with the Registration Service](#)
- [Add Application Code to Log Audit Events](#)
- [Integrate with Oracle Business Intelligence Publisher](#)
- [Update and Maintain Audit Definitions](#)

28.5 Create Audit Definition Files

This task involves creating the following files:

- component_events.xml definition file
- translation files

component_events.xml File

The component_events.xml file includes these elements:

- basic properties - and the major and minor version
 - the component type, which is the property that applications use to register with the audit service and obtain runtime auditor instances
 - Major and minor version of the application.
- at most one custom attribute group
- event categories with attribute references and events
- component level filter definitions
- runtime policies, which include:
 - filterPreset - specifies the audit filter level
 - Custom FilterPresetDefinition- specifies the custom Filter Preset Definition
 - specialUsers - specifies the users to always audit
 - maxBusstopDirSize
 - maxBusstopFileSize

For details about run-time policies, see [Section 13.3](#).

Here is an example component_events.xml file:

```
<?xml version="1.0"?>
<AuditConfig xmlns="http://xmlns.oracle.com/ias/audit/audit-2.0.xsd">
  <AuditComponent componentType="ApplicationAudit" major="1" minor="0">
    <Attributes ns="accounting" version="1.0">
      <Attribute name="TransactionType" displayName="Transaction Type"
type="string" order="1">
        <HelpText>Transaction type.</HelpText>
      </Attribute>
      <Attribute name="AccountNumber" displayName="Account Number"
```

```

type="int" order="2">
    <HelpText>Account number.</HelpText>
</Attribute>
<Attribute name="Date" displayName="Date" type="dateTime" order="3"/>
<Attribute name="Amount" displayName="Amount" type="float" order="4">
    <HelpText>Transaction amount.</HelpText>
</Attribute>
<Attribute name="Status" displayName="Account Status" type="string"
order="5">
    <HelpText>Account status.</HelpText>
</Attribute>

</Attributes>

<Events>
<Category displayName="Transaction" name="Transaction">
<Attributes>
    <Attribute name="AccountNumber" ns="accounting" version="1.0"
/>
        <Attribute name="Date" ns="accounting" version="1.0" />
        <Attribute name="Amount" ns="accounting" version="1.0" />
</Attributes>

<Event displayName="purchase" name="purchase">
    <HelpText>direct purchase.</HelpText>
</Event>
<Event displayName="deposit" name="deposit">
    <HelpText>depositing funds.</HelpText>
</Event>
<Event displayName="withdrawing" name="withdrawing">
    <HelpText>withdrawing funds.</HelpText>
</Event>
<Event displayName="payment" name="payment">
    <HelpText>paying bills.</HelpText>
</Event>
</Category>
<Category displayName="Account" name="Account">
<Attributes>
    <Attribute name="AccountNumber" ns="accounting" version="1.0"
/>
        <Attribute name="Status" ns="accounting" version="1.0" />
</Attributes>

<Event displayName="open" name="open">
    <HelpText>Open a new account.</HelpText>
</Event>
<Event displayName="close" name="close">
    <HelpText>Close an account.</HelpText>
</Event>
<Event displayName="suspend" name="suspend">
    <HelpText>Suspend an account.</HelpText>
</Event>
</Category>
</Events>
<FilterPresetDefinitions>
<FilterPresetDefinition displayName="Low" helpText="" name="Low">
    <FilterCategory enabled="partial"
name="Transaction">deposit.SUCSESSESONLY(HostId -eq
'NorthEast') ,withdrawing</FilterCategory>
    <FilterCategory enabled="partial"

```



```

name="Account">open,SUCCESESONLY,close.FAILURESONLY</FilterCategory>
    </FilterPresetDefinition>
    <FilterPresetDefinition displayName="Medium" helpText=" "
name="Medium">
    <FilterCategory enabled="partial"
name="Transaction">deposit,withdrawing</FilterCategory>
    <FilterCategory enabled="partial"
name="Account">open,close</FilterCategory>
    </FilterPresetDefinition>
    <FilterPresetDefinition displayName="High" helpText=" " name="High">
    <FilterCategory enabled="partial"
name="Transaction">deposit,withdrawing,payment</FilterCategory>
    <FilterCategory enabled="true" name="Account"/>
    </FilterPresetDefinition>
</FilterPresetDefinitions>

<Policy filterPreset="Low">
    <CustomFilters>
    <FilterCategory enabled="partial"
name="Transaction">purchase</FilterCategory>
    </CustomFilters>

</Policy>

</AuditComponent>
</AuditConfig>

```

Translation Files

Create the translation files required for your application.

Translation files are used to display audit definition in different languages. Generate the files in XLIFF format; during registration, this information is stored in the audit metadata store along with the component audit event definition.

28.5.1 Understand Mapping and Versioning Rules

When creating your audit definition file, you must be aware of certain rules that the registration service uses to create the audit metadata for the application. This metadata is used to maintain different versions of the audit definition, and to load audit data and generate reports.

28.5.1.1 Version Numbers

Each audit definition must have a major and a minor version number, which are integers, for example, major = 1 minor=3. Any change to an audit event definition requires that the version ID be modified by changing the minor and/or major number.

Version numbers are used by the audit registration service to determine the compatibility of event definitions and attribute mappings between versions.

Note: These version numbers have no relation to Oracle Fusion Middleware version numbers.

Versioning for Oracle Components

When registering an Oracle component such as Oracle Virtual Directory, the audit registration service checks if this is a first-time registration or an upgrade.

For a new registration, the service:

1. retrieves the component audit and translation information.
2. parses and validates the definition, and stores it in the audit metadata store.
3. generates the attribute-column mapping table, and saves this in the audit metadata store.

For upgrades, the current major and minor numbers for the component in the metadata store are compared to the new major and minor numbers to determine whether to proceed with the upgrade.

Versioning for JavaEE Applications

When modifying your application's audit definition, it is recommended that you set the major and minor numbers as follows:

- Only increase the minor version number when making version-compatible changes, meaning changes in an audit definition such that the attribute database mapping table generated from the new audit definition should still work with the audit data created by the previous attribute database mapping table.

For example, suppose the current definition version is major=2 and minor=1. When adding a new event that does not affect the attribute database mapping table, you can change the minor version to 2 (minor=2), while the major version remains unchanged (major=2).

- Increase major version number when making version changes where the new mapping table is incompatible with the previous table.

28.5.1.2 Custom Attribute to Database Column Mappings

When registering a new component or application, the registration service creates an attribute-to-database column mapping table from the component's custom attributes, and then saves this table to the audit metadata store.

Attribute-database mapping tables are required to ensure unique mappings between your application's attribute definitions and database columns. The audit loader uses mapping tables to load data into the audit store; the tables are also used to generate audit reports from custom database table IAU_CUSTOM.

A custom attribute-database column mapping has properties of attribute name, database column name, and data type.

Each custom attribute must have a mapping order number in its definition. Attributes with the same data type are mapped to the database column in the sequence of attribute mapping order. For example, if the definition file looks like this:

```
<Attributes ns="accounting" version="1.1">
  <Attribute name="TransactionType" type="string" maxLength="0"
    displayName="Transaction Type" order="1"/>
  <Attribute name="AccountNumber" type="int" displayName="Account Number"
    order="2">
  <Attribute name="Date" type="dateTime" displayName="Date" order="3"/>
  <Attribute name="Amount" type="float" displayName="Amount" order="4"/>
  <Attribute name="Status" type="string" maxLength="0" displayName="Account
    Status" order="5"/>
  <Attribute name="Balance" type="float" displayName="Account Balance"
    order="6"/>
</Attributes>
```

then the mapping is as follows:

```

<AttributesMapping ns="accounting" tableName="IAU_CUSTOM" version="1.1">
  <AttributeColumn attribute="TransactionType" column="IAU_STRING_001"
    datatype="string"/>
  <AttributeColumn attribute="AccountNumber" column="IAU_INT_001"
    datatype="int"/>
  <AttributeColumn attribute="Date" column="IAU_DATETIME_001"
    datatype="dateTime"/>
  <AttributeColumn attribute="Amount" column="IAU_FLOAT_001" datatype="float"/>
  <AttributeColumn attribute="Status" column="IAU_STRING_002" datatype="string"/>
  <AttributeColumn attribute="Balance" column="IAU_FLOAT_002" datatype="float"/>
</AttributesMapping>

```

The version ID of the attribute-database column mapping table matches the version ID of the custom attribute group. This allows your application to maintain the backward compatibility of attribute mappings across audit definition versions. For more information about versioning, see [Section 28.5.1.1](#).

28.6 Register Application with the Registration Service

Java EE applications can be registered by packaging `component_events.xml` and `component_events_xlf.jar` in the META-INF folder of the application's EAR files. The audit registration service will process them automatically when the application is deployed.

Options include:

- Deployment - Registers the audit event definition to the audit metadata store if the application is not yet registered.
- Redeployment - Upgrades the component audit event definition if the component is already registered. See [Section 28.5.1](#) for details.
- Undeployment - Removes the application's audit event definition from the audit metadata store.

Registration parameters are set in the OPSS deployment descriptor `opss-application.xml`, which is also packaged in the META-INF folder of the application EAR files. [Table 28–2](#) shows the parameters with their options:

Table 28–2 Parameters for Audit Registration Service

Parameter	Option	Description
opss.audit.registration	OVERWRITE	Register component audit definition whether or not it is registered.
	UPGRADE	Register component audit definition according to versioning support.
	DISABLE	Do not register component audit definition.
opss.audit.deregistration	DELETE (default option)	Delete component audit definition from audit store when undeploying applications.
	DISABLE	Keep component audit definition in audit store when undeploying applications.

28.7 Add Application Code to Log Audit Events

Applications can programmatically access the run-time audit service to generate their own audit events using the client API.

28.7.1 Audit Client API

The audit client API is as follows:

```
Interface AuditService {

    Auditor getAuditor(String componentType);

    void register(AuditRegistration auditRegistration);

    void unregister(AuditRegistration auditRegistration);

}

Interface Auditor {

    boolean log(AuditEvent ev);

    boolean isEnabled(String categoryName, String eventType, boolean eventStatus,
        Map<String, Object> properties);

}

public class oracle.security.jps.service.audit.AuditEvent {
    public AuditEvent(AuditContext ctx, String eventType,
        String eventCategory, boolean eventStatus, String messageText);
    public void setInitiator(String initiator);
    public void setAttributeBoolean(String attributeName, Boolean attributeValue);
    public void setAttributeDouble(String attributeName, double attributeValue);
    public void setAttributeDate(String attributeName, Date attributeValue);
    public void setAttributeByteArray(String attributeName, byte[] attributeValue);
    public void setAttributeFloat(String attributeName, float attributeValue);
    public void setAttributeLong(String attributeName, long attributeValue);
    public void setAttributeInt(String attributeName, int attributeValue);
    public void setAttributeString(String attributeName, String attributeValue);
    public void setAttribute(String attributeName, Object attributeValue)

}

```

Subsequent sections explain how to obtain permissions and a run-time auditor instance.

28.7.2 Set System Grants

You must have system grants to get auditor instances from the audit service. In this example, the grant allows application `MyApp` to call `auditService.getAuditor("MyApp")` in `AccessController.doPrivileged` block:

```
<grant>
  <grantee>
    <codesource>
      <url>file:${oracle.deployed.app.dir}/MyApp${oracle.deployed.app.ext}</url>
    </codesource>

```

```

</grantee>
<permissions>
  <permission>
    <class>oracle.security.jps.service.audit.AuditStoreAccessPermission</class>
    <name>MyApp</name>
    <actions>read</actions>
  </permission>
</permissions>
</grant>

```

28.7.3 Obtain Auditor Instance

After your application registers to the audit service, it can get its runtime auditor instance programmatically from the OPSS audit service, as shown in the following sample code fragment:

```

//Gets audit service instance

final AuditService auditService =
JpsServiceLocator.getServiceLocator().lookup(AuditService.class);

//Gets Auditor instance for application 'MyApp'
Auditor auditor = AccessController.doPrivileged(
    new PrivilegedExceptionAction<Auditor>() {
        public Auditor run() throws AuditException {
            return auditService.getAuditor("MyApp");
        }
    });

final String category = "Transaction";
final String eventName = "deposit";

//Check if event 'deposit' is enabled in filtering.
boolean enabled = auditor.isEnabled(category, eventName, "true", null);
if (enabled) {
    AuditContext ctx = new AuditContext();
    String message = "deposit transaction";
    //Creates an audit event
    AuditEvent ev = new AuditEvent(ctx, eventName, category, "true", message);

    //Sets event attributes
    ev.setInitiator("johnsmith");
    ev.setAttributeInt("accounting:AccountNumber", 2134567);
    ev.setAttributeDate("accounting:Date", new Date());
    ev.setAttributeFloat("accounting:Amount", 100.00);

    //Logs audit event
    boolean ret = auditor.log(event);
}

```

28.8 Integrate with Oracle Business Intelligence Publisher

You can leverage Oracle Business Intelligence Publisher to generate reports from your application's audit data, utilizing the same reporting capabilities available to Oracle components.

The basic steps are as follows:

1. Use the mapping table to generate an Oracle BI Publisher report template.
2. Set up the Oracle BI Publisher report service.
3. Copy the report template into Oracle BI Publisher to view component audit events.
4. Generate reports with Oracle BI Publisher.

See Also: [Chapter 14, "Using Audit Analysis and Reporting"](#).

28.9 Update and Maintain Audit Definitions

As the application's audit requirements evolve, you can update the integration to reflect the changes. The steps are as follows:

1. Update the audit definition file. Be mindful of the versioning rules as you take this step.

See Also: [Section 28.5.1, "Understand Mapping and Versioning Rules"](#).

2. Redeploy the application EAR file with the updated event definition file. Alternatively, you can notify the audit registration service of the existence of a newer version.
3. Verify your changes.

Part VI

Appendices

This part contains the following appendices:

- [Appendix A, "OPSS Configuration File Reference"](#)
- [Appendix B, "File-Based Identity and Policy Store Reference"](#)
- [Appendix C, "Oracle Fusion Middleware Audit Framework Reference"](#)
- [Appendix D, "User and Role API Reference"](#)
- [Appendix E, "Administration with WLST Scripting and MBean Programming"](#)
- [Appendix F, "OPSS System and Configuration Properties"](#)
- [Appendix G, "Upgrading Security Data"](#)
- [Appendix H, "References"](#)
- [Appendix I, "OPSS Scripts"](#)
- [Appendix J, "Using an OpenLDAP Identity Store"](#)
- [Appendix L, "Troubleshooting Security in Oracle Fusion Middleware"](#)

OPSS Configuration File Reference

This appendix describes the element hierarchy and attributes in the file that configures OPSS services. By default, this file is named `jps-config.xml` (for Java EE applications) or `jps-config-jse.xml` (for Java SE applications) and is located in the directory `$DOMAIN_HOME/config/fmwconfig`.

For Java SE applications, an alternative location can be specified using the system property `oracle.security.jps.config`.

The configuration file is used to configure the policy, credential, and identity stores, the login modules, and the audit service. For a complete example of a configuration file see [Section 21.4.9, "Example of Configuration File jps-config.xml."](#)

To configure services programmatically, see [Section E.2, "Configuring OPSS Services with MBeans."](#)

This appendix includes the following sections:

- [Top- and Second-Level Element Hierarchy](#)
- [Lower-Level Elements](#)

A.1 Top- and Second-Level Element Hierarchy

The top element in the file `jps-config.xml` is `<jpsConfig>`. It contains the following second-level elements:

- `<property>`
- `<propertySets>`
- `<extendedProperty>`
- `<serviceProviders>`
- `<serviceInstances>`
- `<jpsContexts>`

[Table A-1](#) describes the function of these elements. The annotations between curly braces `{ }` indicate the number of occurrences the element is allowed. For example, `{0 or more}` indicates that the element can occur 0 or more times; `{1}` indicates that the element must occur once.

These elements are *not* application-specific configurations: all items in the configuration file pertain to an entire domain and apply to all managed servers and applications deployed in the domain.

Table A-1 First- and Second-Level Elements in *jps-config.xml*

Elements	Description
<code><jpsConfig></code> {1}	Defines the top-level element in the configuration file.
<code><property></code> {0 or more}	Defines names and values of properties. It can also appear elsewhere in the hierarchy, such as under the elements <code><propertySet></code> , <code><serviceProvider></code> , and <code><serviceInstance></code> .
<code><propertySets></code> {0 or 1} <code><propertySet></code> {1 or more} <code><property></code> {1 or more}	Groups one or more <code><propertySet></code> elements so that they can be referenced as a group.
<code><extendedProperty></code> {0 or more} <code><name></code> {1} <code><values></code> {1} <code><value></code> {1 or more}	Defines a property that has multiple values. It can also appear elsewhere in the hierarchy, such as under the elements <code>extendedProperty</code> and <code>serviceInstance</code> .
<code><extendedPropertySets></code> {0 or 1} <code><extendedPropertySet></code> {1 or more} <code><extendedProperty></code> {1 or more} <code><name></code> {1} <code><values></code> {1} <code><value></code> {1 or more}	Groups one or more <code><extendedPropertySet></code> elements so that they can be referenced as a group.
<code><serviceProviders></code> {0 or 1} <code><serviceProvider></code> {1 or more} <code><description></code> {0 or 1} <code><property></code> {0 or more}	Groups one or more <code><serviceProvider></code> elements, each of which defines an implementation of an OPSS service, such as a policy store provider, a credential store provider, or a login module.
<code><serviceInstances></code> {0 or 1} <code><serviceInstance></code> {1 or more} <code><description></code> {0 or 1} <code><property></code> {0 or more} <code><propertySetRef></code> {0 or more} <code><extendedProperty></code> {0 or more} <code><name></code> {1} <code><values></code> {1} <code><value></code> {1 or more} <code><extendedPropertySetRef></code> {0 or more}	Groups one or more <code><serviceInstance></code> elements, each of which configures and specifies property values for a service provider defined in a <code><serviceProvider></code> element.
<code><jpsContexts></code> {1} <code><jpsContext></code> {1 or more} <code><serviceInstanceRef></code> {1 or more}	Groups one or more <code><jpsContext></code> elements, each of which is a collection of service instances that an application can use.

A.2 Lower-Level Elements

This section describes, in alphabetical order, the complete set of elements that can occur in under the second-level elements described in the [Top- and Second-Level Element Hierarchy](#).

- `<description>`
- `<extendedProperty>`
- `<extendedPropertySet>`
- `<extendedPropertySetRef>`
- `<extendedPropertySets>`
- `<jpsConfig>`
- `<jpsContext>`

- <jpsContexts>
- <name>
- <property>
- <propertySet>
- <propertySetRef>
- <propertySets>
- <serviceInstance>
- <serviceInstanceRef>
- <serviceInstances>
- <serviceProvider>
- <serviceProviders>
- <value>
- <values>

<description>

<description>

This element describes the corresponding entity (a service instance or service provider).

Parent Elements

<serviceInstance> or <serviceProvider>

Child Element

None.

Occurrence

<description> can be a child of <serviceInstance> or <serviceProvider>.

- As a child of <serviceInstance>:

```
<serviceInstances> {0 or 1}
  <serviceInstance> {1 or more}
    <description> {0 or 1}
    <property> {0 or more}
    <propertySetRef> {0 or more}
    <extendedProperty> {0 or more}
      <name> {1}
      <values> {1}
        <value> {1 or more}
    <extendedPropertySetRef> {0 or more}
```

- As a child of <serviceProvider>:

```
<serviceProviders> {0 or 1}
  <serviceProvider> {1 or more}
    <description> {0 or 1}
    <property> {0 or more}
```

Example

The following example sets a description for a service provider.

```
<serviceProvider ... >
  <description>XML-based IdStore Provider</description>
  ...
</serviceProvider>
```

<extendedProperty>

This element defines an extended property in the following scenarios:

Table A–2 Scenarios for <extendedProperty>

Location in jps-config.xml	Function
Directly under <jpsConfig>	Defines an extended property for general use. As a child of <jpsConfig>, an extended property can specify, for example, all the base DN's in an LDAP-based authenticators.
Directly under <extendedPropertySet>	Defines an extended property for general use that is part of an extended property set.
Directly under <serviceInstance>	Defines an extended property for a particular service instance.

An extended property typically includes multiple values. Use a <value> element to specify each value. Several LDAP identity store properties are in this category, such as the specification of the following values:

- Object classes used for creating user objects
- Attribute names that must be specified when creating a user
- Base DN's for searching users

Parent Elements

<extendedPropertySet>, <jpsConfig>, or <serviceInstance>

Child Elements

<name> or <values>

Occurrence

<extendedProperty> can be a child of <extendedPropertySet>, <jpsConfig>, or <serviceInstance>.

- As a child of <extendedPropertySet>:

```

<extendedPropertySets> {0 or 1}
  <extendedPropertySet> {1 or more}
    <extendedProperty> {1 or more}
      <name> {1}
      <values> {1}
        <value> {1 or more}

```

- As a child of <jpsConfig>:

```

<jpsConfig>
  <extendedProperty> {0 or more}
    <name> {1}
    <values> {1}
      <value> {1 or more}

```

- As a child of <serviceInstance>:

```

<serviceInstances> {0 or 1}

```

```
<serviceInstance> {1 or more}
  <description> {0 or 1}
  <property> {0 or more}
  <propertySetRef> {0 or more}
  <extendedProperty> {0 or more}
    <name> {1}
    <values> {1}
      <value> {1 or more}
    <extendedPropertySetRef> {0 or more}
```

Example

The following example sets a single value:

```
<extendedProperty>
  <name>user.search.bases</name>
  <values>
    <value>cn=users,dc=us,dc=oracle,dc=com</value>
  </values>
</extendedProperty>
```

<extendedPropertySet>

This element defines a set of extended properties. The extended property set can then be referenced by an <extendedPropertySetRef> element to specify the given properties as part of the configuration of a service instance.

Attributes

Name	Description
name	Designates a name for the extended property set. No two <extendedPropertySet> elements may have the same name attribute setting within a configuration file. Values: string Default: n/a (required)

Parent Element

<extendedPropertySets>

Child Element

<extendedProperty>

Occurrence

Required within <extendedPropertySets>, one or more:

```
<extendedPropertySets> {0 or 1}
  <extendedPropertySet> {1 or more}
    <extendedProperty> {1 or more}
      <name> {1}
      <values> {1}
        <value> {1 or more}
```

<extendedPropertySetRef>

This element configures a service instance by referring to an extended property set defined elsewhere in the file.

Attributes

Name	Description
ref	Refers to an extended property set whose extended properties are used for the service instance defined in the <serviceInstance> parent element. The ref value of <extendedPropertySetRef> must match the name value of an <extendedPropertySet> element. Values: string Default: n/a (required)

Parent Element

[<serviceInstance>](#)

Child Element

None.

Occurrence

Optional, zero or more.

```
<serviceInstances> {0 or 1}
  <serviceInstance> {1 or more}
    <description> {0 or 1}
    <property> {0 or more}
    <propertySetRef> {0 or more}
    <extendedProperty> {0 or more}
      <name> {1}
      <values> {1}
        <value> {1 or more}
    <extendedPropertySetRef> {0 or more}
```


<extendedPropertySets>

This element specifies a set of properties.

Parent Element

<jpsConfig>

Child Element

<extendedPropertySet>

Occurrence

Optional, zero or one.

```
<jpsConfig>
  <extendedPropertySets> {0 or 1}
    <extendedPropertySet> {1 or more}
      <extendedProperty> {1 or more}
        <name> {1}
        <values> {1}
          <value> {1 or more}
```

<jpsConfig>

<jpsConfig>

This is the root element of a configuration file.

Parent Element

None.

Child Elements

[<extendedProperty>](#), [<extendedPropertySets>](#), [<jpsContexts>](#), [<property>](#),
[<propertySets>](#), [<serviceInstances>](#), or [<serviceProviders>](#)

Occurrence

Required, one only.

Example

```
<jpsConfig xmlns="http://xmlns.oracle.com/oracleas/schema/11/jps-config-11_1.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.oracle.com/oracleas/schema/11/jps-config-11_
1.xsd"
  schema-major-version="11" schema-minor-version="1">
  ...
</jpsConfig>
```

<jpsContext>

This element declares an OPSS context, a collection of service instances common to a domain, either by referring to a set of service instances that comprise the context (typical usage), or by referring to another context. Each <jpsContext> in a configuration file must have a distinct name.

Attributes

Name	Description
name	Designates a name for the OPSS context. Each context must have a unique name. Values: string Default: n/a (required)

Parent Element

<jpsContexts>

Child Element

<serviceInstanceRef>

Occurrence

There must be at least one <jpsContext> element under <jpsContexts>. A <jpsContext> element contains the <serviceInstanceRef> element.

```
<jpsContexts> {1}
  <jpsContext> {1 or more}
    <serviceInstanceRef> {1 or more}
```

Example

The following example illustrates the definition of two contexts; the first one, named `default`, is the default context (specified by the attribute `default` in <jpsContexts>), and it references several service instances by name.

The second one, named `anonymous`, is used for unauthenticated users, and it references the `anonymous` and `anonymous.loginmodule` service instances.

```
<serviceInstances>
...
  <serviceInstance provider="credstoressp" name="credstore">
    <description>File Based Default Credential Store Service Instance</description>
    <property name="location" value="{oracle.instance}/config/JpsDataStore/JpsSystemStore"/>
  </serviceInstance>
...
  <serviceInstance provider="anonymous.provider" name="anonymous">
    <property value="anonymous" name="anonymous.user.name"/>
    <property value="anonymous-role" name="anonymous.role.name"/>
  </serviceInstance>
...
  <serviceInstance provider="jaas.login.provider" name="anonymous.loginmodule">
    <description>Anonymous Login Module</description>
    <property value="oracle.security.jps.internal.jaas.module.anonymous.AnonymousLoginModule"
```

<jpsContext>

```
        name="loginModuleClassName"/>
    <property value="REQUIRED"
        name="jaas.login.controlFlag"/>
</serviceInstance>
...
</serviceInstances>
...
<jpsContexts default="default">
...
    <jpsContext name="default">
        <!-- This is the default JPS context. All the mandatory services and Login Modules must be
            configured in this default context -->
        <serviceInstanceRef ref="credstore"/>
        <serviceInstanceRef ref="idstore.xml"/>
        <serviceInstanceRef ref="policystore.xml"/>
        <serviceInstanceRef ref="idstore.loginmodule"/>
        <serviceInstanceRef ref="idm"/>
    </jpsContext>
    <jpsContext name="anonymous">
        <serviceInstanceRef ref="anonymous"/>
        <serviceInstanceRef ref="anonymous.loginmodule"/>
    </jpsContext>
...
</jpsContexts>
```

<jpsContexts>

This element specifies a set of contexts.

Attributes

Name	Description
default	Specifies the context that is used by an application if none is specified. The <code>default</code> value of the <code><jpsContexts></code> element must match the name of a <code><jpsContext></code> child element. Values: string Default: n/a (required) Note: The default context must configure all mandatory services and login modules.

Parent Element

`<jpsConfig>`

Child Element

`<jpsContext>`

Occurrence

Required, one only.

```
<jpsConfig>  
  <jpsContexts> {1}  
    <jpsContext> {1 or more}
```

Example

See `<jpsContext>` for an example.

<name>

<name>

This element specifies the name of an extended property.

Parent Element

[<extendedProperty>](#)

Child Element

None

Occurrence

Required, one only.

```
<extendedProperty> {0 or more}
  <name> {1}
  <values> {1}
    <value> {1 or more}
```

Example

See [<extendedProperty>](#) for an example.

<property>

This element defines a property in the following scenarios:

Table A-3 Scenarios for <property>

Location in jps-config.xml	Function
Directly under <jpsConfig>	Defines a one-value property for general use.
Directly under <propertySet>	Defines a multi-value property for general use that is part of a property set.
Directly under <serviceInstance>	Defines a property for use by a particular service instance.
Directly under <serviceProvider>	Defines a property for use by all service instances of a particular service provider.

For a list of properties, see [Appendix F, "OPSS System and Configuration Properties"](#).

Attributes

Name	Description
name	Specifies the name of the property being set. Values: string Default: n/a (required)
value	Specifies the value of the property being set. Values: string Default: n/a (required)

Parent Elements

<jpsConfig>, <propertySet>, <serviceInstance>, or <serviceProvider>

Child Element

None.

Occurrence

Under a <propertySet>, it is required, one or more; otherwise, it is optional, zero or more.

- As a child of <jpsConfig>:


```
<jpsConfig>
  <property> {0 or more}
```
- As a child of <propertySet>:


```
<propertySets> {0 or 1}
  <propertySet> {1 or more}
    <property> {1 or more}
```
- As a child of <serviceInstance>:


```
<serviceInstances> {0 or 1}
```

```
<serviceInstance> {1 or more}
  <description> {0 or 1}
  <property> {0 or more}
  <propertySetRef> {0 or more}
  <extendedProperty> {0 or more}
    <name> {1}
    <values> {1}
      <value> {1 or more}
    <extendedPropertySetRef> {0 or more}
```

- As a child of <serviceProvider>:

```
<serviceProviders> {0 or 1}
  <serviceProvider> {1 or more}
    <description> {0 or 1}
    <property> {0 or more}
```

Example

The following example illustrates a property to disable JAAS mode for authorization:

```
<jpsConfig ... >
  ...
  <property name="oracle.security.jps.jaas.mode" value="off"/>
  ...
</jpsConfig>
```

For additional examples, see [<propertySet>](#) and [<serviceInstance>](#).

<propertySet>

This element defines a set of properties. Each property set has a name so that it can be referenced by a <propertySetRef> element to include the properties as part of the configuration of a service instance.

Attributes

Name	Description
name	Designates a name for the property set. No two <propertySet> elements may have the same name within a jps-config.xml file. Values: string Default: n/a (required)

Parent Element

<propertySets>

Child Element

<property>

Occurrence

Required within a <propertySets>, one or more

```
<propertySets> {0 or 1}
  <propertySet> {1 or more}
    <property> {1 or more}
```

Example

```
<propertySets>
...
  <!-- For property that points to valid Access SDK installation directory -->
  <propertySet name="access.sdk.properties">
    <property name="access.sdk.install.path" value="$ACCESS_SDK_HOME"/>
  </propertySet>
...
</propertySets>

<serviceInstances>
...
  <serviceInstance provider="jaas.login.provider" name="oam.loginmodule">
    <description>Oracle Access Manager Login Module</description>
    <property
      value="oracle.security.jps.internal.jaas.module.oam.OAMLoginModule"
      name="loginModuleClassName"/>
    <property value="REQUIRED" name="jaas.login.controlFlag"/>
    <propertySetRef ref="access.sdk.properties"/>
  </serviceInstance>
...
</serviceInstances>
```

<propertySetRef>

This element configures a service instance by referring to a property set defined elsewhere in the file.

Attributes

Name	Description
ref	Refers to a property set whose properties are used by the service instance defined in the <serviceInstance> parent element. The ref value of a <propertySetRef> element must match the name of a <propertySet> element. Values: string Default: n/a (required)

Parent Element

[<serviceInstance>](#)

Child Element

None.

Occurrence

Optional, zero or more.

```
<serviceInstances> {0 or 1}
  <serviceInstance> {1 or more}
    <description> {0 or 1}
    <property> {0 or more}
    <propertySetRef> {0 or more}
    <extendedProperty> {0 or more}
      <name> {1}
      <values> {1}
        <value> {1 or more}
    <extendedPropertySetRef> {0 or more}
```

Example

See [<propertySet>](#) for an example.

<propertySets>

This element specifies a set of property sets.

Parent Element

<jpsConfig>

Child Element

<propertySet>

Occurrence

Optional. If present, there can be only one <propertySets> element.

```
<jpsConfig>  
  <propertySets> {0 or 1}  
    <propertySet> {1 or more}  
      <property> {1 or more}
```

Example

See <propertySet> for an example.

<serviceInstance>

This element defines an instance of a service provider, such as an identity store service instance, policy store service instance, or login module service instance.

Each provider instance specifies the name of the instance, used to refer to the provider within the configuration file; the name of the provider being instantiated; and, possibly, the properties of the instance. Properties include the location of the instance and can be specified directly, within the instance element itself, or indirectly, by referencing a property or a property set. To change the properties of a service instance, you can use the procedure explained in [Section E.1, "Configuring OPSS Service Provider Instances with a WLST Script."](#)

Set properties and extended properties of a service instance in the following ways:

- Set properties directly through [<property>](#) subelements.
- Set extended properties directly through [<extendedProperty>](#) subelements.
- Refer to previously defined sets of properties through [<propertySetRef>](#) subelements.
- Refer to previously defined sets of extended properties through [<extendedPropertySetRef>](#) subelements.

Attributes

Name	Description
name	Designates a name for this service instance. Note that no two <serviceInstance> elements may have the same name attribute setting within a <code>jps-config.xml</code> file. Values: string Default: n/a (required)
provider	Indicates which service provider this is an instance of. The <code>provider</code> value of a <serviceInstance> element must match the name value of a <serviceProvider> element. Values: string Default: n/a (required)

Parent Element

[<serviceInstances>](#)

Child Elements

[<description>](#), [<extendedProperty>](#), [<extendedPropertySetRef>](#), [<property>](#), or [<propertySetRef>](#)

Occurrence

Required within [<serviceInstances>](#), one or more.

```
<serviceInstances> {0 or 1}
  <serviceInstance> {1 or more}
    <description> {0 or 1}
    <property> {0 or more}
```

```

<propertySetRef> {0 or more}
<extendedProperty> {0 or more}
  <name> {1}
  <values> {1}
    <value> {1 or more}
<extendedPropertySetRef> {0 or more}

```

Examples

Example 1

The following example illustrates the configuration of a file-based identity store service. For a file-based identity store, the subscriber name is the default realm. The example sets the location using the `location` property.

```

<serviceInstances>
  <serviceInstance name="idstore.xml" provider="idstore.xml.provider">
    <!-- Subscriber name must be defined for XML Identity Store -->
    <property name="subscriber.name" value="jazn.com"/>
    <!-- This is the location of XML Identity Store -->
    <property name="location" value="./system-jazn-data.xml"/>
  </serviceInstance>
  ...
</serviceInstances>

```

Example 2

The following example illustrates the configuration a credential store service. It uses the `location` property to set the location of the credential store.

```

<serviceInstances>
  <serviceInstance provider="credstoressp" name="credstore">
    <description>File Based Default Credential Store Service
      Instance</description>
    <property name="location"
      value="\${oracle.instance}/config/JpsDataStore/JpsSystemStore" />
  </serviceInstance>
  ...
</serviceInstances>

```

Example 3

The following example illustrates the configuration of an LDAP-based identity store using Oracle Internet Directory:

```

<serviceInstance name="idstore.oid" provider="idstore.ldap.provider">
  <property name="subscriber.name" value="dc=us,dc=oracle,dc=com"/>
  <property name="idstore.type" value="OID"/>
  <property name="security.principal.key" value="ldap.credentials"/>
  <property name="security.principal.alias" value="JPS"/>
  <property name="ldap.url" value="ldap://myServerName.com:389"/>
  <extendedProperty>
    <name>user.search.bases</name>
    <values>
      <value>cn=users,dc=us,dc=oracle,dc=com</value>
    </values>
  </extendedProperty>
  <extendedProperty>
    <name>group.search.bases</name>
    <values>

```

```
        <value>cn=groups,dc=us,dc=oracle,dc=com</value>
    </values>
</extendedProperty>
<property name="username.attr" value="uid"/>
<property name="groupname.attr" value="cn"/>
</serviceInstance>
```

Example 4

The following example illustrates the configuration of an audit provider:

```
<serviceInstances>
  <serviceInstance name="audit" provider="audit.provider">
    <property name="audit.filterPreset" value="Low"/>
    <property name="audit.specialUsers" value="admin, fmwadmin" />
    <property name="audit.customEvents" value="JPS:CheckAuthorization,
      CreateCredential, OIF:UserLogin"/>
    <property name="audit.loader.jndi" value="jdbc/AuditDB"/>
    <property name="audit.loader.interval" value="15" />
    <property name="audit.maxDirSize" value="102400" />
    <property name="audit.maxFileSize" value="10240" />
    <property name="audit.loader.repositoryType" value="Db" />
  </serviceInstance>
</serviceInstances>
```

See Also:

- [<serviceProvider>](#), for related examples defining service providers referenced here.
- [<jpsContext>](#), for a corresponding example of [<serviceInstanceRef>](#).

<serviceInstanceRef>

This element refers to service instances.

Attributes

Name	Description
ref	Refers to a service instance that are part of the context defined in the <jpsContext> parent element. The ref value of a <serviceInstanceRef> element must match the name of a <serviceInstance> element. Values: string Default: n/a (required)

Parent Element

<jpsContext>

Child Element

None

Occurrence

Required within a <jpsContext>, one or more.

```
<jpsContexts> {1}
  <jpsContext> {1 or more}
    <serviceInstanceRef> {1 or more}
```

Example

See <jpsContext> for an example.

<serviceInstances>

This element is the parent of a [<serviceInstance>](#) element.

Parent Element

[<jpsConfig>](#)

Child Element

[<serviceInstance>](#)

Occurrence

Optional, zero or one.

```
<jpsConfig>
  <serviceInstances> {0 or 1}
    <serviceInstance> {1 or more}
      <description> {0 or 1}
      <property> {0 or more}
      <propertySetRef> {0 or more}
      <extendedProperty> {0 or more}
        <name> {1}
        <values> {1}
          <value> {1 or more}
      <extendedPropertySetRef> {0 or more}
```

Example

See [<serviceInstance>](#) for an example.

<serviceProvider>

This element defines a service provider. Each provider specifies the type of the provider, such as credential store, authenticators, policy store, or login module; the name of the provider, used to refer to the provider within the configuration file; and the Java class that implements the provider and that is instantiated when the provider is created. Furthermore, the element `property` specifies settings used to instantiate the provider.

It specifies the following data:

- The type of service provider (specified in the `type` attribute)
- A designated name of the service provider (to be referenced in each `<serviceInstance>` element that defines an instance of this service provider)
- The class that implements this service provider and is instantiated for instances of this service provider
- Optionally, properties that are generic to any instances of this service provider

Attributes

Name	Description
<code>type</code>	<p>Specifies the type of service provider being declared; it must be either of the following:</p> <p> CREDENTIAL_STORE IDENTITY_STORE POLICY_STORE AUDIT LOGIN ANONYMOUS KEY_STORE IDM (for pluggable identity management) CUSTOM </p> <p>The implementation class more specifically defines the type of provider, such as by implementing a file-based identity store or LDAP-based policy store, for example.</p> <p>Values: string (a value above)</p> <p>Default: n/a (required)</p>
<code>name</code>	<p>Designates a name for this service provider. This name is referenced in the <code>provider</code> attribute of <code><serviceInstance></code> elements to create instances of this provider. No two <code><serviceProvider></code> elements may have the same name attribute setting within a configuration file.</p> <p>Values: string</p> <p>Default: n/a (required)</p>

Name	Description
class	Specifies the fully qualified name of the Java class that implements this service provider (and that is instantiated to create instances of the service provider). Values: string Default: n/a (required)

Parent Element

[<serviceProviders>](#)

Child Elements

[<description>](#) or [<property>](#)

Occurrence

Required within the [<serviceProviders>](#) element, one or more.

```
<serviceProviders> {0 or 1}
  <serviceProvider> {1 or more}
    <description> {0 or 1}
    <property> {0 or more}
```

Examples

The following example illustrates the specification of a login module service provider:

```
<serviceProviders>
  <serviceProvider type="LOGIN" name="jaas.login.provider"
    class="oracle.security.jps.internal.login.jaas.JaasLoginServiceProvider">
    <description>This is Jaas Login Service Provider and is used to configure
    login module service instances</description>
  </serviceProvider>
</serviceProviders>
```

The following example illustrates the definition of an audit service provider:

```
<serviceProviders>
  <serviceProvider name="audit.provider" type="AUDIT"
class="oracle.security.jps.internal.audit.AuditProvider">
  </serviceProvider>
</serviceProviders>
```

See [<serviceInstance>](#) for other examples.

<serviceProviders>

This element specifies a set of service providers.

Parent Element

<jpsConfig>

Child Element

<serviceProvider>

Occurrence

Optional, one only.

```
<jpsConfig>
  <serviceProviders> {0 or 1}
    <serviceProvider> {1 or more}
      <description> {0 or 1}
      <property> {0 or more}
```

Example

See <serviceProvider> for an example.

<value>

<value>

This element specifies a value of an extended property, which can have multiple values. Each <value> element specifies one value.

Parent Element

<values>

Child Element

None.

Occurrence

Required within <values>, one or more.

```
<extendedProperty> {0 or more}
  <name> {1}
  <values> {1}
    <value> {1 or more}
```

Example

See <extendedProperty> for an example.

<values>

This element is the parent element of a [<value>](#) element.

Parent Element

[<extendedProperty>](#)

Child Element

[<value>](#)

Occurrence

Required within [<extendedProperty>](#), one only.

```
<extendedProperty> {0 or more}
  <name> {1}
  <values> {1}
    <value> {1 or more}
```

Example

See [<extendedProperty>](#) for an example.

<values>

B

File-Based Identity and Policy Store Reference

This appendix describes the elements and attributes in `system-jazn-data.xml`, which is the default store for file-based identity and policy stores in Oracle Platform Security Services.

Note: The file-based identity store is supported for Java SE applications only.

This appendix covers the following topics:

- [Hierarchy of Elements in system-jazn-data.xml](#)
- [Elements and Attributes of system-jazn-data.xml](#)

B.1 Hierarchy of Elements in system-jazn-data.xml

This section shows the element hierarchy of `system-jazn-data.xml`, or an application-specific `jazn-data.xml` file. The direct subelements of the `<jazn-data>` root element are:

- `<jazn-realm>`
- `<policy-store>`
- `<jazn-policy>`

Note: The `<jazn-principal-classes>` and `<jazn-permission-classes>` elements and their subelements may appear in the `system-jazn-data.xml` schema definition as subelements of `<policy-store>`, but are for backward compatibility only.

Table B-1 Hierarchy of Elements in system-jazn-data.xml

Hierarchy	Description
<code><jazn-data></code>	This is the top-level element in the <code>system-jazn-data.xml</code> file.

Table B-1 (Cont.) Hierarchy of Elements in system-jazn-data.xml

Hierarchy	Description
<pre> <jazn-realm> {0 or 1} <realm> {0 or more} <name> {1} <users> {0 or 1} <user> {0 or more} <name> {1} <display-name> {0 or 1} <description> {0 or 1} <guid> {0 or 1} <credentials> {0 or 1} <roles> {0 or 1} <role> {0 or more} <name> {1} <display-name> {0 or 1} <description> {0 or 1} <guid> {0 or 1} <members> {0 or 1} <member> {0 or more} <type> {1} <name> {1} <owners> {0 or 1} <owner> {0 or more} <type> {1} <name> {1} </pre>	<p>The <code><jazn-realm></code> section specifies security realms, and the users and enterprise groups (as opposed to application-level roles) included in each realm.</p>

Table B-1 (Cont.) Hierarchy of Elements in system-jazn-data.xml

Hierarchy	Description
<pre> <policy-store> {0 or 1} <applications> {0 or 1} <application> {1 or more} <name> {1} <description> {0 or 1} <app-roles> {0 or 1} <app-role> {1 or more} <name> {1} <class> {1} <display-name> {0 or 1} <description> {0 or 1} <guid> {0 or 1} <uniquename> {0 or 1} <extended-attributes> {0 or 1} <attribute> {1 or more} <name> {1} <values> {1} <value> {1 or more} <members> {0 or 1} <member> {1 or more} <name> {1} <class> {1} <uniquename> {0 or 1} <guid> {0 or 1} <role-categories> <role-category> <name> <display-name> <description> <members> <role-name-ref> <resource-types> <resource-type> <name> <display-name> <description> <provider-name> <matcher-class> <actions-delimiter> <actions> <resources> <resource> <name> <display-name> <description> <type-name-ref> <permission-sets> <permission-set> <name> <member-resources> <member-resource> <resource-name> <type-name-ref> <actions> <jazn-policy> {0 or 1} <grant> {0 or more} <description> {0 or 1} <grantee> {0 or 1} <principals> {0 or 1} <principal> {0 or more} <name> {1} <class> {1} <uniquename> {0 or 1} </pre>	<p>The <code><policy-store></code> section configures application-level policies. You can define roles at the application level, and members in the roles. Members can be users or roles.</p> <p>When <code><jazn-policy></code> is specified under the <code><application></code> element, it specifies policies at the application level.</p> <p><code><jazn-policy></code> can also appear under the <code><jazn-data></code> element, in which case it specifies policies at the system level.</p>

Table B-1 (Cont.) Hierarchy of Elements in system-jazn-data.xml

Hierarchy	Description
<pre> <jazn-policy> {0 or 1} <grant> {0 or more} <description> {0 or 1} <grantee> {0 or 1} <principals> {0 or 1} <principal> {0 or more} <name> {1} <class> {1} <uniqueusername> {0 or 1} <guid> {0 or 1} <codesource> {0 or 1} <url> {1} <permissions> {0 or 1} <permission> {1 or more} <class> {1} <name> {0 or 1} <actions> {0 or 1} <permission-sets> <permission-set> <name> </pre>	<p>When the <code><jazn-policy></code> element is located under the <code><jazn-data></code> element, it specifies policies at the system-level.</p> <p><code><jazn-policy></code> can also appear under the <code><application></code> element, in which case it specifies policies at the application level.</p>

B.2 Elements and Attributes of system-jazn-data.xml

This section describes the elements and attributes in the `system-jazn-data.xml` file.

Notes:

- You can update most settings in `system-jazn-data.xml` through Oracle Enterprise Manager Fusion Middleware Control.
-
-

- `<actions>`
- `<actions-delimiter>`
- `<app-role>`
- `<app-roles>`
- `<application>`
- `<applications>`
- `<attribute>`
- `<class>`
- `<codesource>`
- `<credentials>`
- `<description>`
- `<display-name>`
- `<extended-attributes>`

- <grant>
- <grantee>
- <guid>
- <jazn-data>
- <jazn-policy>
- <jazn-realm>
- <matcher-class>
- <member>
- <member-resource>
- <member-resources>
- <members>
- <name>
- <owner>
- <owners>
- <permission>
- <permissions>
- <permission-set>
- <permission-sets>
- <policy-store>
- <principal>
- <principals>
- <provider-name>
- <realm>
- <resource>
- <resource-name>
- <resources>
- <resource-type>
- <resource-types>
- <role>
- <role-categories>
- <role-category>
- <role-name-ref>
- <roles>
- <type>
- <type-name-ref>
- <uniqueusername>
- <url>

- `<user>`
- `<users>`
- `<value>`
- `<values>`

<actions>

This element specifies the operations permitted by the associated permission class. Values are case-sensitive and are specific to each permission implementation. Examples of actions are "invoke" and "read,write".

Parent Element

[<permission>](#)

Child Elements

None

Occurrence

Optional, zero or one:

```
<jazn-policy> {0 or 1}
  <grant> {0 or more}
    <description> {0 or 1}
    <grantee> {0 or 1}
      <principals> {0 or 1}
      ...
    <codesource> {0 or 1}
      <url> {1}
    <permissions> {0 or 1}
      <permission> {1 or more}
        <class> {1}
        <name> {0 or 1}
        <actions> {0 or 1}
```

Examples

See [<jazn-policy>](#) for examples.

<actions-delimiter>

This element specifies the character used to separate the actions of the associated resource type.

Parent Element

[<resource-types>](#)

Child Elements

[<name>](#), [<display-name>](#), [<description>](#), [<actions>](#)[<roles>](#), [<users>](#)

Occurrence

Optional, zero or more

```
<policy-store> {0 or 1}
  <applications> {0 or 1}
    <application> {1 or more}
      <name> {1}
      <description> {0 or 1}
      <app-roles> {0 or 1}
      ...
      <role-categories>
      ...
      <resource-types>
        <resource-type>
          <name>
          <display-name>
          <description>
          <provider-name>
          <matcher-class>
          <actions-delimiter>
          <actions>
```

Example

For an example, see [<resource-type>](#).

<app-role>

This element specifies an application role.

Required subelements specify the following:

- `<name>` specifies the name of the application role.
- `<class>` specifies the fully qualified name of the class implementing the application role.

Optional subelements can specify the following:

- `<description>` provides more information about the application role.
- `<display-name>` specifies a display name for the application role, such as for use by GUI interfaces.
- `<guid>` specifies a globally unique identifier to reference the application role. This is for internal use only.
- `<members>` specifies the users, roles, or other application roles that are members of this application role.
- `<uniqueusername>` specifies a unique name to reference the application role. This is for internal use only.

Parent Element

`<app-roles>`

Child Elements

`<class>`, `<description>`, `<display-name>`, `<guid>`, `<members>`, `<name>`, `<uniqueusername>`

Occurrence

Required, one or more:

```

<policy-store> {0 or 1}
  <applications> {0 or 1}
    <application> {1 or more}
      <name> {1}
      <description> {0 or 1}
      <app-roles> {0 or 1}
        <app-role> {1 or more}
          <name> {1}
          <class> {1}
          <display-name> {0 or 1}
          <description> {0 or 1}
          <guid> {0 or 1}
          <uniqueusername> {0 or 1}
          <extended-attributes> {0 or 1}
            <attribute> {1 or more}
              <name> {1}
              <values> {1}
                <value> {1 or more}
          <members> {0 or 1}
            <member> {1 or more}
              <name> {1}

```

<app-role>

```
<class> {1}
<uniqueusername> {0 or 1}
<guid> {0 or 1}
```

Examples

See [<policy-store>](#) for examples.

<app-roles>

This element specifies a set of application roles.

Parent Element

<application>

Child Elements

<app-role>

Occurrence

Optional, zero or one:

```
<policy-store> {0 or 1}
  <applications> {0 or 1}
    <application> {1 or more}
      <name> {1}
      <description> {0 or 1}
      <app-roles> {0 or 1}
        <app-role> {1 or more}
        ...
```

Example

See <policy-store> for examples.

<application>

This element specifies roles and policies for an application.

Required subelements specify the following information for an application:

- `<name>` specifies the name of the application.

Optional subelements can specify the following:

- `<description>` provides information about the application and its roles and policies.
- `<app-roles>` specifies any application-level roles
- `<jazn-policy>` specifies any application-level policies.

Parent Element

`<applications>`

Child Elements

`<app-roles>`, `<description>`, `<jazn-policy>`, `<name>`,
`<permission-sets>`, `<resource-types>`, `<resources>`, `<role-categories>`

Occurrence

Required, one or more:

```
<policy-store> {0 or 1}
  <applications> {0 or 1}
    <application> {1 or more}
      <name> {1}
      <description> {0 or 1}
      <app-roles> {0 or 1}
        <app-role> {1 or more}
      ...
```

Example

See `<policy-store>` for examples.

<applications>

This element specifies a set of applications.

Parent Element

<policy-store>

Child Elements

<application>

Occurrence

Optional, zero or one

```
<policy-store> {0 or 1}
  <applications> {0 or 1}
    <application> {1 or more}
    ...
```

Example

See <policy-store> for an example.

<attribute>

<attribute>

This element specifies an attribute of an application role.

Parent Element

<extended-attributes>

Child Elements

<name>, <values>

Occurrence

Required, one or more:

```
<policy-store> {0 or 1}
  <applications> {0 or 1}
    <application> {1 or more}
      <name> {1}
      <description> {0 or 1}
      <app-roles> {0 or 1}
        <app-role> {1 or more}
          <name> {1}
          <class> {1}
          <display-name> {0 or 1}
          <description> {0 or 1}
          <guid> {0 or 1}
          <unique-name> {0 or 1}
          <extended-attributes> {0 or 1}
            <attribute> {1 or more}
              <name> {1}
              <values> {1}
                <value> {1 or more}
              <guid> {0 or 1}
```

<class>

This element specifies several values depending on its location in the configuration file:

- Within the [<app-role>](#) element, `<class>` specifies the fully qualified name of the class implementing the application role.

```
<app-role>
...
  <class>oracle.security.jps.service.policystore.ApplicationRole</class>
```

- Within the [<member>](#) element, `<class>` specifies the fully qualified name of the class implementing the role member.

```
<app-role>
...
  <members>
    <member>
      ...
      <class>
        weblogic.security.principal.WLSUserImpl
      </class>
```

- Within the [<permission>](#) element (for granting permissions to a principal), `<class>` specifies the fully qualified name of the class implementing the permission. Values are case-insensitive.

```
<jazn-policy>
  <grant>
    ...
    <permissions>
      <permission>
        <class>java.io.FilePermission</class>
```

- Within the [<principal>](#) element (for granting permissions to a principal), it specifies the fully qualified name of the principal class, which is the class that is instantiated to represent a principal that is being granted a set of permissions.

```
<jazn-policy>
  <grant>
    ...
    <grantee>
      <principals>
        <principal>
          ...
          <class>oracle.security.jps.service.policystore.TestUser</class>
```

Parent Element

[<app-role>](#), [<member>](#), [<principal>](#), or [<permission>](#)

Child Elements

None

Occurrence

Required, one only

```
<policy-store> {0 or 1}
  <applications> {0 or 1}
    <application> {1 or more}
      <name> {1}
      <description> {0 or 1}
      <app-roles> {0 or 1}
        <app-role> {1 or more}
          <name> {1}
          <class> {1}
          ...
        <members> {0 or 1}
          <member> {1 or more}
            <name> {1}
            <class> {1}
            <uniqueusername> {0 or 1}
            <guid> {0 or 1}

<jazn-policy> {0 or 1}
  <grant> {0 or more}
    <description> {0 or 1}
    <grantee> {0 or 1}
      <principals> {0 or 1}
        <principal> {0 or more}
          <name> {1}
          <class> {1}
          ...
      <permissions> {0 or 1}
        <permission> {1 or more}
          <class> {1}
          <name> {0 or 1}
          <actions> {0 or 1}
```

Example

See `<jazn-policy>` and `<policy-store>` for examples.

<codesource>

This element specifies the URL of the code to which permissions are granted.

The policy configuration can also include a <principals> element, in addition to the <codesource> element. Both elements are children of a <grantee> element and they specify who or what the permissions in question are being granted to.

For variables that can be used in the specification of a <codesource> URL, see <url>.

Parent Element

<grantee>

Child Elements

<url>

Occurrence

Optional, zero or one

```
<jazn-policy> {0 or 1}
  <grant> {0 or more}
    <description> {0 or 1}
    <grantee> {0 or 1}
      <principals> {0 or 1}
        <principal> {0 or more}
          <name> {1}
          <class> {1}
          <uniqueName> {0 or 1}
          <guid> {0 or 1}
          <codesource> {0 or 1}
            <url> {1}
        <permissions> {0 or 1}
          <permission> {1 or more}
            <class> {1}
            <name> {0 or 1}
            <actions> {0 or 1}
```

Example

See <jazn-policy> for examples.

<credentials>

This element specifies the authentication password for a user. The credentials are, by default, in obfuscated form.

Parent Element

[<user>](#)

Child Elements

None

Occurrence

Optional, zero or one

```
<jazn-realm> {0 or 1}
  <realm> {0 or more}
    <name> {1}
    <users> {0 or 1}
      <user> {0 or more}
        <name> {1}
        <display-name> {0 or 1}
        <description> {0 or 1}
        <guid> {0 or 1}
        <credentials> {0 or 1}
```

Example

See [<jazn-realm>](#) for examples.

<description>

This element specifies a text string that provides textual information about an item. Depending on the parent element, the item can be an application role, application policy, permission grant, security role, or user.

Parent Element

<app-role>, <application>, <grant>, <role>, or <user>

Child Elements

None

Occurrence

Optional, zero or one

```
<jazn-realm> {0 or 1}
  <realm> {0 or more}
    <name> {1}
    <users> {0 or 1}
      <user> {0 or more}
      ...
      <description> {0 or 1}
      ...
    <roles> {0 or 1}
      <role> {0 or more}
      ...
      <description> {0 or 1}
      ...
  <policy-store> {0 or 1}
    <applications> {0 or 1}
      <application> {1 or more}
        <name> {1}
        <description> {0 or 1}
        <app-roles> {0 or 1}
          <app-role> {1 or more}
          ...
          <description> {0 or 1}
    ...
  <jazn-policy> {0 or 1}
    <grant> {0 or more}
      <description> {0 or 1}
      <grantee> {0 or 1}
```

Example

The fmwadmin user might have the following description:

```
<description>User with administrative privileges</description>
```

See <jazn-realm> for additional examples.

<display-name>

This element specifies the name of an item typically used by a GUI tool. Depending on the parent element, an item can be an application role, user, or enterprise group.

Parent Element

<app-role>, <role>, or <user>

Child Elements

None

Occurrence

Optional, zero or one

```
<jazn-realm> {0 or 1}
  <realm> {0 or more}
    <name> {1}
    <users> {0 or 1}
      <user> {0 or more}
        <name> {1}
        <display-name> {0 or 1}
        ...
    <roles> {0 or 1}
      <role> {0 or more}
        <name> {1}
        <display-name> {0 or 1}
        ...
  <policy-store> {0 or 1}
    <applications> {0 or 1}
      <application> {1 or more}
        <name> {1}
        <description> {0 or 1}
        <app-roles> {0 or 1}
          <app-role> {1 or more}
            <name> {1}
            <class> {1}
            <display-name> {0 or 1}
```

Example

The fmwadmin user might have the following display name:

```
<display-name>Administrator</display-name>
```

See <jazn-realm> for additional examples.

<extended-attributes>

This element specifies attributes of an application role.

Parent Element

<app-role>

Child Elements

<attribute>

Occurrence

Optional, zero or one

```
<policy-store> {0 or 1}
  <applications> {0 or 1}
    <application> {1 or more}
      <name> {1}
      <description> {0 or 1}
      <app-roles> {0 or 1}
        <app-role> {1 or more}
          <name> {1}
          <class> {1}
          <display-name> {0 or 1}
          <description> {0 or 1}
          <guid> {0 or 1}
          <uniqueName> {0 or 1}
          <extended-attributes> {0 or 1}
            <attribute> {1 or more}
              <name> {1}
              <values> {1}
                <value> {1 or more}
            </attribute>
          <members> {0 or 1}
            <member> {1 or more}
              <name> {1}
              <class> {1}
              <uniqueName> {0 or 1}
              <guid> {0 or 1}
            </member>
          </members>
        </app-roles>
      </application>
    </applications>
  </policy-store>
```

Example

```
<app-roles>
  <app-role>
    <name>Knight</name>
    <display-name>Fellowship For the Ring</display-name>
    <class>oracle.security.jps.service.policystore.ApplicationRole</class>
    <extended-attributes>
      <attribute>
        <name>SCOPE</name>
        <values>
          <value>Part-I</value>
        </values>
      </attribute>
    </extended-attributes>
  </app-role>
```

<grant>

This element specifies the recipient of the grant - a codesource, or a set of principals, or both- and the permissions assigned to it.

Parent Element

<jazn-policy>

Child Elements

<description>, <grantee>, <permissions>, <permission-sets>

Occurrence

Optional, zero or more

```
<jazn-policy> {0 or 1}
  <grant> {0 or more}
    <description> {0 or 1}
    <grantee> {0 or 1}
      <principals> {0 or 1}
        <principal> {0 or more}
          <name> {1}
          <class> {1}
          <uniquename> {0 or 1}
          <guid> {0 or 1}
        <codesource> {0 or 1}
          <url> {1}
      <permissions> {0 or 1}
        <permission> {1 or more}
          <class> {1}
          <name> {0 or 1}
          <actions> {0 or 1}
```

Example

See <jazn-policy> for examples.

<grantee>

This element, in conjunction with a parallel [<permissions>](#) element, specifies who or what the permissions are granted to: a set of principals, a codesource, or both.

Parent Element

[<grant>](#)

Child Elements

[<codesource>](#), [<principals>](#)

Occurrence

Optional, zero or one

```
<jazn-policy> {0 or 1}
  <grant> {0 or more}
    <description> {0 or 1}
    <grantee> {0 or 1}
      <principals> {0 or 1}
        <principal> {0 or more}
          <name> {1}
          <class> {1}
          <uniqueName> {0 or 1}
          <guid> {0 or 1}
      <codesource> {0 or 1}
        <url> {1}
    <permissions> {0 or 1}
      <permission> {1 or more}
        <class> {1}
        <name> {0 or 1}
        <actions> {0 or 1}
```

Example

See [<jazn-policy>](#) for examples.

<guid>

This element is for internal use only. It specifies a globally unique identifier (GUID) to reference the item.

Depending on the parent element, the item to be referenced may be an application role, application role member, principal, enterprise group, or user. It is typically used with an LDAP provider to uniquely identify the item (a user, for example). A GUID is sometimes generated and used internally by Oracle Platform Security Services, such as in migrating a user or role to a different security provider. It is not an item that you would set yourself.

Parent Element

<app-role>, <member>, <principal>, <role>, or <user>

Child Elements

None

Occurrence

Optional, zero or one

```
<jazn-realm> {0 or 1}
  <realm> {0 or more}
    <name> {1}
    <users> {0 or 1}
      <user> {0 or more}
        <name> {1}
        <display-name> {0 or 1}
        <description> {0 or 1}
        <guid> {0 or 1}
        <credentials> {0 or 1}
    <roles> {0 or 1}
      <role> {0 or more}
        <name> {1}
        <display-name> {0 or 1}
        <description> {0 or 1}
        <guid> {0 or 1}
    ...

<policy-store> {0 or 1}
  <applications> {0 or 1}
    <application> {1 or more}
      <name> {1}
      <description> {0 or 1}
      <app-roles> {0 or 1}
        <app-role> {1 or more}
          <name> {1}
          <class> {1}
          <display-name> {0 or 1}
          <description> {0 or 1}
          <guid> {0 or 1}
          <unique-name> {0 or 1}
          <extended-attributes> {0 or 1}
            <attribute> {1 or more}
              <name> {1}
              <values> {1}
```

```
        <value> {1 or more}
    <members> {0 or 1}
        <member> {1 or more}
            <name> {1}
            <class> {1}
            <uniqueusername> {0 or 1}
            <guid> {0 or 1}

<jazn-policy> {0 or 1}
    <grant> {0 or more}
        <description> {0 or 1}
        <grantee> {0 or 1}
            <principals> {0 or 1}
                <principal> {0 or more}
                    <name> {1}
                    <class> {1}
                    <uniqueusername> {0 or 1}
                    <guid> {0 or 1}
            <codesource> {0 or 1}
            <url> {1}
    ...
```

Example

See [<jazn-realm>](#) for examples.

<jazn-data>

This element specifies the top-level element in the `system-jazn-data.xml` file-based policy store.

Attributes

Name	Description
<code>schema-major-version</code>	Specifies the major version number of the <code>system-jazn-data.xml</code> XSD. The value of this attribute is fixed at 11 for use with Oracle Fusion Middleware 11g.
<code>schema-minor-version</code>	Specifies the minor version number of the <code>system-jazn-data.xml</code> XSD. The value of this attribute is fixed at 0 for use with the Oracle Fusion Middleware 11.1.1 implementation.

Parent Element

n/a

Child Elements

`<jazn-policy>`, `<jazn-realm>`, `<policy-store>`

Occurrence

Required, one only

```
<jazn-data ... > {1}
  <jazn-realm> {0 or 1}
  ...
  <policy-store> {0 or 1}
  ...
  <jazn-policy> {0 or 1}
  ...
```

Example

```
<jazn-data
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation=
    "http://xmlns.oracle.com/oracleas/schema/jazn-data-11_0.xsd">
  ...
</jazn-data
```


<jazn-policy>

This element specifies policy grants that associate grantees (principals or codesources) with permissions.

This element can appear in two different locations in the `system-jazn-data.xml` file:

- Under the `<jazn-data>` element, it specifies global policies.
- Under the `<application>` element, it specifies application-level policies.

Parent Element

`<application>` or `<jazn-data>`

Child Elements

`<grant>`

Occurrence

Optional, zero or one

```
<jazn-data> {1}
  <jazn-policy> {0 or 1}
    <grant> {0 or more}
      <description> {0 or 1}
      <grantee> {0 or 1}
        <principals> {0 or 1}
        ...
        <codesource> {0 or 1}
          <url> {1}
      <permissions> {0 or 1}
        <permission> {1 or more}
          <class> {1}
          <name> {0 or 1}
          <actions> {0 or 1}
```

Example

Example B-1 <jazn-policy>

```
<jazn-policy>
  <grant>
    <grantee>
      <principals>
        <principal>
          <class>
            oracle.security.jps.service.policystore.TestUser
          </class>
          <name>jack</name>
        </principal>
        <principal>
          <class>
            oracle.security.jps.service.policystore.TestUser
          </class>
          <name>jill</name>
```

```
        </principal>
    </principals>
    <codesource>
    <url>file:${oracle.deployed.app.dir}/<MyApp>${oracle.deployed.app.ext}</url>
    </codesource>
    </grantee>
    <permissions>
    <permission>
    <class>oracle.security.jps.JpsPermission</class>
    <name>getContext</name>
    </permission>
    <permission>
    <class>java.io.FilePermission</class>
    <name>/foo</name>
    <actions>read,write</actions>
    </permission>
    </permissions>
    </grant>
</jazn-policy>
```

Example B-2 <jazn-policy>

```
<jazn-policy>
    <grant>
    <grantee>
    <principals>
    <principal>
    <class>
    oracle.security.jps.service.policystore.TestAdminRole
    </class>
    <name>Farm=farm1,name=FullAdministrator</name>
    </principal>
    </principals>
    <codesource>
    <url>file://some-file-path</url>
    </codesource>
    </grantee>
    <permissions>
    <permission>
    <class>javax.management.MBeanPermission</class>
    <name>
    oracle.as.management.topology.mbeans.InstanceOperations#getAttribute
    </name>
    <actions>invoke</actions>
    </permission>
    </permissions>
    </grant>
</jazn-policy>
```

<jazn-realm>

This element specifies security realms and the users and enterprise groups (as opposed to application-level roles) they include, and is the top-level element for user and role information

Attribute

Name	Description
default	Specifies which of the realms defined under this element is the default realm. The value of this attribute must match a <name> value under one of the <realm> subelements. Values: string Default: n/a (required)

Parent Element

<jazn-data>

Child Elements

<realm>

Occurrence

Optional, zero or one

```
<jazn-data> {1}
  <jazn-realm> {0 or 1}
    <realm> {0 or more}
      <name> {1}
      <users> {0 or 1}
      ...
      <roles> {0 or 1}
      ...
```

Example

```
<jazn-data ... >
  ...
  <jazn-realm default="jazn.com">
    <realm>
      <name>jazn.com</name>
      <users>
        <user deactivated="true">
          <name>anonymous</name>
          <guid>61FD29C0D47E11DABF9BA765378CF9F3</guid>
          <description>The default guest/anonymous user</description>
        </user>
        <user>
          <name>developer1</name>
          <credentials>!password</credentials>
        </user>
        <user>
          <name>developer2</name>
```

```
        <credentials>!password</credentials>
    </user>
    <user>
        <name>manager1</name>
        <credentials>!password</credentials>
    </user>
    <user>
        <name>manager2</name>
        <credentials>!password</credentials>
    </user>
    <!-- these are for testing the admin role hierachy. -->
    <user>
        <name>farm-admin</name>
        <credentials>!password</credentials>
    </user>
    <user>
        <name>farm-monitor</name>
        <credentials>!password</credentials>
    </user>
    <user>
        <name>farm-operator</name>
        <credentials>!password</credentials>
    </user>
    <user>
        <name>farm-auditor</name>
        <credentials>!password</credentials>
    </user>
    <user>
        <name>farm-auditviewer</name>
        <credentials>!password</credentials>
    </user>
</users>
<roles>
    <role>
        <name>users</name>
        <guid>31FD29C0D47E11DABF9BA765378CF9F7</guid>
        <display-name>users</display-name>
        <description>users role for rmi/ejb access</description>
    </role>
    <role>
        <name>ascontrol_appadmin</name>
        <guid>51FD29C0D47E11DABF9BA765378CF9F7</guid>
        <display-name>ASControl App Admin Role</display-name>
        <description>
            Application Administrative role for ASControl
        </description>
    </role>
    <role>
        <name>ascontrol_monitor</name>
        <guid>61FD29C0D47E11DABF9BA765378CF9F7</guid>
        <display-name>ASControl Monitor Role</display-name>
        <description>Monitor role for ASControl</description>
    </role>
    <role>
        <name>developers</name>
        <members>
            <member>
                <type>user</type>
                <name>developer1</name>
            </member>
        </members>
    </role>
</roles>
```

```
        <member>
          <type>user</type>
          <name>developer2</name>
        </member>
      </members>
    </role>
    <role>
      <name>managers</name>
      <members>
        <member>
          <type>user</type>
          <name>manager1</name>
        </member>
        <member>
          <type>user</type>
          <name>manager2</name>
        </member>
      </members>
    </role>
  </roles>
</realm>
</jazn-realm>
...
</jazn-data>
```

<matcher-class>

This element specifies the fully qualified name of the class within a resource type; queries for resources of this type delegate to this matcher class. Values are case-sensitive.

Parent Element

[<resource-type>](#)

Child Elements

None

Occurrence

Optional, zero or more

```
<policy-store> {0 or 1}
  <applications> {0 or 1}
    <application> {1 or more}
      <name> {1}
      <description> {0 or 1}
      <app-roles> {0 or 1}
      ...
      <role-categories> {0 or 1}
      ...
      <resource-types> {0 or 1}
        <resource-type> {1 or more}
          <name> {1}
          <display-name> {1}
          <description> {0 or 1}
          <provider-name> {1}
          <matcher-class> {1}
          <actions-delimiter> {1}
          <actions> {1 or more}
```

Example

For an example, see [<resource-type>](#).

<member>

This element specifies the members of a set, such as a <role> or an <app-role> element:

- When under a <role> element, it specifies a member of the enterprise group. A member can be a user or another enterprise group. The <name> subelement specifies the name of the member, and the <type> subelement specifies whether the member type (a user or an enterprise group).
- When under an <app-role> element, it specifies a member of the application role. A member can be a user, an enterprise group, or an application role. The <name> subelement specifies the name of the member, and the <class> subelement specifies the class that implements it. The member type is determined through the <class> element.

Optional subelements include <uniqueusername> and <guid>, which specify a unique name and unique global identifier; these optional subelements are for internal use only.

Parent Element

<members>

Child Elements

- When under a <role> element, the <member> element has the following child elements: <name>, <type>
- When under an <app-role> element, the <member> element has the following child elements: <name>, <class>, <uniqueusername>, <guid>

Occurrence

Optional, zero or more

```

<jazn-realm> {0 or 1}
  <realm> {0 or more}
    <name> {1}
    <users> {0 or 1}
    ...
    <roles> {0 or 1}
      <role> {0 or more}
        <name> {1}
        <display-name> {0 or 1}
        <description> {0 or 1}
        <guid> {0 or 1}
        <members> {0 or 1}
          <member> {0 or more}
            <type> {1}
            <name> {1}
          <owners> {0 or 1}
            <owner> {0 or more}
              <type> {1}
              <name> {1}
        <policy-store> {0 or 1}

```

```
<applications> {0 or 1}
  <application> {1 or more}
    <name> {1}
    <description> {0 or 1}
    <app-roles> {0 or 1}
      <app-role> {1 or more}
        <name> {1}
        <class> {1}
        <display-name> {0 or 1}
        <description> {0 or 1}
        <guid> {0 or 1}
        <uniqueusername> {0 or 1}
        <extended-attributes> {0 or 1}
        ...
      <members> {0 or 1}
        <member> {1 or more}
          <name> {1}
          <class> {1}
          <uniqueusername> {0 or 1}
          <guid> {0 or 1}
```

Example

See [<jazn-realm>](#) and [<policy-store>](#) for examples.

<member-resource>

This element specifies resources for a permission set.

Parent Element

[<member-resources>](#)

Child Elements

[<resource-name>](#), [<type-name-ref>](#), [<actions>](#)

Occurrence

Required within [<member-resources>](#), one or more.

```
<policy-store> {0 or 1}
  <applications> {0 or 1}
    <application> {1 or more}
      <name> {1}
      <description> {0 or 1}
      <app-roles> {0 or 1}
      ...
      <role-categories>
      ...
      <permission-sets>
        <permission-set>
          <name>
          <member-resources>
            <member-resource>
              <resource-name>
              <type-name-ref>
              <actions>
```

Example

For an example, see [<permission-set>](#).

<member-resources>

This element specifies a set of member resources.

Parent Element

[<permission-set>](#)

Child Elements

[<member-resource>](#)

Occurrence

Required within [<permission-sets>](#); one or more.

```
<policy-store> {0 or 1}
  <applications> {0 or 1}
    <application> {1 or more}
      <name> {1}
      <description> {0 or 1}
      <app-roles> {0 or 1}
      ...
      <role-categories>
      ...
      <permission-sets>
        <permission-set>
          <name>
            <member-resources>
              <member-resource>
                <resource-name>
                <type-name-ref>
                <actions>
```

Example

For an example, see [<permission-set>](#).

<members>

This element specifies a set of members.

Parent Element

<role>, <app-role>

Child Elements

<member>

Occurrence

Optional, zero or one

```
<jazn-realm> {0 or 1}
  <realm> {0 or more}
    <name> {1}
    <users> {0 or 1}
    ...
    <roles> {0 or 1}
      <role> {0 or more}
        <name> {1}
        <display-name> {0 or 1}
        <description> {0 or 1}
        <guid> {0 or 1}
        <members> {0 or 1}
          <member> {0 or more}
            <type> {1}
            <name> {1}
        <owners> {0 or 1}
          <owner> {0 or more}
            <type> {1}
            <name> {1}

<policy-store> {0 or 1}
  <applications> {0 or 1}
    <application> {1 or more}
      <name> {1}
      <description> {0 or 1}
      <app-roles> {0 or 1}
        <app-role> {1 or more}
          <name> {1}
          <class> {1}
          <display-name> {0 or 1}
          <description> {0 or 1}
          <guid> {0 or 1}
          <uniqueusername> {0 or 1}
          <extended-attributes> {0 or 1}
          ...
          <members> {0 or 1}
            <member> {1 or more}
              <name> {1}
              <class> {1}
              <uniqueusername> {0 or 1}
              <guid> {0 or 1}
```

<members>

Example

See [<jazn-realm>](#) and [<policy-store>](#) for examples.

<name>

This element has different uses, depending on its location in the file:

- Within the `<app-role>` element, it specifies the name of an application-level role in the policy configuration. For example:

```
<name>Farm=farm1,name=FullAdministrator</name>
```

Or a simpler example:

```
<name>Myrolename</name>
```

- Within the `<application>` element, it specifies the policy context identifier. Typically, this is the name of the application during deployment.
- Within the `<attribute>` element, it specifies the name of an additional attribute for the application-level role.
- Within the `<member>` element, it specifies the name of a member of an enterprise group or application role (depending on where the `<member>` element is located). For example, if the `fmwadmin` user is to be a member of the role:

```
<name>fmwadmin</name>
```

- Within the `<owner>` element, it specifies the name of an owner of an enterprise group. For example:

```
<name>mygroupowner</name>
```

- Within the `<permission>` element, as applicable, it can specify the name of a permission that is meaningful to the permission class. Values are case-sensitive. For example:

```
<name>
  oracle.as.management.topology.mbeans.InstanceOperations#getAttribute
</name>
```

Or:

```
<name>getContext</name>
```

- Within the `<principal>` element (for granting permissions to a principal), it specifies the name of a principal within the given realm. For example:

```
<name>Administrators</name>
```

- Within the `<realm>` element, it specifies the name of a realm. For example:

```
<name>jazn.com</name>
```

- Within the `<role>` element, it specifies the name of an enterprise group in a realm. For example:

```
<name>Administrators</name>
```

- Within the `<user>` element, it specifies the name of a user in a realm. For example:

```
<name>fmwadmin</name>
```

<name>

- Within the <resource-type> element, it specifies the name of a resource type and is required. For example:

```
<name>restype1</name>
```

Parent Element

<app-role>, <application>, <attribute>, <member>, <owner>, <permission>, <principal>, <realm>, <role>, or <user>

Child Elements

None

Occurrence

Required within any parent element other than <permission>, one only; optional within <permission>, zero or one

```
<jazn-realm> {0 or 1}
  <realm> {0 or more}
    <name> {1}
    <users> {0 or 1}
      <user> {0 or more}
        <name> {1}
        <display-name> {0 or 1}
        <description> {0 or 1}
        <guid> {0 or 1}
        <credentials> {0 or 1}
    <roles> {0 or 1}
      <role> {0 or more}
        <name> {1}
        <display-name> {0 or 1}
        <description> {0 or 1}
        <guid> {0 or 1}
        <members> {0 or 1}
          <member> {0 or more}
            <type> {1}
            <name> {1}
        <owners> {0 or 1}
          <owner> {0 or more}
            <type> {1}
            <name> {1}

<policy-store> {0 or 1}
  <applications> {0 or 1}
    <application> {1 or more}
      <name> {1}
      <description> {0 or 1}
      <app-roles> {0 or 1}
        <app-role> {1 or more}
          <name> {1}
          <class> {1}
          <display-name> {0 or 1}
          <description> {0 or 1}
          <guid> {0 or 1}
          <unique-name> {0 or 1}
          <extended-attributes> {0 or 1}
            <attribute> {1 or more}
              <name> {1}
```

```
        <values> {1}
            <value> {1 or more}
    <members> {0 or 1}
        <member> {1 or more}
            <name> {1}
            <class> {1}
            <uniquename> {0 or 1}
            <guid> {0 or 1}

<jazn-policy> {0 or 1}
    <grant> {0 or more}
        <description> {0 or 1}
        <grantee> {0 or 1}
            <principals> {0 or 1}
                <principal> {0 or more}
                    <name> {1}
                    <class> {1}
                    <uniquename> {0 or 1}
                    <guid> {0 or 1}
                <codesource> {0 or 1}
                    <url> {1}
            <permissions> {0 or 1}
                <permission> {1 or more}
                    <class> {1}
                    <name> {0 or 1}
                    <actions> {0 or 1}
```

Example

```
<application>
  <name>peanuts</name>
  <app-roles>
    <app-role>
      <name>snoopy</name>
      <display-name>application role snoopy</display-name>
      <class>oracle.security.jps.service.policystore.ApplicationRole</class>
      <members>
        <member>
          .....

```

See [<jazn-policy>](#), [<jazn-realm>](#), and [<policy-store>](#) for examples.

<owner>

This element specifies the owner of the enterprise group, where an owner has administrative authority over the role.

An owner is a user or another enterprise group. The <type> subelement specifies the owner's type. The concept of role (group) owners specifically relates to BPEL or Oracle Internet Directory functionality. For example, in BPEL, a role owner has the capability to create and update workflow rules for the role.

Note: To create a group owner in Oracle Internet Directory, use the Oracle Delegated Administration Services. For external (third-party) LDAP servers, set values for the group's owner attribute through `ldapmodify` or tools of the particular directory server.

Parent Element

<owners>

Child Elements

<name>, <type>

Occurrence

Optional, zero or more

```
<jazn-realm> {0 or 1}
  <realm> {0 or more}
    <name> {1}
    <users> {0 or 1}
    ...
    <roles> {0 or 1}
      <role> {0 or more}
        <name> {1}
        <display-name> {0 or 1}
        <description> {0 or 1}
        <guid> {0 or 1}
        <members> {0 or 1}
          <member> {0 or more}
            <type> {1}
            <name> {1}
        <owners> {0 or 1}
          <owner> {0 or more}
            <type> {1}
            <name> {1}
```


<owners>

This element specifies a set of owners.

Parent Element

<role>

Child Elements

<owner>

Occurrence

Optional, zero or one

```
<jazn-realm> {0 or 1}
  <realm> {0 or more}
    <name> {1}
    <users> {0 or 1}
    ...
    <roles> {0 or 1}
      <role> {0 or more}
        <name> {1}
        <display-name> {0 or 1}
        <description> {0 or 1}
        <guid> {0 or 1}
        <members> {0 or 1}
          <member> {0 or more}
            <type> {1}
            <name> {1}
        <owners> {0 or 1}
          <owner> {0 or more}
            <type> {1}
            <name> {1}
```

<permission>

This element specifies the permission to grant to grantees, where a grantee is a set of principals, a codesource, or both, as part of a policy configuration.

Parent Element

<permissions>

Child Elements

<actions>, <class>, <name>

Occurrence

Required within parent element, one or more

```
<jazn-policy> {0 or 1}
  <grant> {0 or more}
    <description> {0 or 1}
    <grantee> {0 or 1}
      <principals> {0 or 1}
        <principal> {0 or more}
          <name> {1}
          <class> {1}
          <uniquename> {0 or 1}
          <guid> {0 or 1}
        <codesource> {0 or 1}
          <url> {1}
      <permissions> {0 or 1}
        <permission> {1 or more}
          <class> {1}
          <name> {0 or 1}
          <actions> {0 or 1}
```

Example

See <jazn-policy> for examples.

<permissions>

This element specifies a set of permissions.

The <permissions> element (used in conjunction with a parallel <grantee> element) specifies the permissions being granted, through a set of <permission> subelements.

Note: The `system-jazn-data.xml` schema definition does not specify this as a required element, but the Oracle Platform Security runtime implementation requires its use within any <grant> element.

Parent Element

<grant>

Child Elements

<permission>

Occurrence

Optional, zero or one

```
<jazn-policy> {0 or 1}
  <grant> {0 or more}
    <description> {0 or 1}
    <grantee> {0 or 1}
      <principals> {0 or 1}
        <principal> {0 or more}
          <name> {1}
          <class> {1}
          <uniquename> {0 or 1}
          <guid> {0 or 1}
        <codesource> {0 or 1}
          <url> {1}
      <permissions> {0 or 1}
        <permission> {1 or more}
          <class> {1}
          <name> {0 or 1}
          <actions> {0 or 1}
```

Example

See <jazn-policy> for examples.

<permission-set>

A permission set or entitlement specifies a set of permissions.

Parent Element

<permission-sets>

Child Elements

<name>

Occurrence

Optional, zero or more

```
<policy-store> {0 or 1}
  <applications> {0 or 1}
    <application> {1 or more}
      <name> {1}
      <description> {0 or 1}
      <app-roles> {0 or 1}
      ...
      <role-categories>
      ...
      <permission-sets>
        <permission-set>
          <name>
          <member-resources>
            <member-resource>
              <resource-name>
              <type-name-ref>
              <actions>
```

Example

The following fragment illustrates the configuration of a permission set (or entitlement):

```
<permission-sets>
  <permission-set>
    <name>permsetName</name>
    <member-resources>
      <member-resource>
        <type-name-ref>TaskFlowResourceType</type-name-ref>
        <resource-name>resource1</resource-name>
        <actions>customize,view</actions>
      </member-resource>
    </member-resources>
  </permission-set>
</permission-sets>
```

Note the following points about a permission set:

- The actions specified in a <member-resource> must match one or more of the actions specified for the resource type that is referenced through <resource-name-ref>.
- A <member-resources> can have multiple <member-resource> elements in it.

- A permission set must have at least one resource.
- Permission sets can exist without necessarily being referenced in any grants, that is, without granting them to any principal.

In addition, the following strings in a permission set entry conform to the case sensitivity rules:

- The name is case insensitive.
- The description string is case insensitive.
- The display name is case insensitive.

<permission-sets>

This element specifies a set of permission sets.

Parent Element

[<application>](#)

Child Elements

[<permission-set>](#)

Occurrence

Optional, zero or more

```
<policy-store> {0 or 1}
  <applications> {0 or 1}
    <application> {1 or more}
      <name> {1}
      <description> {0 or 1}
      <app-roles> {0 or 1}
      ...
      <role-categories>
      ...
      <permission-sets>
        <permission-set>
          <name>
          <member-resources>
            <member-resource>
              <resource-name>
              <type-name-ref>
              <actions>
```

Example

For an example, see [<permission-set>](#).

<policy-store>

This element configures application-level policies, through an <applications> subelement. Under the <applications> element is an <application> subelement for each application that is to have application-level policies. The policies are specified through a <jazn-policy> subelement of each <application> element.

Note: The <jazn-principal-classes> and <jazn-permission-classes> elements and their subelements may appear in the system-jazn-data.xml schema definition as subelements of <policy-store>, but are for backward compatibility only.

Parent Element

<jazn-data>

Child Elements

<applications>

Occurrence

Optional, zero or one

```
<jazn-data> {1}
  <policy-store> {0 or 1}
    <applications> {0 or 1}
      <application> {1 or more}
      ...
```

Example

```
<jazn-data ... >
  ...
  <policy-store>
    <!-- application policy -->
    <applications>
      <application>
        <name>policyOnly</name>
        <jazn-policy>
          ...
        </jazn-policy>
      </application>
      <application>
        <name>roleOnly</name>
        <app-roles>
          <app-role>
            <name>Fellowship</name>
            <display-name>Fellowship of the Ring</display-name>
            <class>
              oracle.security.jps.service.policystore.ApplicationRole
            </class>
          </app-role>
          <app-role>
            <name>King</name>
```

```
        <display-name>Return of the King</display-name>
        <class>
            oracle.security.jps.service.policystore.ApplicationRole
        </class>
    </app-role>
</app-roles>
</application>
<application>
    <app-roles>
        <app-role>
            <name>Farm=farm1,name=FullAdministrator</name>
            <display-name>farm1.FullAdministrator</display-name>
            <guid>61FD29C0D47E11DABF9BA765378CF9F2</guid>
            <class>
                oracle.security.jps.service.policystore.ApplicationRole
            </class>
            <members>
                <member>
                    <class>
oracle.security.jps.internal.core.principals.JpsXmlEnterpriseRoleImpl
                    </class>
                    <name>admin</name>
                </member>
            </members>
        </app-role>
    </app-roles>
    <jazn-policy>
        ...
    </jazn-policy>
</application>
...
</applications>
</policy-store>
....
</jazn-data
```

See [<jazn-policy>](#) for examples of that element.

<principal>

This element specifies a principal being granted the permissions specified in a [<permissions>](#) element as part of a policy configuration. Required under [<principals>](#).

Subelements specify the name of the principal and the class that implements it, and optionally specify a unique name and unique global identifier (the latter two for internal use only).

For details about how principal names can be compared, see [Section 2.7, "Principal Name Comparison Logic."](#)

Parent Element

[<principals>](#)

Child Elements

[<class>](#), [<guid>](#), [<name>](#), [<uniquename>](#)

Occurrence

Optional, zero or more

```
<jazn-policy> {0 or 1}
  <grant> {0 or more}
    <description> {0 or 1}
    <grantee> {0 or 1}
      <principals> {0 or 1}
        <principal> {0 or more}
          <name> {1}
          <class> {1}
          <uniquename> {0 or 1}
          <guid> {0 or 1}
        <codesource> {0 or 1}
          <url> {1}
      <permissions> {0 or 1}
        <permission> {1 or more}
          <class> {1}
          <name> {0 or 1}
          <actions> {0 or 1}
```

Example

See [<jazn-policy>](#) for examples.

<principals>

This element specifies a set of principals.

For policy configuration, a <principals> element and/or a <codesource> element are used under a <grantee> element to specify who or what the permissions in question are being granted to. A <principals> element specifies a set of principals being granted the permissions.

For a subject to be granted these permissions, the subject should include all the specified principals.

Parent Element

<grantee>

Child Elements

<principal>

Occurrence

Optional, zero or one

```
<jazn-policy> {0 or 1}
  <grant> {0 or more}
    <description> {0 or 1}
    <grantee> {0 or 1}
      <principals> {0 or 1}
        <principal> {0 or more}
          <name> {1}
          <class> {1}
          <uniquename> {0 or 1}
          <guid> {0 or 1}
        <codesource> {0 or 1}
          <url> {1}
      <permissions> {0 or 1}
        <permission> {1 or more}
          <class> {1}
          <name> {0 or 1}
          <actions> {0 or 1}
```

Example

See <jazn-policy> for examples.

<provider-name>

This element specifies the name of a resource type provider. The resource resides in a location external to the OPSS policy store. Values are case-insensitive.

Parent Element

<resource-type>

Child Elements

None

Occurrence

Optional, zero or more

```
<policy-store> {0 or 1}
  <applications> {0 or 1}
    <application> {1 or more}
      <name> {1}
      <description> {0 or 1}
      <app-roles> {0 or 1}
      ...
      <role-categories>
      ...
      <resource-types>
        <resource-type>
          <name>
          <display-name>
          <description>
          <provider-name>
          <matcher-class>
          <actions-delimiter>
          <actions>
```

Example

For an example, see <resource-type>.

<realm>

This element specifies a security realm, and the users and roles that belong to the realm.

Parent Element

<jazn-realm>

Child Elements

<name>, <roles>, <users>

Occurrence

Optional, zero or more

```
<jazn-realm> {0 or 1}
  <realm> {0 or more}
    <name> {1}
    <users> {0 or 1}
    ...
    <roles> {0 or 1}
    ...
```

Example

See <jazn-realm> for an example.

<resource>

This element specifies an application resource and contains information about the resource.

Parent Element

<resources>

Child Elements

<name>, <description>, <display-name>, <type-name-ref>.

Occurrence

Required under <resources>.

```
<resources> (0 or more)
  <resource> (1 or more)
    <name> (1)
    <display-name> (1)
    <description> {0 or 1}
    <type-name-ref> (1)
```

Example

The following fragment illustrates the configuration of a resource (instance):

```
<resources>
  <resource>
    <name>resource1</name>
    <display-name>Resource1DisplayName</display-name>
    <description>Resource1 Description</description>
    <type-name-ref>TaskFlowResourceType</type-name-ref>
  </resource>
</resources>
```

Note the following points about case sensitivity of various strings in a resource entry:

- The name is case sensitive.
- The description string is case insensitive.
- The display name is case insensitive.

<resources>

This element specifies a collection of application resources.

Parent Element

[<application>](#)

Child Elements

[<resource>](#)

Occurrence

Optional, zero or more

```
<resources> (0 or more)
  <resource> (1 or more)
    <name> (1)
    <display-name> (1)
    <description> {0 or 1}
    <type-name-ref> (1)
```

Example

For an example, see [<resource>](#).

<resource-name>

This element specifies a member resource in a permission set. Values are case-sensitive.

Parent Element

[<member-resource>](#)

Child Elements

None

Occurrence

Optional, zero or more

```
<policy-store> {0 or 1}
  <applications> {0 or 1}
    <application> {1 or more}
      <name> {1}
      <description> {0 or 1}
      <app-roles> {0 or 1}
      ...
      <role-categories>
      ...
      <permission-sets>
        <permission-set>
          <name>
          <member-resources>
            <member-resource>
              <resource-name>
              <type-name-ref>
              <actions>
```

Example

For an example, see [<permission-set>](#).

<resource-type>

This element specifies the type of a secured artifact, such as a flow, a job, or a web service. Values are case-insensitive.

Parent Element

<resource-types>

Child Elements

<name>, <display-name>, <description>, <actions>,
<actions-delimiter>, <matcher-class>, <provider-name>.

Occurrence

Optional, zero or more

```
<policy-store> {0 or 1}
  <applications> {0 or 1}
    <application> {1 or more}
      <name> {1}
      <description> {0 or 1}
      <app-roles> {0 or 1}
      ...
      <role-categories>
      ...
      <resource-types>
        <resource-type>
          <name>
          <display-name>
          <description>
          <provider-name>
          <matcher-class>
          <actions-delimiter>
          <actions>
```

Example

The following fragment illustrates the configuration of a resource type:

```
<resource-types>
  <resource-type>
    <name>TaskFlowResourceType</name>
    <display-name>TaskFlowResourceType_disp</display-name>
    <description>Resource Type for Task Flow</description>
    <provider-name>resTypeProv</provider-name>
    <matcher-class>
oracle.adf.controller.security.TaskFlowPermission</matcher-class>
    <actions-delimiter>,</actions-delimiter>
    <actions>customize,view</actions>
  </resource-type>
</resource-types>
```

The following points apply to the specification of a resource type:

- The name is required and case insensitive.

- The provider name is optional and case insensitive. A provider is typically used when there are resources managed in an external store, that is, in a store other than the OPSS domain policy store.

When specified, the class in a <provider-name> element is used as a resource finder; queries for resources of this type (via the `ResourceManager` search APIs) delegate to this matcher class instead of using the built-in resource finder against the OPSS domain policy store.

- The matcher class name is required and case sensitive.
- The description string is optional and case insensitive.
- The display name is optional and case insensitive.
- The action string is optional and case sensitive. The list of actions in a resource type can be empty. An empty action list indicates that the actions on instances of the resource type are determined externally and are opaque to OPSS.

<resource-types>

This element specifies a set of resource types.

Parent Element

<application>

Child Elements

<resource-type>

Occurrence

Optional, zero or more

```
<policy-store> {0 or 1}
  <applications> {0 or 1}
    <application> {1 or more}
      <name> {1}
      <description> {0 or 1}
      <app-roles> {0 or 1}
      ...
      <role-categories>
      ...
      <resource-types>
        <resource-type>
          <name>
          <display-name>
          <description>
          <provider-name>
          <matcher-class>
          <actions-delimiter>
          <actions>
```

Example

For an example, see <resource-type>.

<role>

This element specifies an enterprise security role, as opposed to an application-level role, and the members (and optionally owners) of that role.

Parent Element

<roles>

Child Elements

<description>, <display-name>, <guid>, <members>, <name>, <owners>

Occurrence

Optional, zero or more

```
<jazn-realm> {0 or 1}
  <realm> {0 or more}
    <name> {1}
    <users> {0 or 1}
      <user> {0 or more}
        <name> {1}
        <display-name> {0 or 1}
        <description> {0 or 1}
        <guid> {0 or 1}
        <credentials> {0 or 1}
    <roles> {0 or 1}
      <role> {0 or more}
        <name> {1}
        <display-name> {0 or 1}
        <description> {0 or 1}
        <guid> {0 or 1}
        <members> {0 or 1}
          <member> {0 or more}
            <type> {1}
            <name> {1}
        <owners> {0 or 1}
          <owner> {0 or more}
            <type> {1}
            <name> {1}
```

Example

See <jazn-realm> for examples.

<role-categories>

This element specifies the parent element of [<role-category>](#) elements.

Parent Element

[<application>](#)

Child Elements

[<role-category>](#)

Occurrence

Optional, zero or one

```
<application> {1 or more}
  <name> {1}
  <description> {0 or 1}
  <app-roles> {0 or 1}
    <app-role> {1 or more}
      <name> {1}
      <class> {1}
      <display-name> {0 or 1}
      <description> {0 or 1}
      <guid> {0 or 1}
      <uniqueusername> {0 or 1}
      <extended-attributes> {0 or 1}
        <attribute> {1 or more}
          <name> {1}
          <values> {1}
            <value> {1 or more}
      <members> {0 or 1}
        <member> {1 or more}
          <name> {1}
          <class> {1}
          <uniqueusername> {0 or 1}
          <guid> {0 or 1}
  <role-categories>
    <role-category>
      <name>
      <description>
      <display-name>
```

Example

See [Section 20.3.3.1, "Using the Method checkPermission"](#) for an example.

<role-category>

This element specifies a category, that is, a flat set of application roles.

Parent Element

<role-categories>

Child Elements

<name>, <display-name>, <description>, <members>

Occurrence

Optional, zero or one

```
<application> {1 or more}
  <name> {1}
  <description> {0 or 1}
  <app-roles> {0 or 1}
    <app-role> {1 or more}
      <name> {1}
      <class> {1}
      <display-name> {0 or 1}
      <description> {0 or 1}
      <guid> {0 or 1}
      <unique-name> {0 or 1}
      <extended-attributes> {0 or 1}
        <attribute> {1 or more}
          <name> {1}
          <values> {1}
            <value> {1 or more}
      <members> {0 or 1}
        <member> {1 or more}
          <name> {1}
          <class> {1}
          <unique-name> {0 or 1}
          <guid> {0 or 1}
  <role-categories>
    <role-category>
      <name>
      <description>
      <display-name>
      <members>
```

Example

See [Section 20.3.3.1, "Using the Method checkPermission"](#) for an example.

<role-name-ref>

This element specifies an application role within a role category.

Parent Element

<members>

Child Elements

None

Occurrence

Optional, zero or one

```
<application> {1 or more}
  <name> {1}
  <description> {0 or 1}
  <app-roles> {0 or 1}
    <app-role> {1 or more}
      <name> {1}
      <class> {1}
      <display-name> {0 or 1}
      <description> {0 or 1}
      <guid> {0 or 1}
      <uniqueusername> {0 or 1}
      <extended-attributes> {0 or 1}
        <attribute> {1 or more}
          <name> {1}
          <values> {1}
            <value> {1 or more}
      <members> {0 or 1}
        <member> {1 or more}
          <name> {1}
          <class> {1}
          <uniqueusername> {0 or 1}
          <guid> {0 or 1}
  <role-categories>
    <role-category>
      <name>
      <description>
      <members>
        <role-name-ref>
```

<roles>

This element specifies a set of enterprise security roles that belong to a security realm.

Parent Element

<realm>

Child Elements

<role>

Occurrence

Optional, zero or one

```
<jazn-realm> {0 or 1}
  <realm> {0 or more}
    <name> {1}
    <users> {0 or 1}
      <user> {0 or more}
        <name> {1}
        <display-name> {0 or 1}
        <description> {0 or 1}
        <guid> {0 or 1}
        <credentials> {0 or 1}
    <roles> {0 or 1}
      <role> {0 or more}
        <name> {1}
        <display-name> {0 or 1}
        <description> {0 or 1}
        <guid> {0 or 1}
        <members> {0 or 1}
          <member> {0 or more}
            <type> {1}
            <name> {1}
        <owners> {0 or 1}
          <owner> {0 or more}
            <type> {1}
            <name> {1}
```

Example

See <jazn-realm> for an example.

<type>

<type>

This element specifies the type of an enterprise group member or role owner: specifically, whether the member or owner is a user or another role:

```
<type>user</type>
```

Or:

```
<type>role</type>
```

Parent Element

[<member>](#) or [<owner>](#)

Child Elements

None

Occurrence

Required, one only

```
<jazn-realm> {0 or 1}
  <realm> {0 or more}
    <name> {1}
    <users> {0 or 1}
    ...
    <roles> {0 or 1}
      <role> {0 or more}
        <name> {1}
        <display-name> {0 or 1}
        <description> {0 or 1}
        <guid> {0 or 1}
        <members> {0 or 1}
          <member> {0 or more}
            <type> {1}
            <name> {1}
        <owners> {0 or 1}
          <owner> {0 or more}
            <type> {1}
            <name> {1}
```

Example

See [<jazn-realm>](#) for examples.

<type-name-ref>

This element specifies the resource type of a resource.

Parent Element

<member-resource>, <resource>

Child Elements

None

Occurrence

Required within <resource> or <member-resource>.

```
<resources> (0 or more)
  <resource> (1 or more)
    <name> (1)
    <display-name> (1)
    <description> {0 or 1}
    <type-name-ref> (1)
```

Example

For an example, see <resource>.

<uniquename>

This element, for internal use, takes a string value to specify a unique name to reference the item. (The `JpsPrincipal` class can use a GUID and unique name, both computed by the underlying policy provisioning APIs, to uniquely identify a principal.) Depending on the parent element, the item could be an application role, application role member (not an enterprise group member), or principal. It is typically used with an LDAP provider to uniquely identify the item (an application role member, for example). A unique name is sometimes generated and used internally by Oracle Platform Security.

The unique name for an application role would be: "appid=application_name, name=actual_rolename". For example:

```
<principal>
  <class>
    oracle.security.jps.service.policystore.adminroles.AdminRolePrincipal
  </class>
  <uniquename>
    APPID=App1,name="FARM=D.1.2.3,APPLICATION=PolicyServlet,TYPE=OPERATOR"
  </uniquename>
</principal>
```

Parent Element

<app-role>, <member>, or <principal>

Child Elements

None

Occurrence

Optional, zero or one

```
<policy-store> {0 or 1}
  <applications> {0 or 1}
    <application> {1 or more}
      <name> {1}
      <description> {0 or 1}
      <app-roles> {0 or 1}
        <app-role> {1 or more}
          <name> {1}
          <class> {1}
          <display-name> {0 or 1}
          <description> {0 or 1}
          <guid> {0 or 1}
          <uniquename> {0 or 1}
          <extended-attributes> {0 or 1}
          ...
        <members> {0 or 1}
          <member> {1 or more}
            <name> {1}
            <class> {1}
            <uniquename> {0 or 1}
            <guid> {0 or 1}

<jazn-policy> {0 or 1}
```

```
<grant> {0 or more}
  <description> {0 or 1}
  <grantee> {0 or 1}
    <principals> {0 or 1}
      <principal> {0 or more}
        <name> {1}
        <class> {1}
        <uniqueusername> {0 or 1}
        <guid> {0 or 1}
      <codesource> {0 or 1}
        <url> {1}
    <permissions> {0 or 1}
      <permission> {1 or more}
        <class> {1}
        <name> {0 or 1}
        <actions> {0 or 1}
```

<url>

This element specifies the URL of the code that is granted permissions.

Note the following points:

- URL values cannot be restricted to a single class.
- URL values with ".jar" suffix match the JAR files in the specified directory.
- URL values with "/" suffix match all class files (not JAR files) in the specified directory.
- URL values with "/*" suffix match all files (both class and JAR files) in the specified directory.
- URL values with "/-" suffix match all files (both class and JAR files) in the specified directory and, recursively, all files in subdirectories.
- The system variables `oracle.deployed.app.dir` and `oracle.deployed.app.ext` can be used to specify a URL independent of the platform.

Parent Element

`<codesource>`

Child Elements

None

Occurrence

Required within parent element, one only

```
<jazn-policy> {0 or 1}
  <grant> {0 or more}
    <description> {0 or 1}
    <grantee> {0 or 1}
      <principals> {0 or 1}
        <principal> {0 or more}
          <name> {1}
          <class> {1}
          <uniqueusername> {0 or 1}
          <guid> {0 or 1}
        <codesource> {0 or 1}
          <url> {1}
      <permissions> {0 or 1}
        <permission> {1 or more}
          <class> {1}
          <name> {0 or 1}
          <actions> {0 or 1}
```

Example

The following example illustrates the use of the system variables `oracle.deployed.app.dir` and `oracle.deployed.app.ext` to specify URLs independent of the server platform.

Suppose an application grant requires a codesource URL that differs with the server platform:

```
On WebLogic
<grant>
  <grantee>
    <codesource>
      <url>file:${domain.home}/servers/${weblogic.Name}/tmp/_WL_user/myApp/-</url>
    </codesource>
  </grantee>
  <permissions> ... </permissions>
</grant>
```

```
On WebSphere
<grant>
  <grantee>
    <codesource>
      <url>file:${user.install.root}/installedApps/${was.cell.name}/myApp/-</url>
    </codesource>
  </grantee>
  <permissions> ... </permissions>
</grant>
```

Then, using the following system variable settings:

```
On WebLogic
-Doracle.deployed.app.dir=${DOMAIN_HOME}/servers/${SERVER_NAME}/tmp/_WL_user
-Doracle.deployed.app.ext=-
```

```
On WebSphere
-Doracle.deployed.app.dir=${USER_INSTALL_ROOT}/installedApps/${CELL}
-Doracle.deployed.app.ext=.ear/-
```

the following specification would work for *both* platforms, WebLogic and WebSphere:

```
<grant>
  <grantee>
    <codesource>
      <url>file:${oracle.deployed.app.dir}/<MyApp>${oracle.deployed.app.ext}</url>
    </codesource>
  </grantee>
  <permissions> ... </permissions>
</grant>
```

<user>

<user>

This element specifies a user within a realm.

Attributes

Name	Description
deactivated	Specifies whether the user is valid or not. Set this attribute to <code>true</code> if you want to maintain a user in the configuration file but not have it be a currently valid user. This is the initial configuration of the anonymous user in the <code>jazn.com</code> realm, for example. Values: <code>true</code> or <code>false</code> Default: <code>false</code>

Parent Element

<users>

Child Elements

<name>, <display-name>, <description>, <guid>, <credentials>

Occurrence

Optional, zero or more

```
<jazn-realm> {0 or 1}
  <realm> {0 or more}
    <name> {1}
    <users> {0 or 1}
      <user> {0 or more}
        <name> {1}
        <display-name> {0 or 1}
        <description> {0 or 1}
        <guid> {0 or 1}
        <credentials> {0 or 1}
      <roles> {0 or 1}
    ...
```

Example

See <jazn-realm> for examples.

<users>

This element specifies the set of users belonging to a realm.

Parent Element

[<realm>](#)

Child Elements

[<user>](#)

Occurrence

Optional, zero or one

```
<jazn-realm> {0 or 1}
  <realm> {0 or more}
    <name> {1}
    <users> {0 or 1}
      <user> {0 or more}
        <name> {1}
        <display-name> {0 or 1}
        <description> {0 or 1}
        <guid> {0 or 1}
        <credentials> {0 or 1}
      <roles> {0 or 1}
    ...
```

Example

See [<jazn-realm>](#) for an example.

<value>

This element specifies a value for an attribute. You can specify additional attributes for application-level roles using the [<extended-attributes>](#) element.

Parent Element

[<attribute>](#)

Child Elements

None

Occurrence

Required within the parent element, one only

```
<policy-store> {0 or 1}
  <applications> {0 or 1}
    <application> {1 or more}
      <name> {1}
      <description> {0 or 1}
      <app-roles> {0 or 1}
        <app-role> {1 or more}
          <name> {1}
          <class> {1}
          <display-name> {0 or 1}
          <description> {0 or 1}
          <guid> {0 or 1}
          <uniqueusername> {0 or 1}
          <extended-attributes> {0 or 1}
            <attribute> {1 or more}
              <name> {1}
              <values> {1}
                <value> {1 or more}
          <members> {0 or 1}
            <member> {1 or more}
              <name> {1}
              <class> {1}
              <uniqueusername> {0 or 1}
              <guid> {0 or 1}
```

Example

```
<app-roles>
  <app-role>
    <name>Knight</name>
    <display-name>Fellowship of the Ring</display-name>
    <class>oracle.security.jps.service.policystore.ApplicationRole</class>
    <extended-attributes>
      <attribute>
        <name>SCOPE</name>
        <values>
          <value>Part-I</value>
        </values>
      </attribute>
    </extended-attributes>
  </app-role>
```


<values>

This element specifies a set of values, each of which specify the value of an attribute. An attribute can have more than one value.

Parent Element

<attribute>

Child Elements

<value>

Occurrence

Required within the parent element, one only

```
<policy-store> {0 or 1}
  <applications> {0 or 1}
    <application> {1 or more}
      <name> {1}
      <description> {0 or 1}
      <app-roles> {0 or 1}
        <app-role> {1 or more}
          <name> {1}
          <class> {1}
          <display-name> {0 or 1}
          <description> {0 or 1}
          <guid> {0 or 1}
          <unique-name> {0 or 1}
          <extended-attributes> {0 or 1}
            <attribute> {1 or more}
              <name> {1}
              <values> {1}
                <value> {1 or more}
            </attribute>
          </extended-attributes>
          <members> {0 or 1}
            <member> {1 or more}
              <name> {1}
              <class> {1}
              <unique-name> {0 or 1}
              <guid> {0 or 1}
            </member>
          </members>
        </app-roles>
      </application>
    </applications>
  </policy-store>
```

Example

```
<app-roles>
  <app-role>
    <name>Knight</name>
    <display-name>Fellowship of the Ring</display-name>
    <class>oracle.security.jps.service.policystore.ApplicationRole</class>
    <extended-attributes>
      <attribute>
        <name>SCOPE</name>
        <values>
          <value>Part-I</value>
        </values>
      </attribute>
    </extended-attributes>
  </app-role>
```


<values>

Oracle Fusion Middleware Audit Framework Reference

This appendix provides reference information for the Oracle Fusion Middleware Audit Framework. It contains these topics:

- [Audit Events](#)
- [Pre-built Audit Reports](#)
- [The Audit Schema](#)
- [WLST Commands for Auditing](#)
- [Audit Filter Expression Syntax](#)
- [Naming and Logging Format of Audit Files](#)

C.1 Audit Events

This section describes the components that are audited and the types of events that can be audited.

C.1.1 What Components Can be Audited?

In 11g Release 1 (11.1.1), specific Java components and system components can generate audit records; they are known as audit-aware components.

Java Components that can be Audited

The following components can be audited with Fusion Middleware Audit Framework:

- Directory Integration Platform Server
- Oracle Platform Security Services
- Oracle Web Services Manager
 - Agent
 - Policy Manager
 - Policy Attachment
- Oracle Web Services
- Oracle Identity Federation
- Reports Server

System Components that can be Audited

The following components can be audited with Fusion Middleware Audit Framework:

- Oracle HTTP Server
- Oracle Web Cache
- Oracle Internet Directory
- Oracle Virtual Directory

C.1.2 What Events can be Audited?

The set of tables in this section shows, for each audit-aware system components and subcomponent, what event types can be audited:

- [Oracle Directory Integration Platform Events and their Attributes](#)
- [Oracle Platform Security Services Events and their Attributes](#)
- [Oracle HTTP Server Events and their Attributes](#)
- [Oracle Internet Directory Events and their Attributes](#)
- [Oracle Identity Federation Events and their Attributes](#)
- [Oracle Virtual Directory Events and their Attributes](#)
- [OWSM-Agent Events and their Attributes](#)
- [OWSM-PM-EJB Events and their Attributes](#)
- [Reports Server Events and their Attributes](#)
- [WS-Policy Attachment Events and their Attributes](#)
- [Oracle Web Cache Events and their Attributes](#)
- [Oracle Web Services Manager Events and their Attributes](#)

C.1.2.1 Oracle Directory Integration Platform Events and their Attributes

Table C-1 Oracle Directory Integration Platform Events

Event Category	Event Type	Attributes used by Event
ServiceUtilize	InvokeService	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles
	TerminateService	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles
SynchronizationEvents		

Table C-1 (Cont.) Oracle Directory Integration Platform Events

Event Category	Event Type	Attributes used by Event
	Add	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, AssociateProfileName, ProfileName, EntryDN
	Modify	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, AssociateProfileName, ProfileName, EntryDN
	Delete	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, AssociateProfileName, ProfileName, EntryDN
ProvisioningEvents	UserAdd	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, ProfileName, ProvEvent
	UserModify	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, ProfileName, ProvEvent
	UserDelete	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, ProfileName, ProvEvent
	GroupAdd	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, ProfileName, ProvEvent
	GroupModify	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, ProfileName, ProvEvent
	GroupDelete	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, ProfileName, ProvEvent

Table C-1 (Cont.) Oracle Directory Integration Platform Events

Event Category	Event Type	Attributes used by Event
	IdentityAdd	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, ProfileName, ProvEvent
	IdentityModify	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, ProfileName, ProvEvent
	IdentityDelete	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, ProfileName, ProvEvent
	SubscriptionAdd	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, ProfileName, ProvEvent
	SubscriptionModify	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, ProfileName, ProvEvent
	SubscriptionDelete	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, ProfileName, ProvEvent
ProfileManagementEvents	DeleteProvProfiles	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode
	UpdateProvProfile	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode
	ActivateProvProfile	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode

Table C-1 (Cont.) Oracle Directory Integration Platform Events

Event Category	Event Type	Attributes used by Event
	DeactivateProvProfile	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode
	CreateSyncProfile	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode
	DeleteSyncProfile	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode
	UpdateSyncProfile	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode
	ActivateSyncProfile	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode
	DeactivateSyncProfile	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode
	SyncProfileUpdateChgNum	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode
	ExpressSyncSetup	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode
	SyncProfileBootstrap	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode

Table C-1 (Cont.) Oracle Directory Integration Platform Events

Event Category	Event Type	Attributes used by Event
	SyncProfileExtAuthPlugins	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode
	ProvProfileBulkProv	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode
SchedulerEvents	AddJob	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles, JobName, JobType
	RemoveJob	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles, JobName, JobType

C.1.2.2 Oracle Platform Security Services Events and their Attributes

Table C-2 Oracle Platform Security Services Events

Event Category	Event Type	Attributes used by Event
Authorization	CheckPermission	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles, CodeSource, Principals, InitiatorGUID, Subject, PermissionAction, PermissionTarget, PermissionClass
	CheckSubject	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles, CodeSource, Principals, InitiatorGUID, Subject

Table C-2 (Cont.) Oracle Platform Security Services Events

Event Category	Event Type	Attributes used by Event
CredentialManagement	CreateCredential	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles, mapName, key, CodeSource, Principals, InitiatorGUID
	DeleteCredential	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles, mapName, key, CodeSource, Principals, InitiatorGUID
	AccessCredential	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles, mapName, key, CodeSource, Principals, InitiatorGUID
	ModifyCredential	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles, mapName, key, CodeSource, Principals, InitiatorGUID
PolicyManagement	PolicyGrant	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles, CodeSource, Principals, InitiatorGUID, PermissionAction, PermissionTarget, PermissionClass, PermissionScope
	PolicyRevoke	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles, CodeSource, Principals, InitiatorGUID, PermissionAction, PermissionTarget, PermissionClass, PermissionScope

Table C-2 (Cont.) Oracle Platform Security Services Events

Event Category	Event Type	Attributes used by Event
RoleManagement	RoleMembershipAdd	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles, CodeSource, Principals, InitiatorGUID, ApplicationRole, EnterpriseRoles, PermissionScope
	RoleMembershipRemove	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles, CodeSource, Principals, InitiatorGUID, ApplicationRole, EnterpriseRoles, PermissionScope

C.1.2.3 Oracle HTTP Server Events and their Attributes

Table C-3 Oracle HTTP Server Events

Event Category	Event Type	Attributes used by Event
UserSession	UserLogin	ComponentType, InstanceId, HostId, HostNwaddr, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Resource, AuthenticationMethod, Reason
	UserLogout	ComponentType, InstanceId, HostId, HostNwaddr, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Resource, AuthenticationMethod, Reason

Table C-3 (Cont.) Oracle HTTP Server Events

Event Category	Event Type	Attributes used by Event
	Authentication	ComponentType, InstanceId, HostId, HostNwaddr, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Resource, AuthenticationMethod, Reason, SSLConnection
Authorization	CheckAuthorization	ComponentType, InstanceId, HostId, HostNwaddr, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Resource, Reason, AuthorizationType

C.1.2.4 Oracle Internet Directory Events and their Attributes

Table C-4 Oracle Directory Integration Platform Events

Event Category	Event Type	Attributes used by Event
UserSession	UserLogin	ComponentType, InstanceId, HostId, HostNwaddr, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Roles, custEventStatusDetail, custEventOp, AuthenticationMethod
	UserLogout	ComponentType, InstanceId, HostId, HostNwaddr, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Roles, custEventStatusDetail, custEventOp
Authorization	CheckAuthorization	ComponentType, InstanceId, HostId, HostNwaddr, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, custEventStatusDetail, custEventOp
DataAccess	ModifyDataItemAttributes	ComponentType, InstanceId, HostId, HostNwaddr, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Resource, custEventStatusDetail, custEventOp

Table C-4 (Cont.) Oracle Directory Integration Platform Events

Event Category	Event Type	Attributes used by Event
	CompareDataItemAttributes	ComponentType, InstanceId, HostId, HostNwaddr, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Resource, custEventStatusDetail, custEventOp
AccountManagement	ChangePassword	ComponentType, InstanceId, HostId, HostNwaddr, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, custEventStatusDetail, custEventOp
	CreateAccount	ComponentType, InstanceId, HostId, HostNwaddr, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, custEventStatusDetail, custEventOp
	DeleteAccount	ComponentType, InstanceId, HostId, HostNwaddr, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, custEventStatusDetail, custEventOp
	DisableAccount	ComponentType, InstanceId, HostId, HostNwaddr, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, custEventStatusDetail, custEventOp
	EnableAccount	ComponentType, InstanceId, HostId, HostNwaddr, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, custEventStatusDetail, custEventOp
	ModifyAccount	ComponentType, InstanceId, HostId, HostNwaddr, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, custEventStatusDetail, custEventOp

Table C-4 (Cont.) Oracle Directory Integration Platform Events

Event Category	Event Type	Attributes used by Event
	LockAccount	ComponentType, InstanceId, HostId, HostNwaddr, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, custEventStatusDetail, custEventOp
LDAPEntryAccess	custInternalOperation	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles, custEventStatusDetail, custEventOp

C.1.2.5 Oracle Identity Federation Events and their Attributes

Table C-5 Oracle Identity Federation Events

Event Category	Event Type	Attributes used by Event
UserSession	LocalAuthentication	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles, SessionID, AuthenticationMethod, UserID, AuthenticationMechanism, AuthenticationEngineID
	LocalLogout	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles, SessionID, AuthenticationMethod, UserID
	CreateUserSession	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles, SessionID, AuthenticationMethod, UserID, AuthenticationMechanism

Table C-5 (Cont.) Oracle Identity Federation Events

Event Category	Event Type	Attributes used by Event
	DeleteUserSession	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles, SessionID, AuthenticationMethod, UserID
	CreateUserFederation	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles, RemoteProviderID, ProtocolVersion, NameIDQualifier, NameIDValue, NameIDFormat, FederationID, UserID, FederationType
	DeleteUserFederation	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles, RemoteProviderID, ProtocolVersion, NameIDQualifier, NameIDValue, NameIDFormat, FederationID, UserID, FederationType
	CreateActiveUserFederation	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles, RemoteProviderID, ProtocolVersion, SessionID, FederationID, AuthenticationMethod, UserID, FederationType
	DeleteActiveUserFederation	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles, RemoteProviderID, ProtocolVersion, SessionID, FederationID, AuthenticationMethod, UserID, FederationType

Table C-5 (Cont.) Oracle Identity Federation Events

Event Category	Event Type	Attributes used by Event
	UpdateUserFederation	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles, RemoteProviderID, ProtocolVersion, NameIDQualifier, NameIDValue, NameIDFormat, FederationID, UserID, FederationType, OldNameIDQualifier, OldNameIDValue
ProtocolFlow	IncomingMessage	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles, RemoteProviderID, ProtocolVersion, Binding, Role, UserID, MessageType, IncomingMessageString, IncomingMessageStringCLOB
	OutgoingMessage	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles, RemoteProviderID, ProtocolVersion, Binding, Role, UserID, MessageType, OutgoingMessageString, OutgoingMessageStringCLOB
	AssertionCreation	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles, RemoteProviderID, ProtocolVersion, NameIDQualifier, NameIDValue, NameIDFormat, SessionID, FederationID, UserID, AssertionVersion, IssueInstant, Issuer, AssertionID
	AssertionConsumption	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles, RemoteProviderID, ProtocolVersion, NameIDQualifier, NameIDValue, NameIDFormat, SessionID, FederationID, UserID, AssertionVersion, IssueInstant, Issuer, AssertionID

Table C-5 (Cont.) Oracle Identity Federation Events

Event Category	Event Type	Attributes used by Event
Security	CreateSignature	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles, RemoteProviderID, ProtocolVersion, NameIDQualifier, NameIDValue, NameIDFormat, SessionID, FederationID, Type
	VerifySignature	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles, RemoteProviderID, ProtocolVersion, NameIDQualifier, NameIDValue, NameIDFormat, SessionID, FederationID, Type
	EncryptData	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles, RemoteProviderID, ProtocolVersion, NameIDQualifier, NameIDValue, NameIDFormat, SessionID, FederationID, Type
	DecryptData	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles, RemoteProviderID, ProtocolVersion, NameIDQualifier, NameIDValue, NameIDFormat, SessionID, FederationID, Type
ServerConfiguration	ChangeCOT	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles, RemoteProviderID, NameIDQualifier, NameIDValue, NameIDFormat, SessionID, FederationID, COTBefore, COTAfter

Table C-5 (Cont.) Oracle Identity Federation Events

Event Category	Event Type	Attributes used by Event
	ChangeServerProperty	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles, RemoteProviderID, NameIDQualifier, NameIDValue, NameIDFormat, SessionID, FederationID, ServerConfigBefore, ServerConfigAfter
	ChangeDataStore	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles, RemoteProviderID, NameIDQualifier, NameIDValue, NameIDFormat, SessionID, FederationID, DataStoreBefore, DataStoreAfter
	CreateConfigProperty	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles, RemoteProviderID, NameIDQualifier, NameIDValue, NameIDFormat, SessionID, FederationID, PropertyName, PropertyType, PeerProviderID, PropertyContext, NewValue
	ChangeConfigProperty	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles, RemoteProviderID, NameIDQualifier, NameIDValue, NameIDFormat, SessionID, FederationID, PropertyName, PropertyType, PeerProviderID, PropertyContext, OldValue, NewValue
	DeleteConfigProperty	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles, RemoteProviderID, ProtocolVersion, NameIDQualifier, NameIDValue, NameIDFormat, SessionID, FederationID, PropertyName, PropertyType, PeerProviderID, PropertyContext, Description, OldValue

Table C-5 (Cont.) Oracle Identity Federation Events

Event Category	Event Type	Attributes used by Event
	CreatePeerProvider	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles, RemoteProviderID, ProtocolVersion, NameIDQualifier, NameIDValue, NameIDFormat, SessionID, FederationID, PeerProviderID, Description, ProviderType
	UpdatePeerProvider	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles, RemoteProviderID, ProtocolVersion, NameIDQualifier, NameIDValue, NameIDFormat, SessionID, FederationID, PeerProviderID, Description, ProviderType
	DeletePeerProvider	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles, RemoteProviderID, ProtocolVersion, NameIDQualifier, NameIDValue, NameIDFormat, SessionID, FederationID, PeerProviderID, Description, ProviderType
	LoadMetadata	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles, RemoteProviderID, NameIDQualifier, NameIDValue, NameIDFormat, SessionID, FederationID, Description, Metadata
	SetDataStoreType	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles, RemoteProviderID, NameIDQualifier, NameIDValue, NameIDFormat, SessionID, FederationID, OldValue, NewDataStoreType, DataStoreName

C.1.2.6 Oracle Virtual Directory Events and their Attributes

Table C-6 Oracle Virtual Directory Events

Event Category	Event Type	Attributes used by Event
UserSession	UserLogin	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles, AuthenticationMethod
	UserLogout	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles
Authorization	CheckAuthorization	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles
DataAccess	QueryDataItemAttributes	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles
	ModifyDataItemAttributes	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles
	CompareDataItemAttributes	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles
ServiceManagement	RemoveService	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles, ServiceOperation

Table C-6 (Cont.) Oracle Virtual Directory Events

Event Category	Event Type	Attributes used by Event
	ModifyServiceConfig	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles, ServiceOperation
	AddService	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles, ServiceOperation
LDAPEntryAccess	Add	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles
	Delete	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles
	Modify	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles
	Rename	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles
	Compare	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles

C.1.2.7 OWSM-Agent Events and their Attributes

Table C-7 OWSM-Agent Events

Event Category	Event Type	Attributes used by Event
UserSession	Authentication	ComponentType, InstanceId, HostId, HostNwaddr, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Resource, AssertionName, CompositeName, Endpoint, AgentMode, ModelObjectName, Operation, ProcessingStage, Version, Protocol
Authorization	CheckAuthorization	ComponentType, InstanceId, HostId, HostNwaddr, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Resource, AssertionName, CompositeName, Endpoint, AgentMode, ModelObjectName, Operation, ProcessingStage, Version, Protocol
PolicyEnforcement	EnforceConfidentiality	ComponentType, InstanceId, HostId, HostNwaddr, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Resource, AssertionName, CompositeName, Endpoint, AgentMode, ModelObjectName, Operation, ProcessingStage, Version, Protocol
	EnforceIntegrity	ComponentType, InstanceId, HostId, HostNwaddr, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Resource, AssertionName, CompositeName, Endpoint, AgentMode, ModelObjectName, Operation, ProcessingStage, Version, Protocol
	EnforcePolicy	ComponentType, InstanceId, HostId, HostNwaddr, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Resource, AssertionName, CompositeName, Endpoint, AgentMode, ModelObjectName, Operation, ProcessingStage, Version, Protocol

C.1.2.8 OWSM-PM-EJB Events and their Attributes

Table C-8 OWSM-PM-EJB Events

Event Category	Event Type	Attributes used by Event
AssertionTemplateAuthori ng	CreateAssertionTemplate	ComponentType, InstanceId, HostId, HostNwaddr, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, Resource, Version
	DeleteAssertionTemplate	ComponentType, InstanceId, HostId, HostNwaddr, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, Resource, Version, ToVersion
	ModifyAssertionTemplat e	ComponentType, InstanceId, HostId, HostNwaddr, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, Resource, Version
PolicyAuthoring	CreatePolicy	ComponentType, InstanceId, HostId, HostNwaddr, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, Resource, Version
	DeletePolicy	ComponentType, InstanceId, HostId, HostNwaddr, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, Resource, Version, ToVersion,
	ModifyPolicy	ComponentType, InstanceId, HostId, HostNwaddr, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, Resource, Version

C.1.2.9 Reports Server Events and their Attributes

Table C–9 Reports Server Events

Event Category	Event Type	Attributes used by Event
UserSession	UserLogin	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles
Authorization	CheckAuthorization	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles

C.1.2.10 WS-Policy Attachment Events and their Attributes

Table C–10 WS-Policy Attachment Events

Event Category	Event Type	Attributes used by Event
PolicyAttachment	PolicyAttachmentEvent	ComponentType, InstanceId, HostId, HostNwaddr, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, PolicyChangeType, PolicyURI, PolicyCategory, PolicyStatus, ServiceEndPoint, PolicySubjRescPattern

C.1.2.11 Oracle Web Cache Events and their Attributes

Table C-11 Oracle Web Cache Events

Event Category	Event Type	Attributes used by Event
UserSession	UserLogin	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles, AuthenticationMethod
	UserLogout	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles, AuthenticationMethod
Authorization	CheckAuthorization	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles
DataAccess	FilterRequest	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles
ServiceManagement	ModifyServiceConfig	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles
	ConfigServicePermissions	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles

Table C-11 (Cont.) Oracle Web Cache Events

Event Category	Event Type	Attributes used by Event
ServiceUtilize	InvokeService	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles
	TerminateService	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles
PeerAssocManagement	CreatePeerAssoc	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles
	TerminatePeerAssoc	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles
	ChallengePeerAssoc	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles

Table C–11 (Cont.) Oracle Web Cache Events

Event Category	Event Type	Attributes used by Event
Authentication	ClientAuthentication	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles
	ServerAuthentication	ComponentType, InstanceId, HostId, HostNwaddr, ModuleId, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Roles

C.1.2.12 Oracle Web Services Manager Events and their Attributes

Table C–12 Oracle Web Services Manager Events

Event Category	Event Type	Attributes used by Event
WS-Processing	RequestReceived	ComponentType, InstanceId, HostId, HostNwaddr, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Protocol, Endpoint, Operation, FaultUrl
	ResponseSent	ComponentType, InstanceId, HostId, HostNwaddr, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, Protocol, Endpoint, Operation, FaultUri
WS-Fault	SoapFaultEvent	ComponentType, InstanceId, HostId, HostNwaddr, ProcessId, OracleHome, HomeInstance, ECID, RID, ContextFields, SessionId, TargetComponentType, ApplicationName, EventType, EventCategory, EventStatus, TstzOriginating, ThreadId, ComponentName, Initiator, MessageText, FailureCode, RemoteIP, Target, Resource, URI, Source, Protocol, Endpoint, Operation

C.1.3 Event Attribute Descriptions

lists all attributes for all audited events. Use this table to learn about the attributes used in the event of interest.

Table C-13 Attributes of Audited Events

Attribute Name	Description
AgentMode	Mode in which agent performed policy enforcement.
ApplicationName	The Java EE application name
ApplicationRole	This attribute used for application roles audit for role membership management
AssertionID	The value of the "AssertionID" attribute of the assertion
AssertionName	Name of the assertion that failed enforcement.
AssertionVersion	The version number of the assertion corresponding to this event (ex. 2.0)
AssociateProfileName	This attribute is used to audit the Associate Profile Name
AuthenticationEngineID	The identifier of the authentication engine used during local authentication
AuthenticationMechanism	The authentication mechanism used during local authentication
AuthenticationMethod	The Authentication method - password / SSL / Kerberos and so on.
AuthorizationType	Access/authorization configuration directive: Regular = 'Require' directive, SSL = 'SSLRequire' directive
Binding	The binding used to send the message (SOAP, POST, GET, Artifact,...)
COTAfter	The contents of the federations configuration file after the change
COTBefore	The contents of the federations configuration file before the change
CodeSource	This attribute used for code source audit for rolemembershipmanagement
ComponentName	ComponentName
ComponentType	Type of the component.
CompositeName	Name of the composite (apply to SOA application only) against which the policy is being enforced.
ContextFields	This attribute contains the context fields extracted from dms context.
custEventOp	This attribute specifies the LDAP operation name associated with this event, e.g. ldapbind, ldapadd, ldapsearch and so on.
custEventStatusDetail	This attribute conveys event status detail info, e.g. error code and other details in case of failure of the associated LDAP operation.
DataStoreAfter	The data stores configuration after the change
DataStoreBefore	The data stores configuration before the change
DataStoreName	The name of the data store being modified (examples: user data store, federation datastore)
Description	Description of the trusted provider
ECID	Identifies the thread of execution that the originating component participates in.
Endpoint	The URI which identifies the endpoint for which the event was triggered. For example, an HTTP require will record the URL.

Table C-13 (Cont.) Attributes of Audited Events

Attribute Name	Description
EnterpriseRoles	This attribute used for enterprise roles audit for rolemembershipmanagement
EntryDN	This attribute is used to audit the entry Distinguished Name
EventCategory	The category of the audit event.
EventStatus	The outcome of the audit event - success or failure
EventType	The type of the audit event. Use wlst listAuditEvents to list out all the events.
FailureCode	The error code in case EventStatus = failure
FaultUri	If processing yielded a fault, the URI of the fault that will be sent.
FederationID	The ID of the federation
FederationType	The type of the federation that is being created or deleted (SP/IdP)
HomeInstance	The ORACLE_INSTANCE directory of the component
HostId	DNS hostname of originating host
HostNwaddr	IP or other network address of originating host
IncomingMessageString	null
IncomingMessageStringCLOB	null
Initiator	Identifies the UID of the user who is doing the operation
InitiatorGUID	This attribute used for initiator guid audit for authorization
InstanceId	Name of the Oracle Instance to which this component belongs.
IssueInstant	The value of the "IssueInstant" attribute of the assertion
Issuer	The value of the "Issuer" attribute of the assertion
JobName	This attribute is used to audit the Scheduler Job Name
JobType	This attribute is used to audit the Scheduler Job Name
key	This is the credential key for the Credential Store
mapName	This is the map name (alias name) for the Credential Store
MessageText	Description of the audit event
MessageType	The type of the message (ex. SSOLoginRequest/SSOLoginResponse/SSOLogoutRequest/...)
Metadata	The provider metadata loaded
ModelObjectName	Name of the Web service or client name against which the policy is being enforced.
ModuleId	ID of the module that originated the message. Interpretation is specific to the Component ID.
NameIDFormat	The format of the NameID of the subject
NameIDQualifier	The qualifier of the nameID of the subject
NameIDValue	The value of the nameID of the subject
NewDataStoreType	The new type of the data store

Table C-13 (Cont.) Attributes of Audited Events

Attribute Name	Description
NewValue	The value of the property after the configuration change
OldNameIDQualifier	The nameID qualifier before the update took place
OldNameIDValue	The nameID value before the update took place
OldValue	The value of the property before the configuration change
Operation	For SOAP requests, the operation for which the event was triggered.
OracleHome	The ORACLE_HOME directory of the component
OutgoingMessageString	null
OutgoingMessageStringLOB	null
PeerProviderID	The ID of the trusted provider associated with the modified property (If the modified property does not correspond to a trusted provider, this attribute is empty.)
PermissionAction	This attribute used for permission action audit for authorization
PermissionClass	This attribute used for permission class audit for policy store
PermissionScope	This attribute used for permission scope audit for role membership management
PermissionTarget	This attribute used for permission target audit for policy store
PolicyCategory	The category of the policy for which the event was triggered.(comma-separated list)
PolicyChangeType	The type of change that occurred.
PolicyStatus	The status of the policy for which the event was triggered.(comma-separated list)
PolicySubjRescPattern	The policy subject resource pattern which identifies the policy subject for which the event was triggered.
PolicyURI	The URI which identifies the policy for which the event was triggered.(comma-separated list)
Principals	This attribute used for principals audit for role membership management
ProcessId	ID of the process that originated the message
ProcessingStage	Processing stage during which the policy enforcement occurred.
ProfileName	This attribute is used to audit the Sync Profile Name
PropertyContext	The location of the property in the configuration
PropertyName	The name of the configuration property
PropertyType	The type of the property (examples: PropertiesList, PropertiesMap, String, Boolean)
Protocol	The protocol of the request.
ProtocolVersion	The version of the protocol being used (examples: SAML2.0, Libv11)
ProvEvent	This attribute is used to audit the Prov Event
ProviderType	The type of the provider (examples: sp, idp, sp idp)

Table C-13 (Cont.) Attributes of Audited Events

Attribute Name	Description
RID	This is the relationship identifier, it is used to provide the full and correct calling relationships between threads and processes.
Reason	The reason this event occurred
RemoteIP	IP address of the client initiating this event
RemoteProviderID	The provider ID of the remote server
Resource	Identifies a resource that is being accessed. A resource can be many things - web page, file, directory share, web service, XML document, a portlet. The resource can be named as a combination of a host name, and an URI.
Role	The role of Oracle Identity Federation during the protocol step performed (for example Service Provider/ Identity Provider/Attribute Authority/..)
Roles	The roles that the user was granted at the time of login.
SSLConnection	Was SSL connection used by client to transmit request?
ServerConfigAfter	The server configuration after the change
ServerConfigBefore	The server configuration before the change
ServiceEndPoint	The URI which identifies the service for which the event was triggered.
ServiceOperation	Name of the operation performed that changes the service configuration
SessionID	The ID of the current session
SessionId	ID of the login session.
Source	The source of the fault.
Subject	This attribute used for subject audit for authorization
Target	Identifies the UID of the user on whom the operation is being done. E.g. is Alice changes Bob's password, then Alice is the initiator and Bob is the target
TargetComponentType	This is the target component type.
ThreadId	ID of the thread that generated this event
ToVersion	Upper end when deleting a range of policy versions.
TstzOriginating	Date and time when the audit event was generated
Type	The type of cryptographic data being processed (XML, String)
URI	The URI of the fault.
UserID	The identifier of the user in this protocol step
Version	Version of policy that was modified.

C.2 Pre-built Audit Reports

Oracle Fusion Middleware Audit Framework provides a range of out-of-the-box reports that are accessible through Oracle Business Intelligence Publisher. The reports are grouped according to the type of audit data they contain:

- [Common Audit Reports](#)
- [Component-Specific Audit Reports](#)

C.2.1 Common Audit Reports

A list of common reports appears in [Section 14.5, "Audit Report Details"](#).

C.2.2 Component-Specific Audit Reports

Component-Specific reports are organized as follows:

- Oracle Fusion Middleware Audit Framework
 - Configuration Changes
- Oracle HTTP Server
 - Errors and Exceptions
 - User Activities
 - All Events
- Oracle Internet Directory
 - Account Management
 - * Account Profile History
 - * Accounts Deleted
 - * Accounts Enabled
 - * Password Changes
 - * Accounts Created
 - * Accounts Disabled
 - * Accounts Locked Out
 - User Activities
 - * Authentication History
 - * Authorization History
 - Errors and Exceptions
 - * All Errors and Exceptions
 - * Authentication Failures
 - * Authorization Failures
 - All Events
- Oracle Virtual Directory
 - User Activities
 - * Authentication History
 - * Authorization History
 - Errors and Exceptions
 - * All Errors and Exceptions
 - * Authentication Failures
 - * Authorization Failures
 - All Events

- Reports Server
 - User Activities
 - * Authentication History
 - * Authorization History
 - Errors and Exceptions
 - * All Errors and Exceptions
 - * Authentication Failures
 - * Authorization Failures
 - All Events
- Oracle Directory Integration Platform
 - All Errors and Exceptions
 - Profile Management Events
 - All Events
- Oracle Identity Federation
 - Errors and Exceptions
 - * All Errors and Exceptions
 - * Authentication Failures
 - All Events
 - Federation user Activity
 - Authentication History
 - Assertion Activity
- Oracle Platform Security Services
 - Errors and Exceptions
 - * All Errors and Exceptions
 - * Authentication Failures
 - All Events
 - Application Role Management
 - Credential Management
 - Authorization History
 - Application Policy Management
 - Credential Access
 - System Policy Management
- Oracle Web Services Manager
 - User Activities
 - * Authentication History
 - * Authorization History
 - Errors and Exceptions

- * All Errors and Exceptions
- * Authentication Failures
- * Authorization Failures
- All Events
- Policy Management
 - * Assertion Template Management
 - * Web Services Policy Management
- Policy Enforcements
 - * Confidentiality Enforcements
 - * Policy Enforcements
 - * Message Integrity Enforcements
 - * Violations
- Request Response
- Policy Attachments
- Oracle Web Cache
 - User Activities
 - * Authentication History
 - * Authorization History
 - Errors and Exceptions
 - * All Errors and Exceptions
 - * Authentication Failures
 - * Authorization Failures
 - All Events

C.3 The Audit Schema

If you have additional audit reporting requirements beyond the pre-built reports described in [Section C.2, "Pre-built Audit Reports"](#), you can create custom reports using your choice of reporting tools. For example, while the pre-built reports use a subset of the event attributes, you can make use of the entire audit attribute set for an event in creating custom reports.

[Table C–14](#) and [Table C–15](#) describe the audit schema, which is useful when building custom reports.

Table C–14 The Audit Schema

Table Name	Column Name	Data Type	Nullable	Column ID
BASE TABLE	IAU_ID	NUMBER	Yes	1
	IAU_ORGID	VARCHAR2(255 Bytes)	Yes	2
	IAU_COMPONENTID	VARCHAR2(255 Bytes)	Yes	3

Table C-14 (Cont.) The Audit Schema

Table Name	Column Name	Data Type	Nullable	Column ID
	IAU_COMPONENTTYPE	VARCHAR2(255 Bytes)	Yes	4
	IAU_INSTANCEID	VARCHAR2(255 Bytes)	Yes	5
	IAU_HOSTINGCLIENTID	VARCHAR2(255 Bytes)	Yes	6
	IAU_HOSTID	VARCHAR2(255 Bytes)	Yes	7
	IAU_HOSTNWADDR	VARCHAR2(255 Bytes)	Yes	8
	IAU_MODULEID	VARCHAR2(255 Bytes)	Yes	9
	IAU_PROCESSID	VARCHAR2(255 Bytes)	Yes	10
	IAU_ORACLEHOME	VARCHAR2(255 Bytes)	Yes	11
	IAU_HOMEINSTANCE	VARCHAR2(255 Bytes)	Yes	12
	IAU_UPSTREAMCOMPONENTID	VARCHAR2(255 Bytes)	Yes	13
	IAU_DOWNSTREAMCOMPONENTID	VARCHAR2(255 Bytes)	Yes	14
	IAU_ECID	VARCHAR2(255 Bytes)	Yes	15
	IAU_RID	VARCHAR2(255 Bytes)	Yes	16
	IAU_CONTEXTFIELDS	VARCHAR2(2000 Bytes)	Yes	17
	IAU_SESSIONID	VARCHAR2(255 Bytes)	Yes	18
	IAU_SECONDARYSESSIONID	VARCHAR2(255 Bytes)	Yes	19
	IAU_APPLICATIONNAME	VARCHAR2(255 Bytes)	Yes	20
	IAU_TARGETCOMPONENTTYPE	VARCHAR2(255 Bytes)	Yes	21
	IAU_EVENTTYPE	VARCHAR2(255 Bytes)	Yes	22
	IAU_EVENTCATEGORY	VARCHAR2(255 Bytes)	Yes	23
	IAU_EVENTSTATUS	NUMBER	Yes	24
	IAU_TSTZORIGINATING	TIMESTAMP(6)	Yes	25
	IAU_THREADID	VARCHAR2(255 Bytes)	Yes	26
	IAU_COMPONENTNAME	VARCHAR2(255 Bytes)	Yes	27
	IAU_INITIATOR	VARCHAR2(255 Bytes)	Yes	28

Table C-14 (Cont.) The Audit Schema

Table Name	Column Name	Data Type	Nullable	Column ID
	IAU_MESSAGETEXT	VARCHAR2(255 Bytes)	Yes	29
	IAU_FAILURECODE	VARCHAR2(255 Bytes)	Yes	30
	IAU_REMOTEIP	VARCHAR2(255 Bytes)	Yes	31
	IAU_TARGET	VARCHAR2(255 Bytes)	Yes	32
	IAU_RESOURCE	VARCHAR2(255 Bytes)	Yes	33
	IAU_ROLES	VARCHAR2(255 Bytes)	Yes	34
	IAU_AUTHENTICATIONMETHOD	VARCHAR2(255 Bytes)	Yes	35
	IAU_TRANSACTIONID	VARCHAR2(255 Bytes)	Yes	36
	IAU_DOMAINNAME	VARCHAR2(255 Bytes)	Yes	37
	IAU_COMPONENTDATA	clob	yes	38
DIP	IAU_ID	NUMBER	Yes	1
	IAU_TSTZORIGINATING	TIMESTAMP(6)	Yes	2
	IAU_EVENTTYPE	VARCHAR2(255 Bytes)	Yes	3
	IAU_EVENTCATEGORY	VARCHAR2(255 Bytes)	Yes	4
	IAU_ASSOCIATEPROFILENAME	VARCHAR2(512 Bytes)	Yes	5
	IAU_PROFILENAME	VARCHAR2(512 Bytes)	Yes	6
	IAU_ENTRYDN	VARCHAR2(1024 Bytes)	Yes	7
	IAU_PROVEVENT	VARCHAR2(2048 Bytes)	Yes	8
	IAU_JOBNAME	VARCHAR2(128 Bytes)	Yes	9
	IAU_JOBTYPE	VARCHAR2(128 Bytes)	Yes	10
IAU_DISP_NAME_TL	IAU_LOCALE_STR	VARCHAR2(7 Bytes)		1
	IAU_DISP_NAME_KEY	VARCHAR2(255 Bytes)		2
	IAU_COMPONENT_TYPE	VARCHAR2(255 Bytes)		3
	IAU_DISP_NAME_KEY_TYPE	VARCHAR2(255 Bytes)		4

Table C-14 (Cont.) The Audit Schema

Table Name	Column Name	Data Type	Nullable	Column ID
	IAU_DISP_NAME_TRANS	VARCHAR2(4000 Bytes)	Yes	5
IAU_LOCALE_MAP_TL	IAU_LOC_LANG	VARCHAR2(2 Bytes)	Yes	1
	IAU_LOC_CNTRY	VARCHAR2(3 Bytes)	Yes	2
	IAU_LOC_STR	VARCHAR2(7 Bytes)	Yes	3
OPSS	IAU_ID	NUMBER	Yes	1
	IAU_TSTZORIGINATING	TIMESTAMP(6)	Yes	2
	IAU_EVENTTYPE	VARCHAR2(255 Bytes)	Yes	3
	IAU_EVENTCATEGORY	VARCHAR2(255 Bytes)	Yes	4
	IAU_CODESOURCE	VARCHAR2(1024 Bytes)	Yes	5
	IAU_PRINCIPALS	VARCHAR2(1024 Bytes)	Yes	6
	IAU_INITIATORGUID	VARCHAR2(1024 Bytes)	Yes	7
	IAU_SUBJECT	VARCHAR2(1024 Bytes)	Yes	8
	IAU_PERMISSIONACTION	VARCHAR2(1024 Bytes)	Yes	9
	IAU_PERMISSIONTARGET	VARCHAR2(1024 Bytes)	Yes	10
	IAU_PERMISSIONCLASS	VARCHAR2(1024 Bytes)	Yes	11
	IAU_MAPNAME	VARCHAR2(1024 Bytes)	Yes	12
	IAU_KEY	VARCHAR2(1024 Bytes)	Yes	13
	IAU_PERMISSIONSCOPE	VARCHAR2(1024 Bytes)	Yes	14
	IAU_APPLICATIONROLE	VARCHAR2(1024 Bytes)	Yes	15
	IAU_ENTERPRISEROLES	VARCHAR2(1024 Bytes)	Yes	16
	IAU_INITIATORDN	VARCHAR2(1024 Bytes)	Yes	17
	IAU_GUID	VARCHAR2(1024 Bytes)	Yes	18
	IAU_PERMISSION	VARCHAR2(1024 Bytes)	Yes	19
	IAU_MODIFIEDATTRIBUTENAME	VARCHAR2(1024 Bytes)	Yes	20

Table C-14 (Cont.) The Audit Schema

Table Name	Column Name	Data Type	Nullable	Column ID
	IAU_MODIFIEDATTRIBUTEVALUE	VARCHAR2(2048 Bytes)	Yes	21
	IAU_PERMISSIONSETNAME	VARCHAR2(1024 Bytes)	Yes	22
	IAU_RESOURCEACTIONS	VARCHAR2(1024 Bytes)	Yes	23
	IAU_RESOURCETYPE	VARCHAR2(1024 Bytes)	Yes	24
OHS/OHS Component	IAU_ID	NUMBER	Yes	1
	IAU_TSTZORIGINATING	TIMESTAMP(6)	Yes	2
	IAU_EVENTTYPE	VARCHAR2(255 Bytes)	Yes	3
	IAU_EVENTCATEGORY	VARCHAR2(255 Bytes)	Yes	4
	IAU_REASON	CLOB	Yes	5
	IAU_SSLCONNECTION	VARCHAR2(255 Bytes)	Yes	6
	IAU_AUTHORIZATIONTYPE	VARCHAR2(255 Bytes)	Yes	7
OID/OID Component	IAU_ID	NUMBER	Yes	1
	IAU_TSTZORIGINATING	TIMESTAMP(6)	Yes	2
	IAU_EVENTTYPE	VARCHAR2(255 Bytes)	Yes	3
	IAU_EVENTCATEGORY	VARCHAR2(255 Bytes)	Yes	4
	IAU_CUSTEVENTSTATUSDETAIL	VARCHAR2(255 Bytes)	Yes	5
	IAU_CUSTEVENTTOP	VARCHAR2(255 Bytes)	Yes	6
OIF	IAU_ID	NUMBER	Yes	1
	IAU_TSTZORIGINATING	TIMESTAMP(6)	Yes	2
	IAU_EVENTTYPE	VARCHAR2(255 Bytes)	Yes	3
	IAU_EVENTCATEGORY	VARCHAR2(255 Bytes)	Yes	4
	IAU_REMOTEPROVIDERID	VARCHAR2(255 Bytes)	Yes	5
	IAU_PROTOCOLVERSION	VARCHAR2(255 Bytes)	Yes	6
	IAU_NAMEIDQUALIFIER	VARCHAR2(255 Bytes)	Yes	7

Table C-14 (Cont.) The Audit Schema

Table Name	Column Name	Data Type	Nullable	Column ID
	IAU_NAMEIDVALUE	VARCHAR2(255 Bytes)	Yes	8
	IAU_NAMEIDFORMAT	VARCHAR2(255 Bytes)	Yes	9
	IAU_SESSIONID	VARCHAR2(255 Bytes)	Yes	10
	IAU_FEDERATIONID	VARCHAR2(255 Bytes)	Yes	11
	IAU_USERID	VARCHAR2(255 Bytes)	Yes	12
	IAU_FEDERATIONTYPE	VARCHAR2(255 Bytes)	Yes	13
	IAU_AUTHENTICATIONMECHANISM	VARCHAR2(255 Bytes)	Yes	14
	IAU_AUTHENTICATIONENGINEID	VARCHAR2(255 Bytes)	Yes	15
	IAU_OLDNAMEIDQUALIFIER	VARCHAR2(255 Bytes)	Yes	16
	IAU_OLDNAMEIDVALUE	VARCHAR2(255 Bytes)	Yes	17
	IAU_BINDING	VARCHAR2(255 Bytes)	Yes	18
	IAU_ROLE	VARCHAR2(255 Bytes)	Yes	19
	IAU_MESSAGE_TYPE	VARCHAR2(255 Bytes)	Yes	20
	IAU_ASSERTIONVERSION	VARCHAR2(255 Bytes)	Yes	21
	IAU_ISSUEINSTANT	VARCHAR2(255 Bytes)	Yes	22
	IAU_ISSUER	VARCHAR2(255 Bytes)	Yes	23
	IAU_ASSERTIONID	VARCHAR2(255 Bytes)	Yes	24
	IAU_INCOMINGMESSAGESTRING	VARCHAR2(3999 Bytes)	Yes	25
	IAU_INCOMINGMESSAGESTRINGCLOB	CLOB	Yes	26
	IAU_OUTGOINGMESSAGESTRING	VARCHAR2(3999 Bytes)	Yes	27
	IAU_OUTGOINGMESSAGESTRINGCLOB	CLOB	Yes	28
	IAU_TYPE	VARCHAR2(255 Bytes)	Yes	29
	IAU_PROPERTYNAME	VARCHAR2(255 Bytes)	Yes	30

Table C-14 (Cont.) The Audit Schema

Table Name	Column Name	Data Type	Nullable	Column ID
	IAU_PROPERTYTYPE	VARCHAR2(255 Bytes)	Yes	31
	IAU_PEERPROVIDERID	VARCHAR2(255 Bytes)	Yes	32
	IAU_PROPERTYCONTEXT	VARCHAR2(255 Bytes)	Yes	33
	IAU_DESCRIPTION	VARCHAR2(255 Bytes)	Yes	34
	IAU_OLDVALUE	VARCHAR2(255 Bytes)	Yes	35
	IAU_NEWVALUE	VARCHAR2(255 Bytes)	Yes	36
	IAU_PROVIDERTYPE	VARCHAR2(255 Bytes)	Yes	37
	IAU_COTBEFORE	CLOB	Yes	38
	IAU_COTAFTER	CLOB	Yes	39
	IAU_SERVERCONFIGBEFORE	CLOB	Yes	40
	IAU_SERVERCONFIGAFTER	CLOB	Yes	41
	IAU_DATASTOREBEFORE	CLOB	Yes	42
	IAU_DATASTOREAFTER	CLOB	Yes	43
	IAU_METADATA	VARCHAR2(255 Bytes)	Yes	44
	IAU_NEWDATASTORETYPE	VARCHAR2(255 Bytes)	Yes	45
	IAU_DATASTORENAME	VARCHAR2(255 Bytes)	Yes	46
OVD/OVD Component	IAU_ID	NUMBER	Yes	1
	IAU_TSTZORIGINATING	TIMESTAMP(6)	Yes	2
	IAU_EVENTTYPE	VARCHAR2(255 Bytes)	Yes	3
	IAU_EVENTCATEGORY	VARCHAR2(255 Bytes)	Yes	4
	IAU_SERVICEOPERATION	VARCHAR2(255 Bytes)	Yes	5
OWSM Agent	IAU_ID	NUMBER	Yes	1
	IAU_TSTZORIGINATING	TIMESTAMP(6)	Yes	2
	IAU_EVENTTYPE	VARCHAR2(255 Bytes)	Yes	3
	IAU_EVENTCATEGORY	VARCHAR2(255 Bytes)	Yes	4
	IAU_APPNAME	VARCHAR2(255 Bytes)	Yes	5

Table C-14 (Cont.) The Audit Schema

Table Name	Column Name	Data Type	Nullable	Column ID
	IAU_ASSERTIONNAME	VARCHAR2(255 Bytes)	Yes	6
	IAU_COMPOSITENAME	VARCHAR2(255 Bytes)	Yes	7
	IAU_ENDPOINT	VARCHAR2(4000 Bytes)	Yes	8
	IAU_AGENTMODE	VARCHAR2(255 Bytes)	Yes	9
	IAU_MODELOBJECTNAME	VARCHAR2(255 Bytes)	Yes	10
	IAU_OPERATION	VARCHAR2(255 Bytes)	Yes	11
	IAU_PROCESSINGSTAGE	VARCHAR2(255 Bytes)	Yes	12
	IAU_VERSION	NUMBER	Yes	13
	IAU_PROTOCOL	VARCHAR2(255 Bytes)	Yes	14
OWSM_PM_EJB	IAU_ID	NUMBER	Yes	1
	IAU_TSTZORIGINATING	TIMESTAMP(6)	Yes	2
	IAU_EVENTTYPE	VARCHAR2(255 Bytes)	Yes	3
	IAU_EVENTCATEGORY	VARCHAR2(255 Bytes)	Yes	4
	IAU_VERSION	NUMBER	Yes	5
	IAU_TOVERSION	NUMBER	Yes	6
ReportsServer/ReportsServer Components	IAU_ID	NUMBER	Yes	1
	IAU_TSTZORIGINATING	TIMESTAMP(6)	Yes	2
	IAU_EVENTTYPE	VARCHAR2(255 Bytes)	Yes	3
	IAU_EVENTCATEGORY	VARCHAR2(255 Bytes)	Yes	4
WebCache/WebCache Component	IAU_ID	NUMBER	Yes	1
	IAU_TSTZORIGINATING	TIMESTAMP(6)	Yes	2
	IAU_EVENTTYPE	VARCHAR2(255 Bytes)	Yes	3
	IAU_EVENTCATEGORY	VARCHAR2(255 Bytes)	Yes	4

Table C-14 (Cont.) The Audit Schema

Table Name	Column Name	Data Type	Nullable	Column ID
WebServices	IAU_ID	NUMBER	Yes	1
	IAU_TSTZORIGINATING	TIMESTAMP(6)	Yes	2
	IAU_EVENTTYPE	VARCHAR2(255 Bytes)	Yes	3
	IAU_EVENTCATEGORY	VARCHAR2(255 Bytes)	Yes	4
	IAU_PROTOCOL	VARCHAR2(255 Bytes)	Yes	5
	IAU_ENDPOINT	VARCHAR2(4000 Bytes)	Yes	6
	IAU_OPERATION	VARCHAR2(255 Bytes)	Yes	7
	IAU_FAULTURI	VARCHAR2(4000 Bytes)	Yes	8
	IAU_URI	VARCHAR2(4000 Bytes)	Yes	9
	IAU_SOURCE	VARCHAR2(255 Bytes)	Yes	10
WS_Policy Attachment	IAU_ID	NUMBER	Yes	1
	IAU_TSTZORIGINATING	TIMESTAMP(6)	Yes	2
	IAU_EVENTTYPE	VARCHAR2(255 Bytes)	Yes	3
	IAU_EVENTCATEGORY	VARCHAR2(255 Bytes)	Yes	4
	IAU_PROTOCOL	VARCHAR2(255 Bytes)	Yes	5
	IAU_ENDPOINT	VARCHAR2(4000 Bytes)	Yes	6
	IAU_OPERATION	VARCHAR2(255 Bytes)	Yes	7
	IAU_FAULTURI	VARCHAR2(4000 Bytes)	Yes	8
	IAU_URI	VARCHAR2(4000 Bytes)	Yes	9
	IAU_SOURCE	VARCHAR2(255 Bytes)	Yes	10
OAM (Oracle Access Manager)	IAU_ID	NUMBER	Yes	1
	IAU_TSTZORIGINATING	TIMESTAMP(6)	Yes	2
	IAU_EVENTTYPE	VARCHAR2(255)	Yes	3
	IAU_EVENTCATEGORY	VARCHAR2(255)	Yes	4

Table C-14 (Cont.) The Audit Schema

Table Name	Column Name	Data Type	Nullable	Column ID
	IAU_APPLICATIONDOMAINNAME	VARCHAR2(40)	Yes	5
	IAU_AUTHENTICATIONSchemeID	VARCHAR2(40)	Yes	6
	IAU_AGENTID	VARCHAR2(40)	Yes	7
	IAU_SSOSESSIONID	VARCHAR2(100)	Yes	8
	IAU_ADDITIONALINFO	VARCHAR2(1000)	Yes	9
	IAU_AUTHORIZATIONScheme	VARCHAR2(40)	Yes	10
	IAU_USERDN	VARCHAR2(255)	Yes	11
	IAU_RESOURCEID	VARCHAR2(40)	Yes	12
	IAU_AUTHORIZATIONPOLICYID	VARCHAR2(40)	Yes	13
	IAU_AUTHENTICATIONPOLICYID	VARCHAR2(255)	Yes	14
	IAU_USERID	VARCHAR2(40)	Yes	15
	IAU_RESOURCEHOST	VARCHAR2(255)	Yes	16
	IAU_REQUESTID	VARCHAR2(255)	Yes	17
	IAU_POLICYNAME	VARCHAR2(40)	Yes	18
	IAU_SCHEMENAME	VARCHAR2(40)	Yes	19
	IAU_RESOURCEHOSTNAME	VARCHAR2(100)	Yes	20
	IAU_OLDATTRIBUTES	VARCHAR2(1000)	Yes	21
	IAU_NEWATTRIBUTES	VARCHAR2(1000)	Yes	22
	IAU_SCHMETYPE	VARCHAR2(40)	Yes	23
	IAU_RESPONSETYPE	VARCHAR2(40)	Yes	24
	IAU_AGENTTYPE	VARCHAR2(40)	Yes	25
	IAU_CONSTRAINTTYPE	VARCHAR2(40)	Yes	26
	IAU_INSTANCENAME	VARCHAR2(40)	Yes	27
	IAU_DATASOURCENAME	VARCHAR2(100)	Yes	28
	IAU_DATASOURCETYPE	VARCHAR2(100)	Yes	29
	IAU_HOSTIDENTIFIERNAME	VARCHAR2(100)	Yes	30
	IAU_RESOURCEURI	VARCHAR2(255)	Yes	31
	IAU_RESOURCETEMPLATENAME	VARCHAR2(100)	Yes	32
OAAM (Oracle Adaptive Access Manager)	IAU_ID	NUMBER	Yes	1
	IAU_TSTZORIGINATING	TIMESTAMP(6)	Yes	2
	IAU_EVENTTYPE	VARCHAR2(255)	Yes	3
	IAU_EVENTCATEGORY	VARCHAR2(255)	Yes	4
	IAU_ACTIONNOTES	VARCHAR2(4000)	Yes	5
	IAU_CASEACTIONENUM	NUMBER(38)	Yes	6
	IAU_CASEACTIONRESULT	NUMBER	Yes	7

Table C-14 (Cont.) The Audit Schema

Table Name	Column Name	Data Type	Nullable	Column ID
	IAU_CASECHALLENGEQUESTION	VARCHAR2(4000)	Yes	8
	IAU_CASECHALLENGERESULT	NUMBER(38)	Yes	9
	IAU_CASEDISPOSITION	NUMBER(38)	Yes	10
	IAU_CASEEXPRDURATIONINHRS	NUMBER(38)	Yes	11
	IAU_CASEID	NUMBER	Yes	12
	IAU_CASEIDS	VARCHAR2(4000)	Yes	13
	IAU_CASESEVERITY	NUMBER(38)	Yes	14
	IAU_CASESTATUS	NUMBER(38)	Yes	15
	IAU_CASESUBACTIONENUM	NUMBER(38)	Yes	16
	IAU_DESCRIPTION	VARCHAR2(4000)	Yes	17
	IAU_GROUPID	NUMBER	Yes	18
	IAU_GROUPIDS	VARCHAR2(4000)	Yes	19
	IAU_GROUPNAME	VARCHAR2(4000)	Yes	20
	IAU_GROUPDETAILS	VARCHAR2(4000)	Yes	21
	IAU_GROUPELEMENTID	NUMBER	Yes	22
	IAU_GROUPELEMENTIDS	NUMBER	Yes	23
	IAU_GROUPELEMENTVALUE	VARCHAR2(4000)	Yes	24
	IAU_GROUPELEMENTSDetails	VARCHAR2(4000)	Yes	25
	IAU_KBACATEGORYID	NUMBER	Yes	26
	IAU_KBACATEGORYIDS	VARCHAR2(4000)	Yes	27
	IAU_KBACATEGORYNAME	VARCHAR2(4000)	Yes	28
	IAU_KBACATEGORYDETAILS	VARCHAR2(4000)	Yes	29
	IAU_KBAQUESTIONID	NUMBER	Yes	30
	IAU_KBAQUESTIONIDS	VARCHAR2(4000)	Yes	31
	IAU_KBAQUESTION	VARCHAR2(4000)	Yes	32
	IAU_KBAQUESTIONTYPE	NUMBER(38)	Yes	33
	IAU_KBAQUESTIONDETAILS	VARCHAR2(4000)	Yes	34
	IAU_KBAVALIDATIONID	NUMBER	Yes	35
	IAU_KBAVALIDATIONIDS	VARCHAR2(4000)	Yes	36
	IAU_KBAVALIDATIONNAME	VARCHAR2(4000)	Yes	37
	IAU_KBAVALIDATIONDETAILS	VARCHAR2(4000)	Yes	38
	IAU_KBAREGLOGICDETAILS	VARCHAR2(4000)	Yes	39
	IAU_KBAANSWERLOGICDETAILS	VARCHAR2(4000)	Yes	40
	IAU_LOGINID	VARCHAR2(255)	Yes	41
	IAU_POLICYDETAILS	VARCHAR2(4000)	Yes	42
	IAU_POLICYID	NUMBER	Yes	43
	IAU_POLICYIDS	VARCHAR2(4000)	Yes	44
	IAU_POLICYNAME	NUMBER	Yes	45
	IAU_POLICYOVERRIDEDETAILS	VARCHAR2(4000)	Yes	46

Table C-14 (Cont.) The Audit Schema

Table Name	Column Name	Data Type	Nullable	Column ID
	IAU_POLICYOVERRIDEID	NUMBER	Yes	47
	IAU_POLICYOVERRIDEIDS	VARCHAR2(4000)	Yes	48
	IAU_POLICYOVERRIDEROWID	NUMBER	Yes	49
	IAU_POLICYRULEMAPID	NUMBER	Yes	50
	IAU_POLICYRULEMAPIDS	VARCHAR2(4000)	Yes	51
	IAU_POLICYRULEMAPDETAILS	VARCHAR2(4000)	Yes	52
	IAU_RULEID	NUMBER	Yes	53
	IAU_RULECONDITIONID	NUMBER	Yes	54
	IAU_RULECONDITIONIDS	VARCHAR2(4000)	Yes	55
	IAU_RULENAME	VARCHAR2(4000)	Yes	56
	IAU_RULEDETAILS	VARCHAR2(4000)	Yes	57
	IAU_RULECONDITIONMAPID	NUMBER	Yes	58
	IAU_RULECONDITIONMAPIDS	VARCHAR2(4000)	Yes	59
	IAU_RULEPARAMVALUEDETAILS	VARCHAR2(4000)	Yes	60
	IAU_SOURCEPOLICYID	NUMBER	Yes	61
	IAU_USERGROUPNAME	VARCHAR2(255)	Yes	62
	IAU_USERID	NUMBER	Yes	63
	IAU_USERIDS	VARCHAR2(4000)	Yes	64

Table C-15 shows additional tables in the audit schema; these tables support the dynamic metadata model.

Table C-15 Additional Audit Schema Tables

Table Name	Column Name	Data Type
IAU_COMMON	IAU_ID	NUMBER
	IAU_OrgId	VARCHAR(255)
	IAU_ComponentId	VARCHAR(255)
	IAU_ComponentType	VARCHAR(255)
	IAU_MajorVersion	VARCHAR(255)
	IAU_MinorVersion	VARCHAR(255)
	IAU_InstanceId	VARCHAR(255)
	IAU_HostingClientId	VARCHAR(255)
	IAU_HostId	VARCHAR(255)
	IAU_HostNwaddr	VARCHAR(255)
	IAU_ModuleId	VARCHAR(255)
	IAU_ProcessId	VARCHAR(255)
	IAU_OracleHome	VARCHAR(255)
	IAU_HomeInstance	VARCHAR(255)
	IAU_UpstreamComponentId	VARCHAR(255)

Table C-15 (Cont.) Additional Audit Schema Tables

Table Name	Column Name	Data Type
	IAU_DownstreamComponentId	VARCHAR(255)
	IAU_ECID	VARCHAR(255)
	IAU_RID	VARCHAR(255)
	IAU_ContextFields	VARCHAR(2000)
	IAU_SessionId	VARCHAR(255)
	IAU_SecondarySessionId	VARCHAR(255)
	IAU_ApplicationName	VARCHAR(255)
	IAU_TargetComponentType	VARCHAR(255)
	IAU_EventType	VARCHAR(255)
	IAU_EventCategory	VARCHAR(255)
	IAU_EventStatus	NUMBER
	IAU_TstzOriginating	TIMESTAMP
	IAU_ThreadId	VARCHAR(255)
	IAU_ComponentName	VARCHAR(255)
	IAU_Initiator	VARCHAR(255)
	IAU_MessageText	VARCHAR(2000)
	IAU_FailureCode	VARCHAR(255)
	IAU_RemoteIP	VARCHAR(255)
	IAU_Target	VARCHAR(255)
	IAU_Resource	VARCHAR(255)
	IAU_Roles	VARCHAR(255)
	IAU_AuthenticationMethod	VARCHAR(255)
	IAU_TransactionId	VARCHAR(255)
	IAU_DomainName	VARCHAR(255)
	IAU_ComponentVersion	VARCHAR(255)
	IAU_ComponentData	CLOB
IAU_CUSTOM	IAU_ID	NUMBER
	IAU_BOOLEAN_001 through IAU_BOOLEAN_050	NUMBER
	IAU_INT_001 through IAU_INT_050	NUMBER
	IAU_LONG_001 through IAU_LONG_050	NUMBER
	IAU_FLOAT_001 through IAU_FLOAT_050	NUMBER

Table C–15 (Cont.) Additional Audit Schema Tables

Table Name	Column Name	Data Type
	IAU_DOUBLE_001 through IAU_DOUBLE_050	NUMBER
	IAU_STRING_001 through IAU_STRING_100	VARCHAR(2048)
	IAU_DATETIME_001 through IAU_DATETIME_050	TIMESTAMP
	IAU_LONGSTRING_001 through IAU_LONGSTRING_050	CLOB
	IAU_BINARY_001 through IAU_BINARY_050	BLOB
IAU_AuditService	IAU_ID	NUMBER
	IAU_TransactionId	VARCHAR(255)
IAU_USERSSESSION	IAU_ID	NUMBER
	IAU_AuthenticationMethod	VARCHAR(255)

C.4 WLST Commands for Auditing

WLST is the command-line utility for administration of Oracle Fusion Middleware components and applications. It provides another option for administration in addition to Oracle Enterprise Manager Fusion Middleware Control.

Use the WLST commands listed in [Table C–16](#) to view and manage audit policies and the audit store configuration.

Note: When running audit WLST commands, you must invoke the WLST script from the Oracle Common home. See "Using Custom WLST Commands" in the *Oracle Fusion Middleware Administrator's Guide* for more information.

See Also: *Oracle Fusion Middleware Third-Party Application Server Guide* for details about executing audit commands on third-party application servers.

Table C–16 WLST Audit Commands

Use this command...	To...	Use with WLST...
getNonJavaEEAuditMBeanName	Display the mBean name for a system component.	Online
getAuditPolicy	Display audit policy settings.	Online

Table C-16 (Cont.) WLST Audit Commands

Use this command...	To...	Use with WLST...
setAuditPolicy	Update audit policy settings.	Online
getAuditRepository	Display audit store settings.	Online
setAuditRepository	Update audit store settings.	Online
listAuditEvents	List audit events for one or all components.	Online
exportAuditConfig	Export a component's audit configuration.	Online
importAuditConfig	Import a component's audit configuration.	Online

C.4.1 getNonJava EEAuditMBeanName

Online command that displays the mbean name for system components.

The MBean name must be provided when using WLST commands for system components; since the MBean name can have a complex composition, use this command to get the name.

C.4.1.1 Description

This command displays the mbean name for system components given the instance name, component name, component type, and the name of the Oracle WebLogic Server on which the component's audit mbean is running. The mbean name is a required parameter to other audit WLST commands when managing a system component.

C.4.1.2 Syntax

```
getNonJava EEAuditMBeanName('instance-name', 'component-name', 'component-type')
```

Argument	Definition
instName	Specifies the name of the application server instance.
compName	Specifies the name of the component instance.
compType	Specifies the type of component. Valid values are ohs, oid, ovd, and WebCache.

C.4.1.3 Example

The following interactive command displays the mBean name for an Oracle Internet Directory component:

```
wls:/mydomain/serverConfig> getNonJava EEAuditMBeanName (instName='inst1',
compName='oid1', compType='oid')
```

C.4.2 getAuditPolicy

Online command that displays the audit policy settings.

C.4.2.1 Description

Online command that displays audit policy settings including the audit level, special users, custom events, maximum log file size, and maximum log directory size. The

component mbean name is an optional parameter. If no parameter is provided, the domain audit policy is displayed.

C.4.2.2 Syntax

```
getAuditPolicy(['mbeanName', componentType])
```

Argument	Definition
mbeanName	Specifies the name of the component audit MBean for system components.
componentType	Requests the audit policy for a specific component type registered in the audit store. If not specified, the audit policy in <code>jps-config.xml</code> is returned.

C.4.2.3 Example

The following command displays the audit settings for all Java EE components configured in the WebLogic Server domain:

```
wls:/mydomain/serverConfig> getAuditPolicy()
```

The following command displays the audit settings for MBean `CSAuditProxyMBean`:

```
wls:/mydomain/serverConfig>
getAuditPolicy(on='oracle.security.audit.test:type=CSAuditMBean,
name=CSAuditProxyMBean')
```

C.4.3 setAuditPolicy

Online command that updates an audit policy.

C.4.3.1 Description

Online command that configures the audit policy settings. You can set the audit level, add or remove special users, and add or remove custom events. The component mbean name is required for system components like Oracle Internet Directory and Oracle Virtual Directory.

Remember to call `save` after issuing `setAuditPolicy` for system components. Otherwise, the new settings will not take effect.

C.4.3.2 Syntax

```
setAuditPolicy(['mbeanName'], ['filterPreset'], ['addSpecialUsers'],
['removeSpecialUsers'], ['addCustomEvents'], ['removeCustomEvents'],
[componentType], [maxDirSize], [maxFileSize], [andCriteria], [orCriteria],
[componentEventsFile])
```

Argument	Definition
mbeanName	Specifies the name of the component audit MBean for system components.
filterPreset	Specifies the audit level to be changed.
addSpecialUsers	Specifies the special users to be added.
removeSpecialUsers	Specifies the special users to be removed.
addCustomEvents	Specifies the custom events to be added.
removeCustomEvents	Specifies the custom events to be removed.

Argument	Definition
componentType	Specifies the component definition type to be updated. If not specified, the audit configuration defined in <code>jps-config.xml</code> is modified.
maxDirSize	Specifies the maximum size of the log directory.
maxFileSize	Specifies the maximum size of the log file.
andCriteria	Specifies the <code>and</code> criteria in a custom filter preset definition.
orCriteria	Specifies the <code>or</code> criteria in a custom filter preset definition.
componentEventsFile	Specifies a component definition file under the 11g Release 1 (11.1.1) PS5 metadata model. This parameter is required if you wish to create/update an audit policy in the audit store for an 11g Release 1 (11.1.1) PS5 metadata model component, and the filter preset level is set to "Custom".

C.4.3.3 Example

The following interactive command a) sets the audit level to `Low`, and b) adds users `user2` and `user3` while removing user `user1` from the policy:

```
wls:/mydomain/serverConfig> setAuditPolicy
(filterPreset='Low',addSpecialUsers='user2,user3',removeSpecialUsers='user1')
```

The following interactive command adds login events while removing logout events from the policy:

```
wls:/mydomain/serverConfig>
setAuditPolicy(filterPreset='Custom',addCustomEvents='UserLogin',removeCustomEvent
s='UserLogout')
```

C.4.4 getAuditRepository

Online command that displays audit store settings.

C.4.4.1 Description

Online command that displays audit store settings for Java components and applications (for system components like Oracle Internet Directory, the configuration resides in `opmn.xml`). Also displays database configuration if the data is stored in a database.

C.4.4.2 Syntax

```
getAuditRepository
```

C.4.4.3 Example

The following command displays audit store configuration:

```
wls:/mydomain/serverConfig> getAuditRepository()
```

C.4.5 setAuditRepository

Online command that updates audit store settings.

C.4.5.1 Description

Online command that sets the audit store settings for Java components and applications (for system components like Oracle Internet Directory, the store is configured by editing `opmn.xml`).

C.4.5.2 Syntax

```
setAuditRepository(['switchToDB'], ['dataSourceName'], ['interval'])
```

Argument	Definition
switchToDB	If <code>true</code> , switches the store from file to database.
dataSourceName	Specifies the name of the data source.
interval	Specifies intervals at which the audit loader moves file records to the database.

C.4.5.3 Example

The following interactive command changes audit store to a database defined by the data source `jdbcAuditDB` and sets the audit loader interval to 14 seconds:

```
wls:/mydomain/serverConfig>
setAuditRepository(switchToDB='true', dataSourceName='jdbcAuditDB', interval='14')
```

Note: The data source is created using the Oracle WebLogic Server administration console.

C.4.6 listAuditEvents

Online command that displays the definition of a component's audit events, including its attributes.

C.4.6.1 Description

This command displays a component's audit events and attributes. For system components, pass the component mbean name as a parameter. Java applications and services like Oracle Platform Security Services (OPSS) do not need the mbean parameter. Without a component type, all generic attributes applicable to all components are displayed.

C.4.6.2 Syntax

```
listAuditEvents(['mbeanName'], ['componentType'])
```

Argument	Definition
mbeanName	Specifies the name of the component MBean.
componentType	Specifies the component type to limit the list to all events of the component type.

C.4.6.3 Example

The following command displays audit events for an Oracle Internet Directory instance:

```
wls:/mydomain/serverConfig>
listAuditEvents(on='oracle.as.management.mbeans.register:
```

```
type=component.auditconfig,name=auditconfig1,instance=oid1,component=oid')
```

The following command displays audit events for Oracle Identity Federation:

```
wls:/mydomain/serverConfig> listAuditEvents(componentType='oif')
```

C.4.7 exportAuditConfig

Online command that exports a component's audit configuration.

See Also: This command is useful in migrating to production environments. For details, see [Section 6.5.3, "Migrating Audit Policies"](#).

C.4.7.1 Description

This command exports the audit configuration to a file. For system components, pass the component mbean name as a parameter. Java applications and services like Oracle Platform Security Services (OPSS) do not need the mbean parameter.

C.4.7.2 Syntax

```
exportAuditConfig(['mbeanName'], 'fileName', [componentType])
```

Argument	Definition
mbeanName	Specifies the name of the system component MBean.
fileName	Specifies the path and file name to which the audit configuration should be exported.
componentType	Specifies that only events of the given component be exported to the file. If not specified, the audit configuration in <code>jps-config.xml</code> is exported.

C.4.7.3 Example

The following interactive command exports the audit configuration for a component:

```
wls:/mydomain/serverConfig>
exportAuditConfig(on='oracle.security.audit.test:type=CSAAuditMBean,name=CSAAuditPro
xyMBean', fileName='/tmp/auditconfig')
```

The following interactive command exports the audit configuration for a component; no mBean is specified:

```
wls:/mydomain/serverConfig> exportAuditConfig(fileName='/tmp/auditconfig')
```

C.4.8 importAuditConfig

Online command that imports a component's audit configuration.

See Also: This command is useful in migrating to production environments. For details, see [Section 6.5.3, "Migrating Audit Policies"](#).

C.4.8.1 Description

This command imports the audit configuration from an external file. For system components, pass the component mbean name as a parameter. Java applications and services like Oracle Platform Security Services (OPSS) do not need the mbean parameter.

Remember to call `save` after issuing `importAuditConfig` for system components. Otherwise, the new settings will not take effect.

C.4.8.2 Syntax

```
importAuditConfig(['mbeanName'], 'fileName', [componentType])
```

Argument	Definition
mbeanName	Specifies the name of the system component MBean.
fileName	Specifies the path and file name from which the audit configuration should be imported.
componentType	Specifies that only events of the given component be imported from the file. If not specified, the audit configuration in <code>jps-config.xml</code> is imported.

C.4.8.3 Example

The following interactive command imports the audit configuration for a component:

```
wls:/mydomain/serverConfig>
importAuditConfig(on='oracle.security.audit.test:type=CSAuditMBean,name=CSAuditPro
xyMBean',fileName='/tmp/auditconfig')
```

The following interactive command imports the audit configuration for a Java EE application (no mBean is specified):

```
wls:/mydomain/serverConfig> importAuditConfig(fileName='/tmp/auditconfig')
```

C.5 Audit Filter Expression Syntax

When you select a custom audit policy, you have the option of specifying a filter expression along with an event.

For example, you can use the following expression:

```
Host Id -eq "myhost123"
```

to enable the audit event for a particular host only.

You enter this expression either through the Fusion Middleware Control Edit Filter Dialog or through the `setAuditPolicy WLST` command.

See Also:

- [Section 13.3.1, "Manage Audit Policies for Java Components with Fusion Middleware Control"](#)
- [Section 13.3.2, "Manage Audit Policies for System Components with Fusion Middleware Control"](#)
- [Section C.4.3, "setAuditPolicy"](#)

There are some syntax rules you should follow when creating a filter expression.

The expression can either be a Boolean expression or a literal.

```
<Expr> ::= <BooleanExpression> | <BooleanLiteral>
```

A boolean expression can use combinations of RelationalExpression with `-and`, `-or`, `-not` and parenthesis. For example, `(Host Id -eq "stad117" -or)`.

```
<BooleanExpression> ::= <RelationalExpression>
| "(" <BooleanExpression> ")"
| <BooleanExpression> "-and" <BooleanExpression>
| <BooleanExpression> "-or" <BooleanExpression>
| "-not" <BooleanExpression>
```

A relational expression compares an attribute name (on the left hand side) with a literal (on the right-hand side). The literal and the operator must be of the correct data type for the attribute.

```
<RelationalExpression> ::= <AttributeName> <RelationalOperator> <Literal>
```

Relational operators are particular to data types:

- `-eq`, `-ne` can be used with all data types
- `-contains`, `-startswith`, `-endswith` can be only used with strings
- `-contains_case`, `-startswith_case` and `-endswith_case` are case sensitive versions of the above three functions
- `-lt`, `-le`, `-gt`, `-ge` can be used with numeric and datetime

```
<RelationalOperator> := "-eq" | "-ne" | "-lt" | "-le" | "-gt" | "-ge"
| "-contains" | "-contains_case"
| "-startswith" | "-startswith_case"
| "-endswith" | "-endswith_case"
```

Rules for literals are as follows:

- Boolean literals are true or false, without quotes
- Date time literals have to be in double quotes and can be in many different formats; "June 25, 2006", "06/26/2006 2:00 pm" are all valid
- String literals have to be quotes, back-slash can be used to escape an embedded double quote
- Numeric literals are in their usual format

```
<Literal> ::= <NumericLiteral> | <BooleanLiteral> | <DateTimeLiteral> |
<StringLiteral>
<BooleanLiteral> ::= "true" | "false"
```

C.6 Naming and Logging Format of Audit Files

This section explains the rules that are used to maintain audit files.

For Java components (both Java EE and Java SE), the file containing audit records is named "audit.log".

When that file is full (it reaches the configured maximum audit file size which is 100MB), it is renamed to "audit1.log" and a new "audit.log" is created. If this file too gets full, the audit.log file is renamed to "audit2.log" and a new audit.log is created.

This continues until the configured maximum audit directory size is reached (default is 0, which means unlimited size). When the max directory size is reached, the oldest auditn.log file is deleted.

If you have configured a database audit store, then the audit loader reads these files and transfers the records to the database in batches. After reading a complete audit<n>.log file, it deletes the file.

Note: The audit loader never deletes the "current" file, that is, audit.log; it only deletes archive files audit<n>.log.

OPMN-managed components follow the same model, except the file name is slightly different. It has the process ID embedded in the file name; thus, if the process id is 11925 the current file is called "audit-pid11925.log", and after rotation it will be called audit-pid11925-1.log.

For applications with audit definitions in the dynamic model, the file name format is *audit_major version number_minor version number.log*; for example, *audit_1_2.log*.

Here is a sample audit.log file:

```
#Fields:Date Time Initiator EventType EventStatus MessageText HomeInstance ECID
RID ContextFields SessionId TargetComponentType ApplicationName EventCategory
ThreadId InitiatorDN TargetDN FailureCode RemoteIP Target Resource Roles
CodeSource InitiatorGUID Principals PermissionAction PermissionClass mapName key
#Remark Values:ComponentType="JPS"
2008-12-08 10:46:05.492 - "CheckAuthorization" true "Oracle Platform Security
Authorization Check Permission SUCCEEDED." - - - - - "Authorization" "48" - -
"true" - - "(oracle.security.jps.service.policystore.PolicyStoreAccessPermission
context=APPLICATION,name=SimpleServlet getApplicationPolicy)" -
"file:/oracle/work/middleware/oracle_common/modules/oracle.jps_
11.1.1/jps-internal.jar" - "[]" - - - -
```

This file follows the W3C extended logging format, which is a very common log format that is used by many Web Servers e.g. Apache and IIS:

- The first line is a "#Fields" line; it specifies all the fields in the rest of the file.
- The second line is a comment like "#Remark" which has a comment indicating some common attributes like the ComponentType.
- All subsequent lines are data lines; they follow the exact format defined in the "#Fields" line. All attributes are separated by spaces, missing attributes are indicated by a dash.

User and Role API Reference

This appendix contains reference information that you will need when developing applications for LDAP directories based on the User and Role APIs. It contains these sections:

- [Mapping User Attributes to LDAP Directories](#)
- [Mapping Role Attributes to LDAP Directories](#)
- [Default Configuration Parameters](#)
- [Secure Connections for Microsoft Active Directory](#)

See Also: [Chapter 25, "Developing with the User and Role API"](#)

Note: IBM Tivoli directory parameters are the same as those specified for openLDAP.

Microsoft ADAM parameters are the same as those specified for Microsoft Active Directory.

D.1 Mapping User Attributes to LDAP Directories

[Table D-1](#) lists each user attribute in `UserProfile.property` and its corresponding attribute in the different directory servers.

Table D-1 *User Attributes in UserProfile.Property*

Attribute	Oracle Internet Directory	Oracle WebLogic Server Embedded LDAP	Microsoft Active Directory	Oracle Directory Server Enterprise Edition	Novell eDirectory	OpenLDAP
GUID	orclguid	uid	objectguid	nsuniqueid	guid	entryuuid
USER_ID	username (see Note below)	uid	uid	uid	uid	uid
DISPLAY_NAME	displayname	displayname	displayname	displayname	displayname	displayname
BUSINESS_EMAIL	mail	mail	mail	mail	mail	mail
DESCRIPTION	description	description	description	description	description	description
EMPLOYEE_TYPE	employeeType	employeeType	employeeType	employeeType	employeeType	employeeType

Table D-1 (Cont.) User Attributes in UserProfile.Property

Attribute	Oracle Internet Directory	Oracle WebLogic Server Embedded LDAP	Microsoft Active Directory	Oracle Directory Server Enterprise Edition	Novell eDirectory	OpenLDAP
DEPARTMENT	departmentnumber	departmentnumber	departmentnumber	departmentnumber	departmentnumber	departmentnumber
DATE_OF_BIRTH	orcldateofbirth	-	-	-	-	-
BUSINESS_FAX	facsimiletelephonenumber	facsimiletelephonenumber	facsimiletelephonenumber	facsimiletelephonenumber	facsimiletelephonenumber	facsimiletelephonenumber
BUSINESS_CITY	l	l	l	l	l	l
BUSINESS_COUNTRY	c	c	c	c	c	c
DATE_OF_HIRE	orclhiredate	-	-	-	-	-
NAME	cn	uid	cn	uid	cn	cn
PREFERRED_LANGUAGE	Preferredlanguage	preferredlanguage	preferredlanguage	preferredlanguage	preferredlanguage	preferredlanguage
BUSINESS_POSTAL_ADDR	postaladdresses	postaladdresses	postaladdress	postaladdresses	postaladdress	postaladdresses
MIDDLE_NAME	orclmiddlename	-	-	-	-	-
ORGANIZATIONAL_UNIT	ou	ou	ou	ou	ou	ou
WIRELESS_ACCT_NUMBER	orclwirelessaccountnumber	-	-	-	-	-
BUSINESS_PO_BOX	postofficebox	postofficebox	postofficebox	postofficebox	postofficebox	postofficebox
BUSINESS_STATE	St	st	st	st	st	st
HOME_ADDRESS	Homepostaladdress	homepostaladdress	homepostaladdress	homepostaladdress	homepostaladdress	homepostaladdress
NAME_SUFFIX	Generationqualifier	generationqualifier	generationqualifier	generationqualifier	generationqualifier	generationqualifier
BUSINESS_STREET	street	street	street	street	street	street
INITIALS	initials	initials	initials	initials	initials	initials
USER_NAME	username (see Note below)	uid	samaccountname	uid	uid	uid
BUSINESS_POSTAL_CODE	postalcode	postalcode	postalcode	postalcode	postalcode	postalcode
BUSINESS_PAGER	pager	pager	pager	pager	pager	pager

Table D-1 (Cont.) User Attributes in UserProfile.Property

Attribute	Oracle Internet Directory	Oracle WebLogic Server Embedded LDAP	Microsoft Active Directory	Oracle Directory Server Enterprise Edition	Novell eDirectory	OpenLDAP
LAST_NAME	sn	sn	sn	sn	sn	sn
BUSINESS_PHONE	telephonenumber	telephonenumber	telephonenumber	telephonenumber	telephonenumber	telephonenumber
FIRST_NAME	givenname	givenname	givenname	givenname	givenname	givenname
TIME_ZONE	orcltimezone	-	-	-	-	-
MAIDEN_NAME	orclmaidename	-	-	-	-	-
PASSWORD	userpassword	userpassword	userpassword	userpassword	userpassword	userpassword
DEFAULT_GROUP	orcldefaultprofilegroup	-	-	-	-	-
ORGANIZATION	o	o	o	o	o	o
HOME_PHONE	homephone	homephone	homephone	homephone	homephone	homephone
BUSINESS_MOBILE	mobile	mobile	mobile	mobile	mobile	mobile
UI_ACCESS_MODE	orcluiaccessibilitymode	-	-	-	-	-
JPEG_PHOTO	jpegphoto	jpegphoto	jpegphoto	jpegphoto	jpegphoto	jpegphoto
MANAGER	manager	manager	manager	manager	manager	manager
TITLE	title	title	title	title	title	title
EMPLOYEE_NUMBER	employeeenumber	employeeenumber	employeenumber	employeeenumber	employeeenumber	employeeenumber
LDUser.PASSWORD	userpassword	userpassword	userpassword	userpassword	userpassword	userpassword

Note: username* : typically uid, but technically, the attribute designated by the orclCommonNicknameAttribute in the subscriber's oraclecontext products common entry.

D.2 Mapping Role Attributes to LDAP Directories

Table D-2 lists each role attribute in UserProfile.property and its corresponding attribute in different directory servers.

Table D-2 Role Attribute Values in LDAP Directories

Role Attribute	Oracle Internet Directory	Oracle WebLogic Server Embedded LDAP	Microsoft Active Directory	Oracle Directory Server Enterprise Edition	Novell eDirectory	OpenLDAP
DISPLAY_NAME	displayname	-	displayname	displayname	displayname	displayname
MANAGER	-	-	-	-	-	-
NAME	cn	cn	cn	cn	cn	cn
OWNER	owner	owner	-	Owner	-	owner
GUID	orclguid	cn	objectguid	NSuniqueid	guid	entryuuid

D.3 Default Configuration Parameters

This section lists parameters for which the APIs can use default configuration values, and the source of the value in different directory servers.

[Table D-3](#) lists the source for Oracle Internet Directory and Microsoft Active Directory.

Table D-3 Default Values - Oracle Internet Directory and Microsoft Active Directory

Parameter	Oracle Internet Directory	Active Directory
RT_USER_OBJECT_CLASSES	#config	{"user" }
RT_USER_MANDATORY_ATTRS	#schema	#schema
RT_USER_CREATE_BASES	#config	cn=users,<subscriberDN>
RT_USER_SEARCH_BASES	#config	<subscriberDN>
RT_USER_FILTER_OBJECT_CLASSES	#config	{"user"}
RT_USER_SELECTED_CREATE_BASE	#config	cn=users,<subscriberDN>
RT_GROUP_OBJECT_CLASSES	#config	{"group" }
RT_GROUP_MANDATORY_ATTRS	#schema	#schema
RT_GROUP_CREATE_BASES	#config	<subscriberDN>
RT_GROUP_SEARCH_BASES	#config	<subscriberDN>
RT_GROUP_FILTER_OBJECT_CLASSES	#config	{"group"}
RT_GROUP_MEMBER_ATTRS	"uniquemember", "member"	"member"
RT_GROUP_SELECTED_CREATE_BASE	#config	<subscriberDN>
RT_GROUP_GENERIC_SEARCH_BASE	<subscriber-DN>	<subscriberDN>
RT_SEARCH_TYPE	#config	#config
ST_SUBSCRIBER_NAME	#config	NULL

Table D-3 (Cont.) Default Values - Oracle Internet Directory and Microsoft Active

Parameter	Oracle Internet Directory	Active Directory
ST_USER_NAME_ATTR	#config	cn
ST_USER_LOGIN_ATTR	#config	samaccountname
ST_GROUP_NAME_ATTR	#config	cn
ST_MAX_SEARCHFILTER_LENGTH	500	500
ST_BINARY_ATTRIBUTES	Choose a Binary Basic Attribute (BBA) See note below about BBAs.	Binary Basic Attribute (BBA)+ { "objectguid", "unicodepwd" } See note below about BBAs.
ST_LOGGER_NAME	oracle.idm.userrole	oracle.idm.userrole

Notes:

- The Basic Binary Attributes include: {"photo", "personalsignature", "audio", "jpegphoto", "Java Serializeddata", "thumbnailphoto", "thumbnaillogo", "userpassword", "usercertificate", "cacertificate", "authorityrevocationlist", "certificaterevocationlist", "crosscertificatepair", "x500UniqueIdentifier"}
- #config is extracted from the meta information present in the directory
- #schema is extracted from the schema in the directory

Table D-4 lists the source for Oracle Directory Server Enterprise Edition and Novell eDirectory.

Table D-4 Default Values - Oracle Directory Server Enterprise Edition and Novell eDirectory

Parameter	Oracle Directory Server Enterprise Edition	Novell eDirectory
RT_USER_OBJECT_CLASSES	{"inetorgperson", "person", "organizationalperson" }	{"person", "inetorgperson", "organizationalPerson", "ndsloginproperties" }
RT_USER_MANDATORY_ATTRS	#schema	#schema
RT_USER_CREATE_BASES	ou=people,<subscriberDN>	ou=users,<subscriberDN>
RT_USER_SEARCH_BASES	<subscriberDN>	<subscriberDN>
RT_USER_FILTER_OBJECT_CLASSES	{"inetorgperson", "person", "organizationalperson" }	{"person", "inetorgperson", "organizationalPerson", "ndsloginproperties" }
RT_USER_SELECTED_CREATE_BASE	ou=people,<subscriberDN>	ou=users,<subscriberDN>
RT_GROUP_OBJECT_CLASSES	"groupofuniquenames"	{"group" }

Table D–4 (Cont.) Default Values - Oracle Directory Server Enterprise Edition and Novell eDirectory

Parameter	Oracle Directory Server Enterprise Edition	Novell eDirectory
RT_GROUP_MANDATORY_ATTRS	#schema	#schema
RT_GROUP_CREATE_BASES	ou=groups,<subscriberDN>	ou=groups,<subscriberDN>
RT_GROUP_SEARCH_BASES	<subscriberDN>	<subscriberDN>
RT_GROUP_FILTER_OBJECT_CLASSES	{"groupofuniquenames"}	{"group"}
RT_GROUP_MEMBER_ATTRS	"uniquemember"	"member"
RT_GROUP_SELECTED_CREATE_BASE	ou=groups,<subscriberDN>	ou=groups,<subscriberDN>
RT_GROUP_GENERIC_SEARCH_BASE	<subscriber-DN>	<subscriberDN>
RT_SEARCH_TYPE	#config	#config
ST_SUBSCRIBER_NAME	NULL	NULL
ST_USER_NAME_ATTR	uid	cn
ST_USER_LOGIN_ATTR	uid	cn
ST_GROUP_NAME_ATTR	cn	cn
ST_MAX_SEARCHFILTER_LENGTH	500	500
ST_BINARY_ATTRIBUTES	Choose a Binary Basic Attribute (BBA) See note below about BBAs.	Binary Basic Attribute (BBA)+ { "guid" See note below about BBAs.
ST_LOGGER_NAME	oracle.idm.userrole	oracle.idm.userrole

Notes:

- The Basic Binary Attributes include: {"photo", "personalsignature", "audio", "jpegphoto", "JavaSerializeddata", "thumbnailphoto", "thumbnaillogo", "userpassword", "usercertificate", "cacertificate", "authorityrevocationlist", "certificaterevocationlist", "crosscertificatepair", "x500UniqueIdentifier"}
- #config is extracted from the metainformation present in the directory
- #schema is extracted from the schema in the directory

Table [Table D–5](#) lists the parameters for OpenLDAP and Oracle Virtual Directory.

Table D-5 Default Values - OpenLDAP and Oracle Virtual Directory

Parameter	OpenLDAP	Oracle Virtual Directory
RT_USER_OBJECT_CLASSES	{"inetorgperson", "person", "organizationalperson" }	{"inetorgperson"}
RT_USER_MANDATORY_ATTRS	#schema	#schema
RT_USER_CREATE_BASES	ou=people,<subscriberDN>	<subscriberDN>
RT_USER_SEARCH_BASES	<subscriberDN>	<subscriberDN>
RT_USER_FILTER_OBJECT_CLASSES	{"inetorgperson", "person", "organizationalperson" }	{"inetorgperson"}
RT_USER_SELECTED_CREATE_BASE	ou=people,<subscriberDN>	<subscriberDN>
RT_GROUP_OBJECT_CLASSES	"groupofuniquenames"	{"groupofuniquenames"}
RT_GROUP_MANDATORY_ATTRS	#schema	#schema
RT_GROUP_CREATE_BASES	ou=groups,<subscriberDN>	<subscriberDN>
RT_GROUP_SEARCH_BASES	<subscriberDN>	<subscriberDN>
RT_GROUP_FILTER_OBJECT_CLASSES	"groupofuniquenames"	{"groupofuniquenames"}
RT_GROUP_MEMBER_ATTRS	"uniquemember"	"uniquemember"
RT_GROUP_SELECTED_CREATE_BASE	ou=groups,<subscriberDN>	<subscriberDN>
RT_GROUP_GENERIC_SEARCH_BASE	<subscriber-DN>	<subscriberDN>
RT_SEARCH_TYPE	#config	#config
ST_SUBSCRIBER_NAME	NULL	#config (namingcontexts)
ST_USER_NAME_ATTR	uid	cn
ST_USER_LOGIN_ATTR	uid	cn
ST_GROUP_NAME_ATTR	cn	cn
ST_MAX_SEARCHFILTER_LENGTH	500	500
ST_BINARY_ATTRIBUTES	Choose a Binary Basic Attribute (BBA) See note below about BBAs.	Binary Basic Attribute (BBA)+ {"guid"} See note below about BBAs.
ST_LOGGER_NAME	oracle.idm.userrole	oracle.idm.userrole

Notes:

- The Basic Binary Attributes include: {"photo", "personalsignature", "audio", "jpegphoto", "Java Serializeddata", "thumbnailphoto", "thumbnaillogo", "userpassword", "usercertificate", "cacertificate", "authorityrevocationlist", "certificaterevocationlist", "crosscertificatepair", "x500UniqueIdentifier"}
- #config is extracted from the meta information present in the directory
- #schema is extracted from the schema in the directory

Table D–6 lists the parameters for Oracle WebLogic Server LDAP.

Table D–6 Default Values - Oracle WebLogic Server LDAP

Parameter	Oracle WebLogic Server Embedded LDAP
RT_USER_OBJECT_CLASSES	{"inetorgperson", "person", "organizationalperson", "wlsUser"}
RT_USER_MANDATORY_ATTRS	#schema
RT_USER_CREATE_BASES	{"ou=people,<subscriberDN>"}
RT_USER_SEARCH_BASES	{"ou=people,<subscriberDN>"}
RT_USER_FILTER_OBJECT_CLASSES	{"inetorgperson", "wlsUser"}
RT_USER_SELECTED_CREATE_BASE	ou=people,<subscriberDN>
RT_GROUP_OBJECT_CLASSES	{"top", "groupofuniquenames", "groupOfURLs"}
RT_GROUP_MANDATORY_ATTRS	#schema
RT_GROUP_CREATE_BASES	{"ou=groups,<subscriberDN>"}
RT_GROUP_SEARCH_BASES	{"ou=groups,<subscriberDN>"}
RT_GROUP_FILTER_OBJECT_CLASSES	{"top", "groupofuniquenames", "groupOfURLs"}
RT_GROUP_MEMBER_ATTRS	"uniquemember"
RT_GROUP_SELECTED_CREATE_BASE	ou=groups,<subscriberDN>
RT_GROUP_GENERIC_SEARCH_BASE	<subscriberDN>
RT_SEARCH_TYPE	#config
ST_SUBSCRIBER_NAME	#config (namingcontexts)
ST_USER_NAME_ATTR	uid
ST_USER_LOGIN_ATTR	uid
ST_GROUP_NAME_ATTR	cn

Table D–6 (Cont.) Default Values - Oracle WebLogic Server LDAP

Parameter	Oracle WebLogic Server Embedded LDAP
ST_MAX_SEARCHFILTER_LENGTH	500
ST_BINARY_ATTRIBUTES	*(BBA) See note below about BBAs.
ST_LOGGER_NAME	oracle.idm.userrole

D.4 Secure Connections for Microsoft Active Directory

Active Directory requires connections to be SSL-enabled when setting sensitive information like passwords. Therefore, operations like creating a user (which set the password) will not succeed if the connection is not SSL-enabled.

Administration with WLST Scripting and MBean Programming

This appendix describes advanced administrative tasks carried out with WLST scripts and MBean programming, in the following sections:

- [Configuring OPSS Service Provider Instances with a WLST Script](#)
- [Configuring OPSS Services with MBeans](#)
- [Access Restrictions](#)

E.1 Configuring OPSS Service Provider Instances with a WLST Script

If your application uses the User and Role API and must access an authenticator user attribute *different* from the default attribute (which is `cn`), then using the WebLogic Administration Console, you would configure the authenticator to use the desired user attribute. But for the User and Role API to use an attribute different from the default, the authenticator must be, in addition, properly initialized.

The procedure below explains how to use a WLST script to change the authenticator initialization, so that the User and Role API uses the configured user attribute to access data in the configured authenticator.

For details about WebLogic scripting, see *Oracle Fusion Middleware Oracle WebLogic Scripting Tool*.

To add or update custom properties of a service instance, proceed as follows:

1. Create a py script file with the following content:

```
import sys
connect('userName', 'userPass', 't3://localhost:portNumber')
domainRuntime()

val = None
key = None
si = None
for i in range(len(sys.argv)):
    if sys.argv[i] == "-si":
        si = sys.argv[i+1]
    if sys.argv[i] == "-key":
        key = sys.argv[i+1]
    if sys.argv[i] == "-value":
        val = sys.argv[i+1]

on = ObjectName("com.oracle.jps:type=JpsConfig")
sign = ["java.lang.String", "java.lang.String", "java.lang.String"]
```

```
params = [si,key,val]
mbs.invoke(on, "updateServiceInstanceProperty", params, sign)
mbs.invoke(on, "persist", None, None)
```

2. In the produced script, replace *userName*, *userPass*, *localhost*, and *portNumber* by the appropriate strings to connect to the administration server in the domain you are interested. Note that the use of `connect` requires that the server to which you want to connect be up and running when the script is invoked.

Let's assume that the script is saved in the file
 /tmp/updateServiceInstanceProperty.py.

3. Change to the directory \$ORACLE_HOME/common/bin, which should contain the file `wlst.sh`:

```
>cd $ORACLE_HOME/common/bin
```

4. Run the following command:

```
>wlst.sh /tmp/updateServiceInstanceProperty.py -si servInstName -key propKey
-value propValue
```

Where:

- *servInstName* is the name of the service instance provider whose properties are to be modified.
- *propKey* identifies the name of the property to insert or modify.
- *propValue* is the name of the value to add or update.

Any argument containing a space character must be enclosed with double quotes.

Each invocation of the above command modifies the domain configuration file \$DOMAIN_HOME/config/fmwconfig/jps-config.xml by adding or updating a property to the passed instance provider. If the passed key matches the name of an existing property, then that property is updated with the passed value.

5. Restart the Oracle WebLogic server: the changes in configuration are not in effect until the server has been restarted.

Example of Use

Assume that the domain configuration file contains an authenticator named `idstore.ldap`. Then the following invocation:

```
wlst.sh /tmp/updateServiceInstanceProperty.py -si idstore.ldap -key "myPropName"
-value "myValue"
```

adds (or updates) the specified property of that instance provider as illustrated in the following snippet:

```
<serviceInstance provider="idstore.ldap.provider" name="idstore.ldap">
  ...
  <property name="myPropName" value="myValue" />
  ...
</serviceInstance>
```

When the authenticator is initialized with the above configuration, the User and Role API can use the user attribute `mail` to access user information in this authenticator.

E.2 Configuring OPSS Services with MBeans

Oracle Platform Security Services provides a set of JMX-compliant Java EE Beans that are used by Oracle Enterprise Manager Fusion Middleware Control and OPSS security scripts to manage, configure, and monitor Oracle Platform Security Services.

The use of MBeans is recommended in Java EE applications only.

Links to OPSS API javadocs, including the OPSS MBeans API javadoc, are available in [Section H.1, "OPSS API References."](#)

This section addresses the following topics:

- [List of Supported OPSS MBeans](#)
- [Invoking an OPSS MBean](#)
- [Programming with OPSS MBeans](#)

E.2.1 List of Supported OPSS MBeans

[Table E-1](#) lists the supported MBeans, their basic function, and the object name to use in custom WLST scripts or Java SE programs to perform a task:

Table E-1 List of OPSS MBeans

MBean	Function	MBeanServer Connection Name
Jps Configuration	Manages domain configuration data, that is in the file <code>jps-config.xml</code> . This MBean provides the only way to modify configuration data. Update or write operations require server restart to effect changes.	<code>com.oracle.jps:type=JpsConfig</code>
Credential Store	Manages credential data, that is, the store service configured in the default context. Update or write operations do not require server restart to effect changes. All changes are effected immediately. Access is restricted to administrators only.	<code>com.oracle.jps:type=JpsCredentialStore</code>
Global Policy Store	Manages global policies in the policy store configured in the default context. Update or write operations do not require server restart to effect changes. All changes are effected immediately.	<code>com.oracle.jps:type=JpsGlobalPolicyStore</code>
Application Policy Store	Manages application policies in the policy store configured in the default context. Update or write operations do not require server restart to effect changes. All changes are effected immediately.	<code>com.oracle.jps:type=JpsApplicationPolicyStore</code>
Administration Policy Store	Validates whether a user logged into the current JMX context belongs to a particular role. It does not facilitate any configuration modifications.	<code>com.oracle.jps:type=JpsAdminPolicyStore</code>

E.2.2 Invoking an OPSS MBean

There are two basic ways to invoke an OPSS MBean:

- To write a script and run it using the Oracle WebLogic scripting tool; for details, see section Navigating MBeans (WLST Online) in *Oracle Fusion Middleware Oracle WebLogic Scripting Tool*.
- To write a Java program; [Section E.2.3, "Programming with OPSS MBeans"](#) contains a sample program illustrating this approach.

Note: An alternative way to invoke an MBean is using the MBean browser in Fusion Middleware Control. This approach, however, allows only a limited number of operations and it involves composite data creation.

To access this browser, login to Fusion Middleware Control and then proceed as follows:

1. Select the menu item **AdminServer > System MBean Browser**, in the appropriate domain, to display the page **System MBean Browser**.
2. In the pane where the hierarchy is displayed, expand the nodes **Application Defined MBeans**, **com.oracle.jps**, and **Domain: myDomain** (where *myDomain* stands for the name of your domain); this last one has under it one node per OPSS MBean.
3. After expanding any of those nodes, select an item, that is an MBean, and use the tabs **Attributes**, **Operations**, and **Notifications** in the right pane to inspect current attribute values or to invoke methods in the selected MBean.

For example, the Jps Configuration MBean is found at the following location in this hierarchy:

```
Application Defined
MBeans/com.oracle.jps/Domain:myDomain/JpsConfig/JpsConfig
```

For complete details about this browser, see the Fusion Middleware Control online help system.

E.2.3 Programming with OPSS MBeans

The following code sample illustrates how to invoke the Jps Configuration MBean over the WebLogic Server t3 protocol; in this sample, note the following important points:

- It assumes that the following JAR files are in the class path:
 - \$ORACLE_HOME/oracle_common/modules/oracle.jps_11.1.1/jps-api.jar
 - \$ORACLE_HOME/oracle_common/modules/oracle.jps_11.1.1/jps-mbeans.jar
 - \$ORACLE_HOME/oracle_common/modules/oracle.jmx_11.1.1/jmxframework.jar
 - \$ORACLE_HOME/oracle_common/modules/oracle.idm_11.1.1/identitystore.jar
 - \$WEBLOGIC_HOME/server/lib/wljmxclient.jar
- The connection is established by the method `init`.
- Any update operation is followed by a call to `persist`.

```
import java.io.IOException;
import java.net.MalformedURLException;
```

```

import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.List;

import javax.management.InstanceNotFoundException;
import javax.management.MBeanException;
import javax.management.MBeanServerConnection;
import javax.management.MalformedObjectNameException;
import javax.management.ObjectName;
import javax.management.ReflectionException;
import javax.management.openmbean.CompositeData;
import javax.management.remote.JMXConnector;
import javax.management.remote.JMXConnectorFactory;
import javax.management.remote.JMXServiceURL;
import javax.naming.Context;

import oracle.security.jps.mas.mgmt.jmx.credstore.PortableCredential;
import oracle.security.jps.mas.mgmt.jmx.credstore.PortablePasswordCredential;
import oracle.security.jps.mas.mgmt.jmx.policy.PortableApplicationRole;
import oracle.security.jps.mas.mgmt.jmx.policy.PortableCodeSource;
import oracle.security.jps.mas.mgmt.jmx.policy.PortableGrant;
import oracle.security.jps.mas.mgmt.jmx.policy.PortableGrantee;
import oracle.security.jps.mas.mgmt.jmx.policy.PortablePermission;
import oracle.security.jps.mas.mgmt.jmx.policy.PortablePrincipal;
import oracle.security.jps.mas.mgmt.jmx.policy.PortableRoleMember;
import oracle.security.jps.mas.mgmt.jmx.util.JpsJmxConstants;

public class InvokeJpsMbeans {
    private static JMXConnector connector;
    private static MBeanServerConnection wlsMBeanConn;
    private static ObjectName configName;
    private static ObjectName credName;
    private static ObjectName appPolName;
    private static ObjectName gloPolName;
    private static ObjectName adminPolName;

    private final static String STR_NAME =String.class.getName();

    public static void main(String args[]) {
        // Intialize connection and retrieve connection object
        init();

        //Check registration
        if (isRegistered(configName))
            System.out.println("Jps Config MBean is registered");
        if (isRegistered(credName))
            System.out.println("Jps Credential Mbean is registered");
        if (isRegistered(appPolName))
            System.out.println("Jps Application policy Mbean is registered");
        if (isRegistered(gloPolName))
            System.out.println("Jps Global policy Mbean is registered");
        if (isRegistered(adminPolName))
            System.out.println("Jps Admin Policy Mbean is registered");

        //invoke MBeans
        invokeConfigMBeanMethods();
        invokeCredentialMBeanMethods();
        invokeApplicationPolicyMBeanMethods();
        invokeGlobalPolicyMBeanMethods();
    }
}

```

```

        invokeAdminPolicyMBeanMethods();
    }

    private static void invokeConfigMBeanMethods() {
        String KEY = "myKey";
        String VALUE = "myValue";
        String strVal;
        try {
            strVal = (String) wlsMBeanConn.invoke(configName, "updateProperty",
                new Object[] { KEY, VALUE },
                new String[] { STR_NAME, STR_NAME });
            wlsMBeanConn.invoke(configName, "persist", null, null);

            strVal = (String) wlsMBeanConn.invoke(configName, "getProperty",
                new Object[] { KEY }, new String[] { STR_NAME });
            System.out.println("Updated the property: " + strVal.equals(strVal));

            strVal = (String) wlsMBeanConn.invoke(configName, "removeProperty",
                new Object[] { KEY }, new String[] { STR_NAME });
            wlsMBeanConn.invoke(configName, "persist", null, null);
        } catch (InstanceNotFoundException e) {
            // auto-generated catch block
            e.printStackTrace();
        } catch (MBeanException e) {
            // auto-generated catch block
            e.printStackTrace();
        } catch (ReflectionException e) {
            // auto-generated catch block
            e.printStackTrace();
        } catch (IOException e) {
            // auto-generated catch block
            e.printStackTrace();
        }
    }

    private static void invokeCredentialMBeanMethods() {

        String USER = "jdoe";
        String PASSWORD = "welcome1";
        String ALIAS = "mapName";
        String KEY = "keyValue";

        PortableCredential cred = new PortablePasswordCredential(USER,
            PASSWORD.toCharArray());

        try {
            //seed a password credential
            wlsMBeanConn.invoke(credName, "setPortableCredential", new Object[] {
                ALIAS, KEY, cred.toCompositeData(null) }, new String[] { STR_NAME, STR_NAME,
                CompositeData.class.getName() });
            boolean bContainsMap = (Boolean) wlsMBeanConn.invoke(credName,
                "containsMap", new Object[] { ALIAS }, new String[] { STR_NAME });
            System.out.println("Credstore contains map: " + ALIAS + " - "
                + bContainsMap);

            boolean bContainsCred = (Boolean) wlsMBeanConn.invoke(credName,
                "containsCredential", new Object[] { ALIAS, KEY }, new String[] { STR_NAME, STR_
                NAME });
            System.out.println("Contains Credential; " + bContainsCred);
        }
    }

```



```

        CompositeData cd = (CompositeData) wlsMBeanConn.invoke(credName,
"getPortableCredential", new Object[] { ALIAS, KEY }, new String[] { STR_NAME,
STR_NAME });
        cred = PortableCredential.from(cd);

        PortablePasswordCredential pc = (PortablePasswordCredential) cred;

        System.out.println("User name should be " + USER + " Retrieved - " +
pc.getName());
        System.out.println("Password should be " + PASSWORD + "retrieved - " +
new String(pc.getPassword()));

        //delete entire map
        wlsMBeanConn.invoke(credName, "deleteCredentialMap", new Object[] {ALIAS},
new String[] {STR_NAME} );

        } catch (InstanceNotFoundException e) {
            // auto-generated catch block
            e.printStackTrace();
        } catch (MBeanException e) {
            // auto-generated catch block
            e.printStackTrace();
        } catch (ReflectionException e) {
            // auto-generated catch block
            e.printStackTrace();
        } catch (IOException e) {
            // auto-generated catch block
            e.printStackTrace();
        }
    }

private static void invokeApplicationPolicyMBeanMethods() {
    //add grants to approles

    //first create application policy
    String TESTGET_APP_ROLES_MEMBERS = "testgetAppRolesMembers";
    try {
        wlsMBeanConn.invoke(appPolName, "deleteApplicationPolicy", new
Object[] { TESTGET_APP_ROLES_MEMBERS }, new String[] { STR_NAME });
    } catch (Exception e) {
        System.out.println("IGNORE: App " + TESTGET_APP_ROLES_MEMBERS + "
might not exist");
    }
    try {
        wlsMBeanConn.invoke(appPolName, "createApplicationPolicy", new
Object[] { TESTGET_APP_ROLES_MEMBERS }, new String[] { STR_NAME });
        // add remove members to applicaiton roles
        // Create App Role here
        String APP_ROLE_NAME = "ravenclaw_house";
        wlsMBeanConn.invoke(appPolName, "createApplicationRole", new Object[]
{ TESTGET_APP_ROLES_MEMBERS, APP_ROLE_NAME, null, null, null }, new String[] {
STR_NAME, STR_NAME, STR_NAME, STR_NAME, STR_NAME });

        CompositeData cd = (CompositeData) wlsMBeanConn.invoke(appPolName,
"getApplicationRole", new Object[] { TESTGET_APP_ROLES_MEMBERS, APP_ROLE_NAME },
new String[] { STR_NAME, STR_NAME });
        PortableApplicationRole appRole = PortableApplicationRole.from(cd);

        //Add custom principal here

```

```

        PortableRoleMember prm_custom = new
PortableRoleMember("My.Custom.Principal", "CustomPrincipal", null, null, null);

        CompositeData[] arrCompData = { prm_custom.toCompositeData(null) };
        cd = (CompositeData) wlsMBeanConn.invoke(appPolName,
"addMembersToApplicationRole", new Object[] { TESTGET_APP_ROLES_MEMBERS,
appRole.toCompositeData(null), arrCompData }, new String[] { STR_NAME,
CompositeData.class.getName(), CompositeData[].class.getName() });

        // Chk if member got added
        CompositeData[] arrCD = (CompositeData[])
wlsMBeanConn.invoke(appPolName, "getMembersForApplicationRole", new Object[] {
TESTGET_APP_ROLES_MEMBERS, appRole.toCompositeData(null) }, new String[] { STR_
NAME, CompositeData.class.getName() });
        PortableRoleMember[] actRM = getRMArrayFromCDArray(arrCD);
        PortableRoleMember[] expRM = { prm_custom};
        chkRoleMemberArrays(actRM, expRM);

        cd = (CompositeData) wlsMBeanConn.invoke(appPolName,
"removeMembersFromApplicationRole", new Object[] { TESTGET_APP_ROLES_MEMBERS,
appRole.toCompositeData(null), arrCompData }, new String[] { STR_NAME,
CompositeData.class.getName(), CompositeData[].class.getName() });

        // Chk if member got removed
        arrCD = (CompositeData[]) wlsMBeanConn.invoke(appPolName,
"getMembersForApplicationRole", new Object[] { TESTGET_APP_ROLES_MEMBERS,
appRole.toCompositeData(null) }, new String[] { STR_NAME,
CompositeData.class.getName() });
        System.out.println("length should be zero : " + arrCD.length);

        // Remove the App Role
        wlsMBeanConn.invoke(appPolName, "removeApplicationRole", new Object[]
{ TESTGET_APP_ROLES_MEMBERS, APP_ROLE_NAME }, new String[] { STR_NAME, STR_NAME
});
        wlsMBeanConn.invoke(appPolName, "deleteApplicationPolicy", new
Object[] { TESTGET_APP_ROLES_MEMBERS }, new String[] { STR_NAME });

    } catch (InstanceNotFoundException e) {
        // auto-generated catch block
        e.printStackTrace();
    } catch (MBeanException e) {
        // auto-generated catch block
        e.printStackTrace();
    } catch (ReflectionException e) {
        // auto-generated catch block
        e.printStackTrace();
    } catch (IOException e) {
        // auto-generated catch block
        e.printStackTrace();
    }
}

private static PortableRoleMember[] getRMArrayFromCDArray(CompositeData[]
arrCD) {
    PortableRoleMember[] actRM = new PortableRoleMember[arrCD.length];
    int idx = 0;
    for (CompositeData cdRM : arrCD) {
        actRM[idx++] = PortableRoleMember.from(cdRM);
    }
    return actRM;
}

```

```

    }

    private static void chkRoleMemberArrays(PortableRoleMember[] arrExpectedRM,
PortableRoleMember[] arrActRM) {

        List < PortableRoleMember > lstExpRM = new ArrayList < PortableRoleMember
>(Arrays.asList(arrExpectedRM));
        List < PortableRoleMember > lstActRM = new ArrayList < PortableRoleMember
>(Arrays.asList(arrActRM));

        for (PortableRoleMember actRM : lstActRM) {
            for (int idx = 0; idx < lstExpRM.size(); idx++) {
                PortableRoleMember expRM = (PortableRoleMember) lstExpRM.get(idx);
                if (expRM.equals(actRM)) {
                    lstExpRM.remove(idx);
                    break;
                }
            }
        }
        System.out.println("List should be empty - " + lstExpRM.size());
    }

    private static void invokeAdminPolicyMBeanMethods() {
        //Connection is established as weblogic user, who by OOTB gets all
permissions
        Boolean bool;
        try {
            bool = (Boolean) wlsMBeanConn.invoke(adminPolName,"checkRole",new
Object[]{"Admin"}, new String[]{STR_NAME});
            System.out.println("Weblogic has Admin role: " + bool);
            bool = (Boolean) wlsMBeanConn.invoke(adminPolName,"checkRole",new
Object[] {"Configurator"}, new String[]{STR_NAME});
            System.out.println("Weblogic has Configurator role: " + bool);
            bool = (Boolean) wlsMBeanConn.invoke(adminPolName,"checkRole", new
Object[]{new String[] {"Operator", "Admin", "Configurator"}},
                new String[]{String[].class.getName()});
            System.out.println("Weblogic has Admin,Operator,Configurator role: "
+ bool);
        } catch (InstanceNotFoundException e) {
            // auto-generated catch block
            e.printStackTrace();
        } catch (MBeanException e) {
            // auto-generated catch block
            e.printStackTrace();
        } catch (ReflectionException e) {
            // auto-generated catch block
            e.printStackTrace();
        } catch (IOException e) {
            // auto-generated catch block
            e.printStackTrace();
        }
    }

    private static void invokeGlobalPolicyMBeanMethods() {
        // lets create a grant in system policy
        PortablePrincipal CUSTOM_JDOE = new
PortablePrincipal("oracle.security.jps.internal.core.principals.CustomXmlUserImpl"
, "jdoe", PortablePrincipal.PrincipalType.CUSTOM);
        PortablePrincipal CUSTOM_APP_ADMINS = new
PortablePrincipal("oracle.security.jps.internal.core.principals.CustomXmlEnterpris

```

```

eRoleImpl", "oc4j-app-administrators", PortablePrincipal.PrincipalType.CUSTOM);
    PortablePrincipal[] arrPrincs = {CUSTOM_JDOE, CUSTOM_APP_ADMINS};
    //code source URL
    String URL = "http://www.oracle.com/as/jps-api.jar";
    PortableCodeSource pcs = new PortableCodeSource(URL);
    PortableGrantee pge = new PortableGrantee(arrPrincs, pcs);
    PortablePermission CSF_PERM = new
PortablePermission("oracle.security.jps.service.credstore.CredentialAccessPermissi
on", "context=SYSTEM,mapName=MY_MAP,keyName=MY_KEY", "read");
    PortablePermission[] arrPerms = {CSF_PERM};
    PortableGrant grnt = new PortableGrant(pge, arrPerms);
    CompositeData[] arrCompData = { grnt.toCompositeData(null) };
    try {
        System.out.println("Creating System Policy grant");
        wlsMBeanConn.invoke(gloPolName, "grantToSystemPolicy", new Object[] {
arrCompData }, new String[] { CompositeData[].class.getName() });
        System.out.println("Deleting the created grant");
        wlsMBeanConn.invoke(gloPolName, "revokeFromSystemPolicy", new Object[]
{ arrCompData }, new String[] { CompositeData[].class.getName() });

        } catch (InstanceNotFoundException e) {
            // auto-generated catch block
            e.printStackTrace();
        } catch (MBeanException e) {
            // auto-generated catch block
            e.printStackTrace();
        } catch (ReflectionException e) {
            // auto-generated catch block
            e.printStackTrace();
        } catch (IOException e) {
            // auto-generated catch block
            e.printStackTrace();
        }
    }

private static boolean isRegistered(ObjectName name) {
    try {
        return wlsMBeanConn.isRegistered(name);
    } catch (IOException e) {
        // auto-generated catch block
        e.printStackTrace();
    }
    return false;
}

private static void init() {
    String protocol = "t3";
    String jndi_root = "/jndi/";
    String wlserver = "myWLServer";
    String host = "myHost.com";
    int port = 7001;
    String adminUsername = "myAdminName";
    String adminPassword = "myAdminPassw";
    JMXServiceURL url;
    try {
        url = new JMXServiceURL(protocol,host,port,jndi_root+wlserver);
        HashMap<String, Object> env = new HashMap<String, Object>();
        env.put(Context.SECURITY_PRINCIPAL, adminUsername);
        env.put(Context.SECURITY_CREDENTIALS, adminPassword);
        env.put(JMXConnectorFactory.PROTOCOL_PROVIDER_PACKAGES,

```

```

        "weblogic.management.remote");
connector = JMXConnectorFactory.connect(url, env);
wlsMBeanConn = connector.getMBeanServerConnection();

//create object names
// the next string is set to com.oracle.jps:type=JpsConfig
configName = new
    ObjectName(JpsJmxConstants.MBEAN_JPS_CONFIG_FUNCTIONAL);
// the next string is set to com.oracle.jps:type=JpsApplicationPolicyStore
appPolName = new
    ObjectName(JpsJmxConstants.MBEAN_JPS_APPLICATION_POLICY_STORE);
// the next string is set to com.oracle.jps:type=JpsGlobalPolicyStore
gloPolName = new
    ObjectName(JpsJmxConstants.MBEAN_JPS_GLOBAL_POLICY_STORE);
// the next string is set to com.oracle.jps:type=JpsAdminPolicyStore
adminPolName = new
    ObjectName(JpsJmxConstants.MBEAN_JPS_ADMIN_POLICY_STORE);
// the next string is set to com.oracle.jps:type=JpsCredentialStore
credName = new ObjectName(JpsJmxConstants.MBEAN_JPS_CREDENTIAL_STORE);
} catch (MalformedURLException e) {
    // take proper action
    e.printStackTrace();
} catch (IOException e) {
    // take proper action
    e.printStackTrace();
} catch (MalformedObjectNameException e) {
    // auto-generated catch block
    e.printStackTrace();
}
}
}

```

For further details about programmatic configuration of services, see part [Part V](#), "[Developing with Oracle Platform Security Services APIs](#)"

E.3 Access Restrictions

The information in this section is not restricted to OPPS MBeans but applies, more generally, to Oracle Fusion Middleware MBeans.

The security access to MBeans is based on logical roles rather than on security permissions. MBeans are annotated using role-based constraints that are enforced at run time by the JMX Framework.

This section illustrates the use of some annotations, describes what they mean, lists the particular access restrictions, and explains the mapping of logical roles to Oracle WebLogic Server enterprise groups.

E.3.1 Annotation Examples

The following code snippet illustrates the use of some enterprise group annotations (in bold text) in an MBean interface:

```

@Description(resourceKey = "demo.ScreenCustomizerRuntimeMBean.description",
             resourceBundleBaseName = "demo.runtime.Messages")
@ImmutableInfo("true")
@Since("1.1")
public interface ScreenCustomizerRuntimeMXBean {
    @Description(resourceKey = "demo.ScreenCustomizerRuntimeMBean.Active",
                 resourceBundleBaseName = "demo.runtime.Messages")

```

```

@AttributeGetterRequiredGlobalSecurityRole(GlobalSecurityRole.Operator)
    public boolean isActive();
@AttributeSetterRequiredGlobalSecurityRole(GlobalSecurityRole.Admin)
    public void setActive(boolean val);

    @Description(resourceKey =
        "demo.ScreenCustomizerRuntimeMBean.ActiveVirtualScreenId",
        resourceBundleBaseName = "demo.runtime.Messages")
    @DefaultValue("0")
    @LegalValues( {"0", "2", "4", "6", "8" })
    @RequireRestart(ConfigUptakePolicy.ApplicationRestart)
@OperationRequiredGlobalSecurityRole(GlobalSecurityRole.Admin)
    public void setActiveVirtualScreenId(int id) throws IllegalArgumentException;
    ...
}

```

In the above code sample, the annotation:

- **@AttributeGetterRequiredGlobalSecurityRole** specifies that a user must belong to the role `Operator` to access the get method `isActive`.
- **@AttributeSetterRequiredGlobalSecurityRole** specifies that a user must belong to the role `Admin` to access the set method `setActive`.
- **@OperationRequiredGlobalSecurityRole** specifies that a user must belong to the role `Admin` to access the MBean method `setActiveVirtualScreenId`.

Note that all three annotations above apply just to a specific item in the interface.

The following code snippet illustrates the use of another annotation (in bold text) with a different scope:

```

    @Description(resourceKey = "demo.ScreenCustomizerRuntimeMBean.description",
        resourceBundleBaseName = "demo.runtime.Messages")
    @ImmutableInfo("true")
    @Since("1.1")
@MBeanRequiredGlobalSecurityRole(GlobalSecurityRole.Admin)
    public interface ScreenCustomizerRuntimeMXBean { ... }

```

In the above code sample, the annotation **@MBeanRequiredGlobalSecurityRole** specifies that a user must belong to the role `Admin` to access *any* operation or attribute of the MBean, that is, its scope is the entire MBean. Annotations with method or attribute scope override annotations that apply to the entire MBean.

The enumeration `GlobalSecurityRole` defines the set of global, logical roles that are mapped to actual roles in the environment before performing security checks. This enumeration includes the value `NONE` to indicate that any user has read and write access to the annotated operation or attribute.

For details, see the `oracle.jmx.framework` Javadoc documentation.

E.3.2 Mapping of Logical Roles to WebLogic Roles

Table E-2 shows the mapping of logical roles to enterprise groups.

Table E-2 Mapping of Logical Roles to WebLogic Groups

Logical Role	Default Privileges	WebLogic Group
Admin	Read and write access to all MBeans	Admin
Configurator	Read and write access to configuration MBeans	Admin

Table E-2 (Cont.) Mapping of Logical Roles to WebLogic Groups

Logical Role	Default Privileges	WebLogic Group
Operator	Read access to configuration MBeans; read and write access to all run time MBeans	Operator
Monitor	Read access to all MBeans	Monitor
ApplicationAdmin	Read and write access to all application MBeans	Admin
ApplicationConfigurator	Read and write access to all application MBeans	Admin
ApplicationOperator	Read access to application configuration MBeans; read and write access to application runtime MBeans	Operator
ApplicationMonitor	Read access to all application runtime and configuration MBeans	Monitor

For details about WebLogic roles, see sections Users, Groups, And Security Roles and in *Oracle Fusion Middleware Securing Resources Using Roles and Policies for Oracle WebLogic Server*.

E.3.3 Particular Access Restrictions

By default, all write and update operations require that the user be a member of the Admin or Configurator roles. In addition, operations annotated with the tag `@Impact (value=1)` require the user to be a member of the Admin role, and operations annotated with the tag `@Impact (value=0)` require the user to be a member of the Admin or Operator roles.

Table E-3 describes the roles required to access attributes and operations of Fusion Middleware Control MBeans:

Table E-3 Roles Required per Operation

Operations with impact value	MBean type	Require any of the roles
INFO or attribute getter	System configuration MBean	Monitor, Operator, Configurator, Admin
INFO or attribute getter	Application configuration MBean	Monitor, Operator, Configurator, Admin, ApplicationMonitor, ApplicationOperator, ApplicationConfigurator, ApplicationAdmin
ACTION, ACTION_ INFO, UNKNOWN, or attribute setter	System configuration MBean	Admin, Configurator
ACTION, ACTION_ INFO, UNKNOWN, or attribute setter	Application configuration MBean	Admin, Configurator, ApplicationAdmin, ApplicationConfigurator
INFO or attribute getter	System runtime MBean	Monitor, Operator, Configurator, Admin
INFO or attribute getter	Application runtime MBean	Monitor, Operator, Configurator, Admin, ApplicationMonitor, ApplicationOperator, ApplicationAdmin

Table E-3 (Cont.) Roles Required per Operation

Operations with impact value	MBean type	Require any of the roles
ACTION, ACTION_ INFO, UNKNOWN, or attribute setter	System runtime MBean	Admin, Operator
ACTION, ACTION_ INFO, UNKNOWN, or attribute setter	Application runtime MBean	Admin, Operator, ApplicationAdmin, ApplicationOperator

OPSS System and Configuration Properties

This appendix documents OPSS system properties (set through the switch `-D` at server start) and configuration properties (set with elements `<property>` and `<extendedProperty>` in the configuration file `jps-config.xml`) in the following sections:

- [OPSS System Properties](#)
- [OPSS Configuration Properties](#)

To manage server properties programmatically, use OPSS MBeans. For details and example, see [Section E.2.3, "Programming with OPSS MBeans."](#)

Note: All OPSS *configuration* changes (manual or through `JpsConfiguration` MBean) require server restart to take effect.

OPSS *data* domain changes do not require server restart to take effect. Data changes include modifying an application policy and creating, deleting, or updating a credential.

F.1 OPSS System Properties

A system property that has been introduced or modified is not in effect until the server is restarted. In order to set a system property the administrator must edit the `setDomainEnv.sh` shell script and add the property to the environment variable `EXTRA_JAVA_PROPERTIES` in that script.

[Table F-1](#) lists the Java system properties available with OPSS.

Table F-1 Java System Properties Used by OPSS

Name	Description
<code>java.security.debug=access, failure</code>	<p>Notifies about a permission failure when the method <code>JpsAuth.checkPermission</code> is called inside a <code>Subject.doAs</code> block and the permission check fails.</p> <p>Note that setting <code>jps.auth.debug</code> or <code>jps.auth.debug.verbose</code> is not enough to get a failure notification in this case.</p> <p>Optional.</p>
<code>java.security.policy</code>	Specifies the location of the Java security policy file.
<code>jps.authz</code>	<p>Enables or disables the delegation of calls to JDK API <code>AccessController.checkPermission</code>, which reduces runtime and debugging overhead.</p> <p>Optional.</p> <p>Valid values: <code>NULL</code>, <code>SM</code>, <code>ACC</code>, and <code>DEBUG_NULL</code>.</p> <p>No default value.</p>
<code>jps.auth.debug</code>	<p>Controls server logging output. Default value: <code>FALSE</code>. For details, see Section L.1.2.1, "<code>jps.auth.debug</code>." See also <code>java.security.debug</code>.</p> <p>Optional.</p>
<code>jps.auth.debug.verbose</code>	<p>Controls server logging output. Default value: <code>FALSE</code>. For details, see Section L.1.2.2, "<code>jps.auth.debug.verbose</code>." See also <code>java.security.debug</code>.</p> <p>Optional.</p>
<code>jps.combiner.optimize</code>	<p>Enables or disables the caching of a subject's protection domain.</p> <p>Optional.</p> <p>Valid values: <code>TRUE</code>, <code>FALSE</code>.</p> <p>Default value: <code>FALSE</code>.</p>
<code>jps.combiner.optimize.lazyeval</code>	<p>Enables or disables the evaluation of a subject's protection domain when a check permission is triggered.</p> <p>Optional.</p> <p>Valid values: <code>TRUE</code>, <code>FALSE</code>.</p> <p>Default value: <code>FALSE</code>.</p>
<code>jps.deployment.handler.disabled</code>	<p>Enables or disables the migration of policies and credentials for applications deployed in a WebLogic Server. Valid only for the WebLogic Server.</p> <p>Set to <code>TRUE</code> to disable the migration of application policies and credentials for all applications deployed in the server regardless of the particular application settings in the application file <code>weblogic-application.xml</code>.</p> <p>Optional.</p> <p>Valid values: <code>TRUE</code>, <code>FALSE</code>.</p> <p>Default value: <code>FALSE</code>.</p>

Table F-1 (Cont.) Java System Properties Used by OPSS

Name	Description
jps.policystore.hybrid.mode	<p>Enables or disables the hybrid mode.</p> <p>The hybrid mode is used to facilitate the transition from the Sun <code>java.security.Policy</code> to the OPSS Java <code>PolicyProvider</code>. When the hybrid mode is enabled, the OPSS Java Policy Provider reads from both files, <code>java.policy</code> and <code>system-jazn-data.xml</code>.</p> <p>Optional.</p> <p>Valid values: <code>TRUE</code>, <code>FALSE</code>.</p> <p>Default value: <code>TRUE</code>.</p>
jps.subject.cache.ttl	<p>Specifies the number of seconds after which group membership changes are in effect.</p> <p>This value must be kept in sych with the value of the WebLogic authenticator <code>Group Hierarchy Cache</code>. If this last parameter value is changed, then <code>jps.subject.cache.ttl</code> must be reset to match the new <code>Group Hierarchy Cache</code> value.</p> <p>Optional.</p> <p>Valid values: any positive interger.</p> <p>Default value: <code>60000</code></p>
oracle.security.jps.config	<p>Specifies the path to the domain configuration files <code>jps-config.xml</code> or <code>jps-config-jse.xml</code>. Paths specifications in those files can be absolute or relative to the location of the configuration file.</p> <p>Required.</p> <p>No default value.</p>
oracle.deployed.app.dir	<p>Specifies the path to the directory of a code source URL.</p> <p>Optional.</p> <p>No default value.</p> <p>For an example of use, see <url>.</p>
oracle.deployed.app.ext	<p>Specifies the extension of code source URL.</p> <p>Optional.</p> <p>No default value.</p> <p>For an example of use, see <url>.</p>
oracle.security.jps.log.for.appl e.substring	<p>Logs the name of an application role that contains a specified substring; if the substring to match is unspecified, it logs all application role names.</p> <p>Optional.</p> <p>No default value.</p> <p>For an example of use and further details, see Section L.1.2.3, "Debugging the Authorization Process."</p>
oracle.security.jps.log.for.permeffect	<p>Logs a grant that was granted or denied according to a specified value; if the value is unspecified, it logs all grants (regardless whether they were granted or denied).</p> <p>Optional.</p> <p>No default value.</p> <p>For an example of use and further details, see Section L.1.2.3, "Debugging the Authorization Process."</p>

Table F-1 (Cont.) Java System Properties Used by OPSS

Name	Description
oracle.security.jps.log.for.permclassname	<p>Logs the name of the permission class that matches exactly a specified name; if the name to match is unspecified, it logs all permission class names.</p> <p>Optional.</p> <p>No default value.</p> <p>For an example of use and further details, see Section L.1.2.3, "Debugging the Authorization Process."</p>
oracle.security.jps.log.for.permtarget.substring	<p>Logs the name of a permission target that contains a specified substring; if the substring to match is unspecified, it logs all permission targets.</p> <p>Optional.</p> <p>No default value.</p> <p>For an example of use and further details, see Section L.1.2.3, "Debugging the Authorization Process."</p>
oracle.security.jps.log.for.enterprise.principalname	<p>Logs the name of the principal (enterprise user or enterprise role) that matches exactly a specified name; if the name to match is unspecified, it logs all principal names.</p> <p>Optional.</p> <p>No default value.</p> <p>For an example of use and further details, see Section L.1.2.3, "Debugging the Authorization Process."</p>

F.2 OPSS Configuration Properties

This section describes the properties of various instances in the following sections:

- [Policy Store Properties](#)
- [Credential Store Properties](#)
- [LDAP Identity Store Properties](#)
- [Properties Common to All LDAP-Based Instances](#)
- [Anonymous and Authenticated Roles Properties](#)
- [Trust Service Properties](#)
- [Audit Service Properties](#)
- [Keystore Service Properties](#)

F.2.1 Policy Store Properties

The policy store properties are described in the following sections:

- [Policy Store Configuration](#)
- [Runtime Policy Store Configuration](#)

F.2.1.1 Policy Store Configuration

The policy store provider class that can be used with LDAP- or DB-based instances is the following:

```
oracle.security.jps.internal.policystore.ldap.LdapPolicyStoreProvider
```

Table F–2 describes the properties of policy store instances. The properties are listed in three blocks according to the kind of application they can be used in.

Table F–2 Policy Store Properties

Name	Description
The following properties are valid in both Java EE and Java SE applications	
<code>bootstrap.security.principal.key</code>	<p>The key for the password credentials to access the LDAP policy store, stored in the CSF store.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Applies to LDAP and DB stores.</p> <p>Required.</p> <p>No default value.</p> <p>The out-of-the-box value is <code>bootstrap</code>.</p>
<code>bootstrap.security.principal.map</code>	<p>The map for the password credentials to access the LDAP policy store, stored in the CSF store.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Applies to LDAP and DB stores.</p> <p>Required.</p> <p>Default value: <code>BOOTSTRAP_JPS</code>.</p>
<code>oracle.security.jps.farm.name</code>	<p>The RDN format of the domain node in the LDAP policy store.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Applies to LDAP and DB stores.</p> <p>Required.</p> <p>No default value.</p>
<code>oracle.security.jps.ldap.root.name</code>	<p>The RDN format of the root node in the LDAP policy store.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Applies to LDAP and DB stores.</p> <p>Required.</p> <p>No default value.</p>
<code>ldap.url</code>	<p>The URL of the LDAP policy store, with the format <code>ldap://host:port</code>.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Applies only to LDAP stores.</p> <p>Required.</p> <p>No default value.</p>
<code>policystore.type</code>	<p>The type of the LDAP policy store.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Applies to LDAP and DB stores.</p> <p>Required.</p> <p>No default value.</p> <p>Value examples: <code>OID</code>, <code>DB_ORACLE</code>.</p>

Table F-2 (Cont.) Policy Store Properties

Name	Description
oracle.security.jps.policystore.re sourcetypeenforcementmode	<p>Controls the throwing of exceptions if any of the following checks fail:</p> <ul style="list-style-type: none"> ▪ Verify that if two resource types share the same permission class, that permission must be either <code>ResourcePermission</code> or extend <code>AbstractTypedPermission</code>, and this last resource type cannot be created. ▪ Verify that all permissions have resource types defined, and that the resource matcher permission class and the permission being granted match. <p>If set to <code>Strict</code>, when any of the above checks fail, the system throws an exception and the operation is aborted.</p> <p>If set to <code>Lenient</code>, when any of the above checks fail, the system does not throw any exceptions, the operation continues without disruption, and any discrepancies encountered are logged in the log files.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Applies to LDAP and DB stores.</p> <p>Optional.</p> <p>Default value: <code>Lenient</code></p> <p>Valid values: <code>Strict</code>, <code>Lenient</code>.</p>
jps.change.notifier.file.delay	<p>Indicates the frequency, in milliseconds, at which the system checks the domain files <code>system-jazn-data.xml</code> and <code>cwallet.sso</code> for changes. (milliseconds).</p> <p>In production environments, it is recommended a frequency of about 10 min. (600000 milliseconds). In development environments, it is recommended a frequency of about 3 min. (180000 milliseconds).</p> <p>Default value: 1000</p>
The following properties are valid in Java EE applications only	
datasource.jndi.name	<p>The JNDI name of the JDBC data source instance.</p> <p>Valid in only Java EE applications.</p> <p>Applies to only DB stores.</p> <p>Required.</p> <p>No default value.</p>
failover.retry.times	<p>The number of retry attempts.</p> <p>Valid in only Java EE applications.</p> <p>Applies to only DB stores.</p> <p>Optional.</p> <p>Default value: 3</p>
failover.retry.interval	<p>The number of seconds between retry attempts.</p> <p>Valid in only Java EE applications.</p> <p>Applies to only DB stores.</p> <p>Optional.</p> <p>Default value: 15</p>
The following properties are valid in Java SE applications only	

Table F-2 (Cont.) Policy Store Properties

Name	Description
security.principal	<p>The clear text name of the principal to use instead of the user name specified in the bootstrap. Not recommended.</p> <p>Valid in only Java SE applications.</p> <p>Applies to LDAP and DB stores.</p> <p>Optional.</p> <p>No default value.</p>
security.credential	<p>The clear text password for the security principal to use instead of the password specified in the bootstrap. Not recommended.</p> <p>Valid in only Java SE applications.</p> <p>Applies to LDAP and DB stores.</p> <p>Optional.</p> <p>No default value.</p>
jdbc.driver	<p>The JDBC driver.</p> <p>Valid in only Java SE applications.</p> <p>Applies to only DB stores.</p> <p>Required.</p> <p>No default value.</p> <p>Value example: oracle.jdbc.driver.OracleDriver</p>
jdbc.url	<p>The URL of the JBDC.</p> <p>Valid in only Java SE applications.</p> <p>Applies to only DB stores.</p> <p>Required.</p> <p>No default value.</p> <p>Value example: jdbc:oracle:thin:@xxx27.com:1345:asi102cn</p>
eclipselink.jdbc.read-connections.min	<p>The minimum number of connections allowed in the JDBC read connection pool.</p> <p>Valid in only Java SE applications.</p> <p>Applies to only DB stores.</p> <p>Optional.</p> <p>Default value: 5</p>
eclipselink.jdbc.read-connections.max	<p>The maximum number of connections allowed in the JDBC read connection pool.</p> <p>Valid in only Java SE applications.</p> <p>Applies to only DB stores.</p> <p>Optional.</p> <p>Default value: 20</p>

Example 1

The following fragment illustrates the configuration of an LDAP-based policy store instance for a Java EE application:

```
<serviceInstance provider="ldap.policystore.provider" name="policystore.ldap">
  <property value="OID" name="policystore.type"/>
  <property value="bootstrap" name="bootstrap.security.principal.key"/>
</serviceInstance>
```

```

    <property value="cn=wls-jrfServer" name="oracle.security.jps.farm.name"/>
    <property value="cn=jpsTestNode" name="oracle.security.jps.ldap.root.name"/>
    <property value="ldap://stadk06.us.oracle.com:3060" name="ldap.url"/>
    <property value="STATIC"
name="oracle.security.jps.policystore.rolemember.cache.type"/>
    <property value="FIFO"
name="oracle.security.jps.policystore.rolemember.cache.strategy"/>
    <property value="1000"
name="oracle.security.jps.policystore.rolemember.cache.size"/>
    <property value="true"
name="oracle.security.jps.policystore.policy.lazy.load.enable"/>
    <property value="PERMISSION_FIFO"
name="oracle.security.jps.policystore.policy.cache.strategy"/>
    <property value="1000"
name="oracle.security.jps.policystore.policy.cache.size"/>
    <property value="true"
name="oracle.security.jps.policystore.refresh.enable"/>
    <property value="43200000"
name="oracle.security.jps.policystore.refresh.purge.timeout"/>
    <property value="600000"
name="oracle.security.jps.ldap.policystore.refresh.interval"/>
</serviceInstance>

```

Example 2

The following fragment illustrates the configuration of an LDAP-based policy store instance for a Java SE application:

```

<serviceInstance name="policystore.oid" provider="policy.oid">
  <property value="OID" name="policystore.type"/>
  <property value="bootstrap" name="bootstrap.security.principal.key"/>
  <property name="ldap.url" value="ldap://sttt:3060"/>
  <property name="oracle.security.jps.ldap.root.name" value="cn=jpsNode"/>
  <property name="oracle.security.jps.farm.name" value="cn=domain1"/>
</serviceInstance>

```

For additional configurations samples for Java SE applications, see [Section 23.1.2, "Configuring LDAP-Based Policy and Credential Stores."](#)

Example 3

The following fragment illustrates the configuration of DB-based stores (including an instance of a runtime service provider) for a Java EE application:

```

<jpsConfig>
...
  <propertySets>
    <!-- property set props.db.1 common to all DB services -->
    <propertySet name="props.db.1">
      <property name="datasource.jndi.name" value="opssds"/>
      <property value="cn=farm" name="oracle.security.jps.farm.name"/>
      <property value="cn=jpsroot" name="oracle.security.jps.ldap.root.name"/>
      <property value="dsrc_lookup_key"
        name="bootstrap.security.principal.key"/>
      <property value="credential_map" name="bootstrap.security.principal.map"/>
    </propertySet>
  </propertySets>

  <serviceProviders>
    <serviceProvider
      class="oracle.security.jps.internal.policystore.ldap.LdapPolicyStoreProvider"

```



```

    type="POLICY_STORE" name="rdbms.policystore.provider" >
      <description>RDBMS based PolicyStore provider</description>
    </serviceProvider>

    <serviceProvider type="KEY_STORE" name="keystore.provider"
      class="oracle.security.jps.internal.keystore.KeyStoreProvider">
      <description>PKI Based Keystore Provider</description>
      <property name="provider.property.name" value="owsm"/>
    </serviceProvider>

    <serviceProvider name="pdp.service.provider" type="PDP"
      class="oracle.security.jps.az.internal.runtime.provider.PDPServiceProvider">
      <description>OPSS Runtime Service provider</description>
    </serviceProvider>
  </serviceProviders>

  <serviceInstances>
    <serviceInstance name="policystore.rdbms"
      provider="rdbms.policystore.provider">
      <property value="DB_ORACLE" name="policystore.type"/>
      <propertySetRef ref = "props.db.1"/>
      <property name="session_expiration_sec" value="60"/>
      <property name="failover.retry.times" value="5"/>
    </serviceInstance>

    <serviceInstance name="credstore.rdbms" provider="rdbms.credstore.provider">
      <propertySetRef ref = "props.db.1"/>
    </serviceInstance>

    <serviceInstance name="keystore.rdbms" provider="rdbms.keystore.provider">
      <propertySetRef ref = "props.db.1"/>
      <property name="keystore.provider.type" value="db"/>
    </serviceInstance>

    <serviceInstance name="pdp.service" provider="pdp.service.provider">
      <property name="sm_configuration_name" value="permissionSm"/>
      <property name="work_folder" value="../../tempdir/permissionSm-work"/>
      <property name="authorization_cache_enabled" value="true"/>
      <property name="role_cache_enabled" value="true"/>
      <property name="session_eviction_capacity" value="500"/>
      <property name="session_eviction_percentage" value="10"/>
      <property name="session_expiration_sec" value="60"/>
      <property name="failover.retry.times" value="5"/>
      <property name="failover.retry.interval" value="20"/>
      <property name="oracle.security.jps.policystore.purge.timeout",
        value="30000"/>
      <propertySetRef ref = "props.db.1"/>
    </serviceInstance>
  </serviceInstances>

  <jpsContexts default="default">
    <jpsContext name="default">
      <serviceInstanceRef ref="pdp.service"/>
      <serviceInstanceRef ref="policystore.rdbms"/>
      <serviceInstanceRef ref="credstore.rdbms"/>
      <serviceInstanceRef ref="keystore.rdbms"/>
    </jpsContext>
  </jpsContexts>

  ...
</jpsConfig>

```

Example 4

The following fragment illustrates the configuration of a DB-based policy store instance for a Java SE application:

```
<serviceInstance name="policystore.rdbms" provider="policy.rdbms">
  <property name="policystore.type" value="DB_ORACLE"/>
  <property name="jdbc.url" value="jdbc:oracle:thin:@sc.us.oracle.com:1722:orcl"/>
  <property name="jdbc.driver" value="oracle.jdbc.driver.OracleDriver"/>
  <property name="bootstrap.security.principal.key" value="bootstrap_
DWgpEJgXwhDIoLYVZ2OWd4R8wOA=" />
  <property name="oracle.security.jps.ldap.root.name" value="cn=jpsTestNode"/>
  <property name="oracle.security.jps.farm.name" value="cn=view_steph.atz"/>
</serviceInstance>
```

For additional configurations samples for Java SE applications, see [Section 23.1.3, "Configuring DB-Based OPSS Security Stores."](#)

F.2.1.2 Runtime Policy Store Configuration

The runtime policy store provider class that can be used with LDAP- or DB-based instances is the following:

```
oracle.seurity.jps.az.internal.runtime.provider.PDPServiceProvider
```

[Table F-3](#) lists the runtime properties of policy store instances.

Table F-3 Runtime Policy Store Properties

Name	Description
oracle.security.jps.policystore.rolemember.cache.type	<p>The type of the role member cache.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Applies to LDAP and DB stores.</p> <p>Optional.</p> <p>Valid values:</p> <ul style="list-style-type: none"> ■ STATIC - Cache objects are statically cached and can be cleaned explicitly only according the applied cache strategy, such as FIFO. The garbage collector does not clean a cache of this type. ■ SOFT - The cleaning of a cache of this type relies on the garbage collector when there is a memory crunch. ■ WEAK - The behavior of a cache of this type is similar to a cache of type SOFT, but the garbage collector cleans it more frequently. <p>Default value: <i>STATIC</i>.</p>

Table F-3 (Cont.) Runtime Policy Store Properties

Name	Description
oracle.security.jps.policystore.rolemember.cache.strategy	<p>The type of strategy used in the role member cache.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Applies to LDAP and DB stores.</p> <p>Optional.</p> <p>Valid values:</p> <ul style="list-style-type: none"> ■ FIFO - The cache implements the first-in-first-out strategy. ■ NONE - All entries in the cache grow until a refresh or reboot occurs; there is no control over the size of the cache; not recommended but typically efficient when the policy footprint is very small. <p>Default value: FIFO.</p>
oracle.security.jps.policystore.rolemember.cache.size	<p>The number of the roles kept in the member cache.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Applies to LDAP and DB stores.</p> <p>Optional.</p> <p>Default value: 1000.</p>
oracle.security.jps.policystore.policy.lazy.load.enable	<p>Enables or disables the policy lazy load.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Applies to LDAP and DB stores.</p> <p>Optional.</p> <p>Valid values: TRUE, FALSE.</p> <p>Default value: TRUE.</p>
oracle.security.jps.policystore.policy.cache.strategy	<p>The type of strategy used in the permission cache.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Applies to LDAP and DB stores.</p> <p>Optional.</p> <p>Valid values:</p> <ul style="list-style-type: none"> ■ PERMISSION_FIFO - The cache implements the first-in-first-out strategy. ■ NONE - All entries in the cache grow until a refresh or reboot occurs; there is no control over the size of the cache; not recommended but typically efficient when the policy footprint is very small. <p>Default value: PERMISSION_FIFO.</p>
oracle.security.jps.policystore.policy.cache.size	<p>The number of permissions kept in the permission cache.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Applies to LDAP and DB stores.</p> <p>Optional.</p> <p>Default value: 1000.</p>

Table F-3 (Cont.) Runtime Policy Store Properties

Name	Description
oracle.security.jps.policystore.refresh.enable	<p>Enables or disables the policy store refresh. If this property is set, then oracle.security.jps.ldap.cache.enable cannot be set.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Applies to LDAP and DB stores.</p> <p>Optional.</p> <p>Valid values: TRUE, FALSE.</p> <p>Default value: TRUE.</p>
oracle.security.jps.ldap.cache.enable	<p>Enables or disables the refresh of the cache. If this property is set, then oracle.security.jps.policystore.refresh.enable cannot be set.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Applies to LDAP and DB stores.</p> <p>Optional.</p> <p>Valid values: TRUE, FALSE.</p> <p>Default value: TRUE.</p>
oracle.security.jps.policystore.purge.timeout	<p>The time, in milliseconds, after which the policy store cache is purged.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Applies to LDAP and DB stores.</p> <p>Optional.</p> <p>Default value: 43200000 (12 hours).</p>
oracle.security.jps.policystore.refresh.interval	<p>The interval, in milliseconds, at which the policy store is polled for changes.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Applies to LDAP and DB stores.</p> <p>Optional.</p> <p>Default value: 600000 (10 minutes).</p>
oracle.security.jps.policystore.refresh.permissions.invalidate.threshold	<p>The number of user's permissions after which the permission cache is invalidated.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Applies to LDAP and DB stores.</p> <p>Optional.</p> <p>Default value: 50.</p>

Table F-3 (Cont.) Runtime Policy Store Properties

Name	Description
oracle.security.jps.policystore.rollemember.cache.warmup.enable	<p>Controls the way the ApplicationRole membership cache is created. If set to TRUE, the cache is created at server startup; otherwise, it is created on demand (lazy loading).</p> <p>Set to TRUE when the number of users and groups is significantly higher than the number of application roles; set to FALSE otherwise, that is, when the number of application roles is very high.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Applies to LDAP and DB stores.</p> <p>Optional.</p> <p>Valid values: TRUE, FALSE.</p> <p>Default value: FALSE.</p>
work_folder	<p>The folder for temporary storage.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Applies to XML, LDAP, and DB stores.</p> <p>Optional.</p> <p>Default value: the system temporary folder.</p>
authorization_cache_enabled	<p>Specifies whether the authorization cache should be enabled.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Applies to XML, LDAP, and DB stores.</p> <p>Optional.</p> <p>Valid values: TRUE, FALSE.</p> <p>Default value: FALSE.</p>
session_eviction_percentage	<p>The percentage of sessions to drop when the eviction capacity is reached.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Applies to XML, LDAP, and DB stores.</p> <p>Optional.</p> <p>Default value: 10</p>
session_eviction_capacity	<p>The maximum number of authorization and role mapping sessions to maintain. When the maximum is reached, old sessions are dropped and reestablished when needed.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Applies to XML, LDAP, and DB stores.</p> <p>Optional.</p> <p>Default value: 500</p>
session_expiration_sec	<p>The number of seconds during which session data is cached.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Applies to XML, LDAP, and DB stores.</p> <p>Optional.</p> <p>Default value: 60</p>

Table F–3 (Cont.) Runtime Policy Store Properties

Name	Description
oracle.security.jps.policystore.resourcetypeenforcementmode	<p>Controls the throwing of exceptions if any of the following checks fail:</p> <ul style="list-style-type: none"> ▪ Verify that if two resource types share the same permission class, that permission must be either <code>ResourcePermission</code> or extend <code>AbstractTypedPermission</code>, and this last resource type cannot be created. ▪ Verify that all permissions have resource types defined, and that the resource matcher permission class and the permission being granted match. <p>If set to <code>Strict</code>, when any of the above checks fail, the system throws an exception and the operation is aborted.</p> <p>If set to <code>Lenient</code>, when any of the above checks fail, the system does not throw any exceptions, the operation continues without disruption, and any discrepancies encountered are logged in the log files.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Applies to LDAP and DB stores.</p> <p>Optional.</p> <p>Default value: <code>Lenient</code></p> <p>Valid values: <code>Strict</code>, <code>Lenient</code>.</p>

F.2.2 Credential Store Properties

[Table F–4](#) lists the properties of credential store instances. The properties are listed in two blocks according to the kind of application they can be used in.

Table F–4 Credential Store Properties

Name	Description
The following properties are valid in Java EE applications only	
bootstrap.security.principal.key	<p>The key for the password credentials to access the LDAP credential store, stored in the CSF store.</p> <p>Valid only in Java EE applications.</p> <p>Applies to LDAP and DB stores.</p> <p>Required.</p> <p>No default value.</p> <p>The out-of-the-box value is <code>bootstrap</code>.</p>
bootstrap.security.principal.map	<p>The map for the password credentials to access the LDAP credential store, stored in the CSF store.</p> <p>Valid only in Java EE applications.</p> <p>Applies to LDAP and DB stores.</p> <p>Required.</p> <p>Default value: <code>BOOTSTRAP_JPS</code>.</p>
The following properties are valid in both Java EE and Java SE applications	

Table F-4 (Cont.) Credential Store Properties

Name	Description
oracle.security.jps.farm.name	<p>The RDN format of the domain node in the LDAP credential store.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Applies to LDAP and DB stores.</p> <p>Required.</p> <p>No default value.</p>
oracle.security.jps.ldap.root.name	<p>The RDN format of the root node in the LDAP policy store.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Applies to LDAP and DB stores.</p> <p>Required.</p> <p>No default value.</p>
ldap.url	<p>Specifies the URL of the LDAP credential store using the format <code>ldap://host:port</code>.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Applies only to LDAP stores.</p> <p>Required.</p> <p>No default value.</p>
encrypt	<p>Specifies whether to encrypt credentials.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Applies only to file and LDAP stores.</p> <p>Valid values: <code>true</code>, <code>false</code>.</p> <p>Optional.</p> <p>Default value: <code>false</code>.</p>

The following fragment illustrates the configuration of a credential store in a Java EE application:

```
<serviceInstance provider="ldap.credentialstore.provider" name="credstore.ldap">
  <property value="bootstrap" name="bootstrap.security.principal.key"/>
  <property value="cn=wls-jrfServer" name="oracle.security.jps.farm.name"/>
  <property value="cn=jpsTestNode" name="oracle.security.jps.ldap.root.name"/>
  <property value="ldap://stttt.us.oracle.com:3060" name="ldap.url"/>
  <property value="true" name="encrypt"/>
</serviceInstance>
```

F.2.3 LDAP Identity Store Properties

[Table F-5](#) lists the properties of LDAP-based identity store instances. Extended properties are explicitly stated. User and Role API properties corresponding to a property are also stated.

See Also: [Chapter 7, "Configuring the Identity Store Service"](#).

Table F-5 LDAP-Based Identity Store Properties

Name	Description
idstore.type	<p>The type of the identity store.</p> <p>Valid in Java SE and Java EE applications.</p> <p>Required</p> <p>Valid values:</p> <p>OID - Oracle Internet Directory</p> <p>OVD - Oracle Virtual Directory</p> <p>ACTIVE_DIRECTORY - Microsoft Active Directory</p> <p>IPLANET - Oracle Directory Server Enterprise Edition</p> <p>EDIRECTORY - Novell eDirectory</p> <p>OPEN_LDAP - OpenLdap</p> <p>LIBOVD - Oracle Library OVD</p> <p>CUSTOM - Any other type</p> <p>If using a custom authenticator, the service instance configuration must include one of the following properties:</p> <pre data-bbox="524 821 1224 898"><property name="idstore.type" value="<your-idstore-type>" <property name="ADF_IM_FACTORY_CLASS" value="<your-IDM-FACTOY_CLASS_NAME>"</pre> <p>Corresponding User and Role API property: ADF_IM_FACTORY_CLASS</p>
security.principal.alias	<p>The CSF map name.</p> <p>Valid in Java SE and Java EE applications.</p> <p>Required.</p> <p>No default value.</p> <p>Value example: myalias.</p>
security.principal.key	<p>The CSF key name.</p> <p>Valid only in Java SE applications.</p> <p>Required.</p> <p>No default value.</p> <p>Value example: mykey.</p> <p>Corresponding User and Role API property: ADF_IM_SECURITY_PRINCIPAL</p>
ldap.url	<p>The LDAP URL value.</p> <p>Valid in Java SE and Java EE applications.</p> <p>Required.</p> <p>No default value.</p> <p>Value example: ldap://myServerName.com:389.</p> <p>Corresponding User and Role API property: ADF_IM_PROVIDER_URL</p>

Table F-5 (Cont.) LDAP-Based Identity Store Properties

Name	Description
user.search.bases	<p>The user search base for the LDAP server in DN format. Extended property.</p> <p>Valid in Java SE and Java EE applications.</p> <p>Required.</p> <p>No default value.</p> <p>Value example: cn=users,dc=us,dc=abc,dc=com</p> <p>Corresponding User and Role API property: USER_SEARCH_BASES</p>
group.search.bases	<p>The group or enterprise search base for the LDAP server in DN format. Extended property.</p> <p>Valid in Java SE and Java EE applications.</p> <p>Required</p> <p>No default value.</p> <p>Value example: cn=groups,dc=us,dc=abc,dc=com</p> <p>Corresponding User and Role API property: ROLE_SEARCH_BASES</p>
idstore.config.provider	<p>The idstore provider class.</p> <p>Valid only in Java EE applications.</p> <p>Required</p> <p>The only supported value is:</p> <p>oracle.security.jps.wls.internal.idstore.WlsLdapIdStoreConfigProvider</p>
group.create.bases	<p>The base DN's used to create groups or enterprise roles. Extended property.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Required to allow writing operations with the User and Role API. Otherwise, optional.</p> <p>Value example of a single DN:</p> <pre data-bbox="602 1251 1208 1415"><extendedProperty> <name>group.create.bases</name> <values> <value>cn=groups,dc=us,dc=oracle,dc=com</value> </values> </extendedProperty></pre> <p>Corresponding User and Role API property: ROLE_CREATE_BASES</p>
user.create.bases	<p>The base DN's used to create users. Extended property.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Required to allow writing operations with the User and Role API. Otherwise, optional.</p> <p>Value example of a single DN:</p> <pre data-bbox="602 1671 1192 1835"><extendedProperty> <name>user.create.bases</name> <values> <value>cn=users,dc=us,dc=oracle,dc=com</value> </values> </extendedProperty></pre> <p>Corresponding User and Role API property: USER_CREATE_BASES</p>

Table F-5 (Cont.) LDAP-Based Identity Store Properties

Name	Description
group.filter.object.classes	<p>The fully qualified names of object classes used to search enterprise roles and groups. Extended property.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Optional.</p> <p>Value example: groupOfUniqueNames.</p> <p>Corresponding User and Role API property: ROLE_FILTER_OBJECT_CLASSES</p>
group.mandatory.attrs	<p>The attributes that must be specified when creating enterprise roles or groups. Extended property.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Optional.</p> <p>Value example:</p> <pre data-bbox="524 709 954 905"><extendedProperty> <name>group.mandatory.attrs</name> <values> <value>cn</value> <value>objectClass</value> </values> </extendedProperty></pre> <p>Corresponding User and Role API property: ROLE_MANDATORY_ATTRS</p>
group.member.attrs	<p>The attribute of a static role that specifies the distinguished names (DNs) of the members of an enterprise role or group. Extended property.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Optional.</p> <p>Value example:</p> <pre data-bbox="524 1205 919 1371"><extendedProperty> <name>group.member.attrs</name> <values> <value>uniqueMember</value> </values> </extendedProperty></pre> <p>Corresponding User and Role API property: ROLE_MEMBER_ATTRS</p>
group.object.classes	<p>The fully qualified names of one or more schema object classes used to represent enterprise roles or groups. Extended property.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Optional.</p> <p>Value example:</p> <pre data-bbox="524 1619 954 1814"><extendedProperty> <name>group.object.classes</name> <values> <value>top</value> <value>groupOfUniqueNames</value> </values> </extendedProperty></pre> <p>Corresponding User and Role API property: ROLE_OBJECT_CLASSES</p>

Table F-5 (Cont.) LDAP-Based Identity Store Properties

Name	Description
group.selected.create.base	<p>The base DN's for creating enterprise roles or groups.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Optional.</p> <p>Value example: cn=users,dc=us,dc=abc,dc=com (single DN)</p> <p>Corresponding User and Role API property: ROLE_SELECTED_CREATEBASE</p>
groupname.attr	<p>The attribute that uniquely identifies the name of the enterprise role or group.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Optional.</p> <p>Value example: cn</p> <p>Corresponding User and Role API property: ROLE_NAME_ATTR</p>
group.selected.search.base	<p>The base DN's for searching enterprise roles or groups.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Optional.</p> <p>Value example: cn=users,dc=us,dc=abc,dc=com (single DN)</p>
max.search.filter.length	<p>The maximum number of characters of the search filter.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Optional.</p> <p>Value: a positive integer.</p> <p>Corresponding User and Role API property: MAX_SEARCHFILTER_LENGTH</p>
search.type	<p>The type of search to employ when the repository is queried.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Optional.</p> <p>Valid values: SIMPLE, PAGED, or VIRTUAL_LIST_VIEW.</p> <p>Corresponding User and Role API property: IDENTITY_SEARCH_TYPE</p>
user.filter.object.classes	<p>The fully qualified names of object classes used to search users.</p> <p>Extended property.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Optional.</p> <p>Value example: inetOrgPerson</p> <p>Corresponding User and Role API property: USER_FILTER_OBJECT_CLASSES</p>
user.login.attr	<p>The login identity of the user.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Optional.</p> <p>Value example:</p> <pre data-bbox="602 1766 1284 1787"><property name="user.login.attr" value="mail"/></pre> <p>Corresponding User and Role API property: USER_LOGIN_ATTR</p>

Table F-5 (Cont.) LDAP-Based Identity Store Properties

Name	Description
user.mandatory.attrs	<p>The attributes that must be specified when creating a user. Extended property.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Optional.</p> <p>Value example:</p> <pre data-bbox="524 449 943 674"><extendedProperty> <name>user.mandatory.attrs</name> <values> <value>cn</value> <value>objectClass</value> <value>sn</value> </values> </extendedProperty></pre> <p>Corresponding User and Role API property: USER_MANDATORY_ATTRS</p>
user.object.classes	<p>The fully qualified names of the schema classes used to represent users. Extended property.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Optional.</p> <p>Corresponding User and Role API property: USER_OBJECT_CLASSES</p>
username.attr	<p>The LDAP attribute that uniquely identifies the name of the user.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Optional.</p> <p>Corresponding User and Role API property: USER_NAME_ATTR</p>
ldap.host	<p>The name of the system hosting the identity store.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Optional.</p>
subscriber.name	<p>The default realm for the identity store.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Optional.</p> <p>Value example: dc=us,dc=oracle,dc=com.</p> <p>Corresponding User and Role API property: ADF_IM_SUBSCRIBER_NAME</p>
virtualize	<p>Controls the authenticators where search and modifications are allowed; if set to TRUE, searching and modifying is available in all configured authenticators; otherwise, if set to FALSE, searching and modifying is available in only the first authenticator in the configured stack.</p> <p>Set to TRUE if you intend to use the User and Role API to search or write information in all authenticators.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Optional.</p> <p>Valid values: TRUE or FALSE.</p> <p>Default value: FALSE.</p> <p>Value example:</p> <pre data-bbox="573 1881 1174 1906"><property name="virtualize" value="true"/></pre>

Note: If the authenticator attribute `username` is changed (because, for example, of post-provisioning or migrating from a test to a production environment), then the identity store service parameter `username.attr` in the identity store service must also be changed accordingly. Those two values should be kept equal.

The following fragment illustrates the configuration of an LDAP-based identity store for a Java SE application:

```
<serviceInstance name="idstore.ldap" provider="idstore.ldap.provider">
  <property name="idstore.type" value="OID"/>
  <property name="security.principal.alias" value="MAP_NAME"/>
  <property name="security.principal.key" value="KEY_NAME"/>
  <property name="ldap.url" value="ldap://stadk06:3060"/>
  <extendedProperty>
    <name>user.search.bases</name>
    <values>
      <value>cn=users,dc=us,dc=oracle,dc=com</value>
    </values>
  </extendedProperty>
  <extendedProperty>
    <name>group.search.bases</name>
    <values>
      <value>cn=groups,dc=us,dc=oracle,dc=com</value>
    </values>
  </extendedProperty>
</serviceInstance>
```

F.2.4 Properties Common to All LDAP-Based Instances

[Table F-6](#) lists generic properties of LDAP-based stores that can be specified in any service instance.

In the case of an LDAP-based identity store service instance, to ensure that the User and Role API picks up the connection pool properties when it is using the JNDI connection factory, the identity store service instance *must* include the following property:

```
<property
name="INITIAL_CONTEXT_FACTORY" value="com.sun.jndi.ldap.LdapCtxFactory"/>
```

Table F-6 Generic LDAP Properties

Name	Description
connection.pool.authentication	<p>Specifies the type of LDAP connection that the JNDI connection pool uses.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Optional.</p> <p>Values: none, simple, and DIGEST-MD5.</p> <p>Default value: simple.</p>
connection.pool.max.size	<p>Specifies the maximum number of connections in the LDAP connection pool.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Optional.</p> <p>Value example: 30</p>
connection.pool.min.size	<p>Specifies the minimum number of connections in the LDAP connection pool.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Optional.</p> <p>Value example: 5</p>
connection.pool.protocol	<p>Specifies the protocol to use for the LDAP connection.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Optional.</p> <p>Values: plain, ssl.</p> <p>Default value: plain.</p>
connection.pool.provider.type	<p>Specifies the connection pool to use.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Optional.</p> <p>Values: JNDI, IDM.</p> <p>Default value: JNDI.</p>
connection.pool.timeout	<p>Specifies the number of milliseconds that an idle connection can remain in the pool; after timeout, the connection is closed and removed from the pool.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Optional.</p> <p>Default value: 300000 (5 minutes)</p>
oracle.security.jps.ldap.max.retry	<p>Specifies the maximum number of retry attempts if there are problems with the LDAP connection.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Optional.</p> <p>Value example: 5</p>

The following fragment illustrates a configuration of several properties:

```
<jpsConfig ... >
...
<!-- common properties used by all LDAPs -->
<property name="oracle.security.jps.farm.name" value="cn=OracleFarmContainer"/>
```

```

<property name="oracle.security.jps.ldap.root.name"
          value="cn=OracleJpsContainer"/>
<property name="oracle.security.jps.ldap.max.retry" value="5"/>
...
</jpsConfig>

```

F.2.5 Anonymous and Authenticated Roles Properties

Table F-7 lists the properties that can be used to configure file-, LDAP-, or DB-based anonymous users, anonymous roles, and authenticated roles.

Table F-7 Anonymous and Authenticated Roles Properties

Name	Description
<code>anonymous.role.description</code>	Specifies a description of the anonymous role. Valid in Java EE and Java SE applications. Optional. No default value.
<code>anonymous.role.name</code>	Specifies the name of the principal in the anonymous role. Valid in Java EE and Java SE applications. Optional. Default value: <code>anonymous-role</code>
<code>anonymous.role.uniquename</code>	Specifies the name of the anonymous role. Valid in Java EE and Java SE applications. Optional. Default value: <code>anonymous-role</code>
<code>anonymous.user.name</code>	Specifies the name of the principal in the anonymous user. Valid in Java EE and Java SE applications. Optional. Default value: <code>anonymous</code>
<code>authenticated.role.description</code>	Specifies a description of the authenticated role. Valid in Java EE and Java SE applications. Optional. No default value.
<code>authenticated.role.name</code>	Specifies the name of the principal in authenticated user roles. Valid in Java EE and Java SE applications. Optional. Default value: <code>authenticated-role</code>
<code>authenticated.role.uniquename</code>	Specifies the name of the authenticated role. Valid in Java EE and Java SE applications. Optional. Default value: <code>authenticated-role</code>

Table F-7 (Cont.) Anonymous and Authenticated Roles Properties

Name	Description
<code>remove.anonymous.role</code>	<p>Specifies whether the anonymous role should be removed from the subject after a user is authenticated.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Optional.</p> <p>Valid values: <code>TRUE</code>, <code>FALSE</code>.</p> <p>Default value: <code>FALSE</code>.</p>

F.2.6 Trust Service Properties

[Table F-8](#) lists the properties that can be used to configure the trust service.

Table F-8 Trust Service Properties

Name	Description
<code>trust.aliasName</code>	<p>Specifies the alias to use to get an X.509 certificate and private key from the keystore.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Optional.</p> <p>Default: the name of the WLS domain of the WAS cell.</p>
<code>trust.issuerName</code>	<p>Specifies the name to be included in the token. It is used by the destination trust service to pick up and validate the token.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Optional.</p> <p>Default: the name of the WLS domain of the WAS cell.</p>
<code>trust.csf.map</code>	<p>Specifies the map of the credential to access the keystore.</p> <p>Valid in Java EE and Java SE applications.</p> <p>Optional.</p> <p>Default: the value of the keystore instance property <code>keystore.csf.map</code>.</p>
<code>trust.csf.keystorePass</code>	<p>Specifies the key of the credential to access the private key (the map is set by <code>trust.csf.map</code>).</p> <p>Valid in Java EE and Java SE applications.</p> <p>Optional.</p> <p>Default: the value of the keystore instance property <code>keystore.pass.csf.key</code>.</p>
<code>trust.csf.keyPass</code>	<p>Specifies the key of the credential to access the keystore (the map is set by <code>trust.csf.map</code>).</p> <p>Valid in Java EE and Java SE applications.</p> <p>Optional.</p> <p>Default: the value of the keystore instance property <code>keystore.sig.csf.key</code>.</p>

The following sample illustrates the configuration of a trust service:

```
<propertySet name="trust.provider.embedded">
  <property name="trust.provider.className">
```



```

value="oracle.security.jps.internal.trust.provider.embedded.EmbeddedProviderImpl" /
>
  <property name="trust.clockSkew" value="60"/>
  <property name="trust.token.validityPeriod" value="1800"/>
  <property name="trust.token.includeCertificate" value="false"/>
  <property name="trust.aliasName" value="orakey"/>
  <property name="trust.issuerName" value="orakey"/>
  <property name="trust.csf.map " value="my-csf-map"/>
  <property name="trust.csf.keystorePass" value="my-keystore-csf-key"/>
  <property name="trust.csf.keypass" value="my-signing-csf-key"/>
</propertySet>

```

F.2.7 Audit Service Properties

Table F-9 lists the properties used to configure the audit service:

Table F-9 Audit Service Properties

Property	Description	Required?	Values	Default Value
auditstore.type	The audit metadata store type	yes	file, ldap, or db	file
audit.filterPreset	The level of auditing - None, Low, Medium, and Custom	no	None, Low, Medium, or All	None
audit.customEvents	For Custom, a list of audit events that should be audited. The events must be qualified using the component type. Commas separate events and a semicolon separates component types. Example: JPS:CheckAuthorization, CreateCredential; OIF:UserLogin	no		
audit.specialUsers	list of one or more users whose activity is always audited, even if filterPreset is None. Usernames that contain commas must be escaped properly. For example, when using Fusion Middleware Control, specify three users like this - "admin, fmwadmin, cn=test\,cn=user\,ou:ST\,L=RS\,c=is\," In WLST, the backslash "\" should also be escaped. For example: setAuditPolicy(addSpecialUsers="cn=orcladmin\\,cn=com") For more information, see Section C.4.3, "setAuditPolicy" .	no		
audit.maxDirSize	Controls the size of the directory where the audit files will be written. Integer is in Bytes.	no		102400000
audit.maxFileSize	Controls the size of a bus stop file where audit events are written. Integer is in Bytes	no		104857600
audit.loader.interval	Controls the frequency with which audit loader uploads to database. Integer is in Seconds.	no		15 seconds

Table F–9 (Cont.) Audit Service Properties

Property	Description	Required?	Values	Default Value
audit.loader.repositoryType	Store type for the audit events. If type is Database (Db), also define audit.loader.jndi or JDBC property.	yes	File, DB	File
audit.loader.jndi	JNDI name of the data source in application servers for uploading audit events into database.	no		jdbc/AuditDB
audit.db.principal.map / audit.db.principal.key	The map and key for the JDBC user name and password credential in bootstrap credential store,when running in JavaSE, and repositoryType is DB.	no		
audit.loader.jdbc.string	The JDBC string for JDBC connection when running in JavaSE, and repositoryType is DB.	no		
audit.logDirectory	The base directory for bus-stop files.	required for JavaSE		jse

The following is an example of audit service configuration:

```
<serviceInstance name="audit" provider="audit.provider"
location="./audit-store.xml">
  <property name="audit.filterPreset" value="None" />
  <property name="audit.loader.jndi" value="jdbc/AuditDB" />
  <property name="audit.loader.repositoryType" value="File" />
  <property name="auditstore.type" value="file" />
</serviceInstance>
```

F.2.8 Keystore Service Properties

Table F–10 lists the properties used to configure the Keystore Service:

Table F–10 Keystore Service Properties

Property	Description	Required?	Values	Default
keystore.provider.type	Keystore repository type	Yes	file, ldap, db	file
keystore.file.path	Location of the file keystores.xml when file provider is configured	Yes, if a file-based keystore provider is configured.	-	./
ca.key.alias	Key alias within "system/castore" of the third party CA used for Keystore service instance	No	-	-
location	Location of the keystore; can be absolute or relative path.	Yes, if keystore.type is JKS.No, if keystore.type is PKCS11 or HSM (LunaSA)	Path to keystore	./default-keystore.jks
keystore.type	Type of keystore	Yes	JKS, PKCS11, Luna	JKS
keystore.csf.map	Credential store map name used by Oracle Web Services Manager.	Yes	Credential store map name	oracle.wsm.security

Table F-10 (Cont.) Keystore Service Properties

Property	Description	Required?	Values	Default
keystore.pass.csf.key	Credential store key that points to Keystore password.	Yes, for JKS and PKCS11. No, for HSM	Credential store csf key name	keystore-csf-key
keystore.sig.csf.key	Credential store key name that points to alias and password of signing key in keystore. For HSM, it is the direct key alias name rather than the credential store key name.	Yes	Credential store csf key name or, for HSM, the direct alias	sign-csf-key
keystore.enc.csf.key	Credential store key name that points to alias and password of encryption key in keystore. For HSM, it is the direct key alias name rather than the credential store key name.	Yes	Credential store csf key name or, for HSM, the direct alias	enc-csf-key

The following is an example of Keystore Service configuration for a file-based provider :

```
<serviceInstance name="keystore" provider="keystore.provider"
  location="./default-keystore.jks">
  <description>Default JPS Keystore Service</description>
  <property name="keystore.provider.type" value="file"/>
  <property name="keystore.file.path" value="."/ />
  <property name="keystore.type" value="JKS"/>
  <property name="keystore.csf.map" value="oracle.wsm.security"/>
  <property name="keystore.pass.csf.key" value="keystore-csf-key"/>
  <property name="keystore.sig.csf.key" value="sign-csf-key"/>
  <property name="keystore.enc.csf.key" value="enc-csf-key"/>
</serviceInstance>
```

The following is an example of Keystore Service configuration for an LDAP-based provider :

```
<serviceInstance name="keystore" provider="keystore.provider"
  location="./default-keystore.jks">
  <description>Default JPS Keystore Service</description>
  <property name="keystore.provider.type" value="ldap"/>
  <property name="keystore.type" value="JKS"/>
  <property name="keystore.csf.map" value="oracle.wsm.security"/>
  <property name="keystore.pass.csf.key" value="keystore-csf-key"/>
  <property name="keystore.sig.csf.key" value="sign-csf-key"/>
  <property name="keystore.enc.csf.key" value="enc-csf-key"/>
  <property value="bootstrap" name="bootstrap.security.principal.key"/>
  <property value="cn=wls-jrfServer" name="oracle.security.jps.farm.name"/>
  <property value="cn=jpsTestNode" name="oracle.security.jps.ldap.root.name"/>
  <property value="ldap://stadk06.us.oracle.com:3060" name="ldap.url"/>
</serviceInstance>
```

The following is an example of Keystore Service configuration for an RDBMS-based provider :

```
<propertySet name="props.db.1">
  <property name="datasource.jndi.name" value="opssds"/>
  <property value="cn=farm" name="oracle.security.jps.farm.name"/>
  <property value="cn=jpsroot" name="oracle.security.jps.ldap.root.name"/>
  <property value="dsrc_lookup_key"
    name="bootstrap.security.principal.key"/>
  <property value="credential_map" name="bootstrap.security.principal.map"/>
</propertySet>
```

```
</propertySet>

...
...
<serviceInstance name="keystore.rdbms" provider="keystore.provider"
    location="./default-keystore.jks">
    <propertySetRef ref = "props.db.1" />
    <property name="keystore.provider.type" value="db" />
    <property name="keystore.type" value="JKS" />
    <property name="keystore.csf.map" value="oracle.wsm.security" />
    <property name="keystore.pass.csf.key" value="keystore-csf-key" />
    <property name="keystore.sig.csf.key" value="sign-csf-key" />
    <property name="keystore.enc.csf.key" value="enc-csf-key" />
</serviceInstance>
```

Upgrading Security Data

This appendix describes several procedures to update security data. Specifically, it describes how to upgrade security data from a major release (10.1.3.x) to a major release (11.1.1), and how to upgrade data from a minor release (11g OPSS PS1, PS2, PS3 or PS4) to 11g OPSS PS5, in the following sections:

- [Upgrading with upgradeSecurityStore](#)
- [Upgrading Policies with upgradeOpss](#)

If upgrading from 11gR1 to 11gR1 PS1: For details about this upgrade combination, see section Special Instructions for Oracle Fusion Middleware 11g Release 1 (11.1.1.1.0) in *Oracle Fusion Middleware Installation Planning Guide*.

For an overview and details about Identity Management upgrade, see *Oracle Fusion Middleware Upgrade Guide for Oracle Identity Management*.

G.1 Upgrading with upgradeSecurityStore

The OPSS script `upgradeSecurityStore` is used only to upgrade application security data from a previous major release (such as 10.1.1.3) to more recent one (such as 11.1.1.1). To upgrade between minor 11g releases, use `upgradeOpss` as described in section [Upgrading Policies with upgradeOpss](#).

If the target of the upgrading is an LDAP-based repository, then some setting up before running the script is required, as described in [Section 8.2.2, "Prerequisites to Using an LDAP-Based Security Store."](#)

The script is **offline**, that is, it does not require a connection to a running server to operate, and can be run in interactive mode or in script mode, on WebLogic, and in interactive mode only, on WebSphere. In interactive mode, you enter the script at a command-line prompt and view the response immediately after. In script mode, you write scripts in a text file and run it without requiring input, much like the directives in a shell script.

For platform-specific requirements to run an OPSS script, see [Important Note](#).

Script and Interactive Modes Syntaxes

The script syntax varies depending on the type of store being upgraded. Optional arguments are enclosed in square brackets; arguments in script mode syntax are written in separate lines for clarity of exposition.

To upgrade 10.1.3.x XML identity data to 11g Release 1 (11.1.1) XML identity data, use either of the following syntaxes:

```
updateSecurityStore -type xmlIdStore
                    -jpsConfigFile jpsConfigFileLocation
                    -srcJaznDataFile srcJazn
                    -srcRealm jaznRealm
                    [-dst dstJpsContext]
```

```
updateSecurityStore(type="xmlIdStore", jpsConfigFile="jpsConfigFileLocation",
srcJaznDataFile="srcJazn", srcRealm="jaznRealm", [dst="dstJpsContext"])
```

To upgrade a 10.1.3.x XML policy data to 11g Release 1 (11.1.1) XML policy data, use either of the following syntaxes:

```
updateSecurityStore -type xmlPolicyStore
                    -jpsConfigFile jpsConfigFileLocation
                    -srcJaznDataFile srcJazn
                    [-dst dstJpsContext]
```

```
updateSecurityStore(type="xmlPolicyStore", jpsConfigFile="jpsConfigFileLocation",
srcJaznDataFile="srcJazn", [dst="dstJpsContext"])
```

To upgrade a 10.1.3.x Oracle Internet DirectoryLDAP-based policy data to 11g Release 1 (11.1.1) XML policy data, use either of the following syntaxes:

```
updateSecurityStore -type oidPolicyStore
                    -jpsConfigFile jpsConfigFileLocation
                    -srcJaznConfigFile srcJazn
                    [-dst dstJpsContext]
```

```
updateSecurityStore(type="oidPolicyStore", jpsConfigFile="jpsConfigFileLocation",
srcJaznConfigFile="srcJazn", [dst="dstJpsContext"])
```

To upgrade file-based application policies from release 11.1.1.1.0 to release 11.1.1.2.0, use either of the following syntaxes:

```
updateSecurityStore -type xmlAppPolicies
                    -srcApp applicationStripeName
                    -jpsConfigFile jpsConfigFileLocation
                    -srcJaznDataFile srcJazn
                    -dstJaznDataFile dstJazn
                    -resourceTypeFile resTypeJazn
```

```
updateSecurityStore(type="xmlAppPolicies", srcApp="applicationStripeName",
jpsConfigFile="jpsConfigFileLocation", srcJaznDataFile="srcJazn",
dstJaznDataFile="dstJazn", srcJaznDataFile="resTypeJazn")
```

To upgrade 11.1.1.1.0 application policies to 11.1.1.2.0 format, use either of the following syntaxes:

```
updateSecurityStore -type appPolicies
                    -srcApp applicationStripeName
                    -jpsConfigFile jpsConfigFileLocation
                    -dst dstContext
                    [-resourceTypeFile resTypeJazn]
```

```
updateSecurityStore(type="appPolicies", srcApp="applicationStripeName",
jpsConfigFile="jpsConfigFileLocation", dst="dstContext" [,
resourceTypeFile="resTypeJazn"])
```

This upgrade works in-place and involves the creation of specified resource types and resources corresponding to permissions in the grants.

Once the run completes, the policy store pointed to by the context passed in `dst` in the configuration file passed in `jpsConfigFile` has new resource types and new resources defined for application passed in `srcApp`. The resource types are read from the file specified in `resourceTypeFile` and resources are created corresponding to permissions in the application grants.

The meaning of the arguments is as follows:

- `type` specifies the kind of security data being upgraded. The only valid values are `xmlIdStore`, `xmlPolicyStore`, `oidPolicyStore`, `xmlCredStore`, `xmlAppPolicies`, and `appPolicies`.
- `jpsConfigFile` specifies the location of a configuration file `jps-config.xml` relative to the directory where the script is run. The target store of the upgrading is read from the context specified with the argument `dst`.

In case the type is `xmlAppPolicies`, the configuration file is not used to point to neither source nor destination, but to configure the audit service only. Note that the location must be passed even when the audit service is not specified in the `jps-config.xml` file.

- `srcJaznDataFile` specifies the location of a 10.1.3.x `jazn-data.xml` file relative to the directory where the script is run. This argument is required if the specified type is `xmlIdStore`, `xmlPolicyStore`, or `xmlCredStore`.

In case the specified type is `xmlAppPolicies`, it specifies the location of the application 11.1.1.0 `jazn-data.xml` file, a file that does not include resource type specifications.

- `srcJaznConfigFile` specifies the location of a 10.1.3.x `jazn` configuration file relative to the directory where the script is run. This argument is required if the specified type is `oidPolicyStore`.
- `users` specifies a comma-delimited list of users each formatted as `realmName/userName`. This argument is required if the specified type is `xmlCredStore`.
- `srcRealm` specifies the name of a realm in the file passed to the argument `srcJaznDataFile` that identifies the identities to be migrated. This argument is required if the specified type is `xmlIdStore`.
- `dst` specifies the name of a `jpsContext` in the file passed to the argument `jpsConfigFile` where the destination store is configured. Optional. If unspecified, it defaults to the default `jpsContext`.
- `srcApp` specifies the application stripe. It should match the application name present in the files `srcJaznDataFile` and `resourceTypeFile`. A stripe with this name is created in the file `dstJaznDataFile`.
- `dstJaznDataFile` specifies the location of the application 11.1.1.2.0 `jazn-data.xml` file. This file includes resource type and resource instance specifications and is the replacement for the original `jazn-data.xml` specified in `srcJaznDataFile`.
- `resourceTypeFile` specifies the location of the 11.1.1.2.0 `jazn-data.xml` file which includes resource type specifications.
- `dst` specifies the destination context that points to the policy store to update.

G.1.1 Examples of Use

The following sections contain examples that illustrate the use of the script `upgradeSecurityStore` in different scenarios:

- [Example 1 - Upgrading Identities](#)
- [Example 2 - Upgrading to File-Based Policies](#)
- [Example 3 - Upgrading to Oracle Internet Directory LDAP-Based Policies](#)
- [Example 4 - Upgrading File-Based Policies to Use the Resource Catalog](#)

G.1.1.1 Example 1 - Upgrading Identities

The following invocation illustrates the migration of 10.1.3 file-based identities to an 11g Release 1 (11.1.1) file-based identity store:

```
upgradeSecurityStore -type xmlIdStore
                    -jpsConfigFile jps-config-idstore.xml
                    -srcJaznDataFile jazn-data.xml
                    -srcRealm jazn.com
```

This use of the script assumes that: (a) the files `jps-config-idstore.xml` and `jazn-data.xml` are located in the directory where the script is run; (b) the default `jpsContext` in the file `jps-config-idstore.xml` references the target identity store; and (c) the file `jazn-data.xml` contains a realm named `jazn.com`.

Here are the relevant excerpts of the two files involved in the use sample above:

```
<!-- excerpt from file jps-config-idstore.xml -->
<serviceProviders>
  <serviceProvider name="R11idstore"
class="oracle.security.jps.internal.idstore.xml.XmlIdentityStoreProvider"
type="IDENTITY_STORE">
  <description>11g XML-based IdStore</description>
  </serviceProvider>
</serviceProviders>
...
<serviceInstances>
  <serviceInstance name="idstore.xml1" provider="R11idstore"
location="./jazn-data-11.xml">
  <property name="subscriber.name" value="jazn.com"/>
  <property name="jps.xml.idstore.pwd.encoding" value="OBFUSCATE"/>
  </serviceInstance>
</serviceInstances>
...
<jpsContexts default="default">
  <jpsContext name="default">
  <serviceInstanceRef ref="idstore.xml1" />
  </jpsContext>
</jpsContexts>

<!-- excerpt from jazn-data.xml -->
<jazn-realm>
  <realm>
  <name>jazn.com</name>
  <users> ... </users>
  <roles> ... </roles>
  </realm>
</jazn-realm>
```


Thus, the sample invocation above migrates every user in the element `<users>`, to the XML identity store `R11idStore`.

G.1.1.2 Example 2 - Upgrading to File-Based Policies

The following invocation illustrates the migration of a 10.1.3 file-based policy store to an 11g Release 1 (11.1.1) policy store:

```
upgradeSecurityStore -type xmlPolicyStore
                    -jpsConfigFile jps-config.xml
                    -srcJaznDataFile jazn-data.xml
                    -dst destContext
```

This use of the script assumes that: the files `jps-config.xml` and `jazn-data.xml` are located in the directory where the script is run; and the file `jps-config.xml` contains a `jpsContext` named `destContext`.

Here are the relevant excerpts of the two files involved in the use sample above:

```
<!-- excerpt from file jps-config.xml -->
<serviceProviders>
  <serviceProvider type="POLICY_STORE" name="policystore.xml.provider"
class="oracle.security.jps.internal.policystore.xml.XmlPolicyStoreProvider">
  <description>R11 XML-based PolicyStore Provider</description>
  </serviceProvider>
</serviceProviders>
...
<serviceInstances>
  <serviceInstance name="policystore1.xml" provider="policystore.xml.provider">
  <property name="R11PolStore" value="jazn-data1.xml"/>
</serviceInstance>
...
<jpsContexts default="default1">
  <jpsContext name="default1"> ... </jpsContext>
  <jpsContext name="destContext">
    ...
    <serviceInstanceRef ref="policystore1.xml"/>
  </jpsContext>
</jpsContexts>

<!-- excerpt from jazn-data.xml -->
<jazn-realm>
  <realm>
    ...
    <roles> ... </roles>
  </realm>
</jazn-realm>
...
<jazn-policy> ... </jazn-policy>
```

Thus, the sample invocation above migrates every role in the element `<roles>` and every policy in the element `<jazn-policy>` to the XML policy store `R11PolStore`.

G.1.1.3 Example 3 - Upgrading to Oracle Internet Directory LDAP-Based Policies

The following invocation illustrates the upgrading of a 10.1.4 Oracle Internet Directory LDAP-based policy store to an 11g Release 1 (11.1.1) Oracle Internet Directory LDAP-based policy store:

```
upgradeSecurityStore -type oidPolicyStore
                    -jpsConfigFile jps-config.xml
```

```
-srcJaznConfigFile jazn.xml
-dst destContext
```

The assumptions about the location of the two XML files involved in this example are similar to those in Example 2. In addition, it is assumed that (a) the file `jps-config.xml` contains the `jpsContext destContext` that points to the target Oracle Internet Directory LDAP-based policy store; and (b) the file `jazn.xml` describes the location of the Oracle Internet Directory LDAP server from where the policies are migrated.

Here is the relevant excerpt from the file `jazn.xml`:

```
<jazn provider="LDAP" location="ldap://myCompany.com:3843">
  <property name="ldap.user" value="cn=orcladmin"/>
  <property name="ldap.password" value="!welcome1"/>
  <property name="ldap.protocol" value="no-ssl"/>
  <property name="ldap.cache.policy.enable" value="false"/>
  <property name="ldap.initctx" value="com.sun.jndi.ldap.LdapCtxFactory"/>
</jazn>
```

G.1.1.4 Example 4 - Upgrading File-Based Policies to Use the Resource Catalog

The following invocation upgrades an application 11.1.1.1.0 file-based policy store to an application 11.1.1.2.0 file-based policy store.

```
updateSecurityStore -type xmlAppPolicies
                   -srcApp PolicyServlet1
                   -jpsConfigFile ./folder/jps-config.xml
                   -srcJaznDataFile ./11.1.1.1.0/jazn-data.xml
                   -dstJaznDataFile ./11.1.1.2.0/final-jazn-data.xml
                   -resourceTypeFile ./resCat/res-jazn-data.xml
```

The point of this upgrade is that the original 11.1.1.1.0 file does not use resource catalog elements, but the resulting 11.1.1.2.0 file does use resource type and resource instance elements.

The script basically takes the original application configuration file, along with another file specifying resource type elements, and it produces a new application configuration file that contains policies as in the original file, but modified to use resource catalog specifications.

The original and the new application configuration files provide identical behavior to the application.

The above invocation assumes that:

- The source file `./11.1.1.1.0/jazn-data.xml` contains policies for the application `PolicyServlet1`.
- The resource type file `./resCat/res-jazn-data.xml` contains resource type specifications for the application `PolicyServlet1`.
- The configuration file `./folder/jps-config.xml` is any valid configuration file that may or may not use an audit service instance. In any case, it must be specified.

The following samples illustrate the relevant portions of three data files: the input source `jazn-data.xml` and resource `res-jazn-data.xml`, and the output `final-jazn-data.xml`.

Input Source File `jazn-data.xml`

```
<policy-store>
```

```

<applications>
  <application>
    <name>PolicyServlet1</name>
    <app-roles>
      <app-role>
        <name>myAppRole2</name>
        <display-name>application role myAppRole</display-name>
        <members>
          <member>
            <class>
oracle.security.jps.service.policystore.ApplicationRole</class>
            <name>myAppRole</name>
          </member>
        </members>
      </app-role>
      <app-role>
        <name>myAppRole</name>
        <display-name>application role myAppRole</display-name>
        <members>
          <member>
            <class>
oracle.security.jps.internal.core.principals.JpsXmlEnterpriseRoleImpl</class>
            <name>developers</name>
          </member>
        </members>
      </app-role>
      <app-role>
        <name>testrole_DATA</name>
        <display-name>application role test</display-name>
        <members>
          <member>
            <class>
oracle.security.jps.internal.core.principals.JpsXmlEnterpriseRoleImpl</class>
            <name>test-entrole</name>
          </member>
        </members>
      </app-role>
      <app-role>
        <name>myAppRole_PRIV</name>
        <display-name>application role private</display-name>
        <description>app role private description</description>
        <members>
          <member>
            <class>
oracle.security.jps.internal.core.principals.JpsXmlEnterpriseRoleImpl</class>
            <name>developers</name>
          </member>
          <member>
            <class>
oracle.security.jps.service.policystore.ApplicationRole</class>
            <name>myAppRole</name>
          </member>
        </members>
      </app-role>
    </app-roles>
  </application>
</jazzn-policy>
<grant>
  <grantee>
    <principals>
      <principal>

```

```

        <class>
oracle.security.jps.service.policystore.ApplicationRole</class>
        <name>myAppRole_PRIV</name>
    </principal>
</principals>
</grantee>
<permissions>
    <permission>
        <class>oracle.security.jps.JpsPermission</class>
        <name>getClassLoader</name>
    </permission>
    <permission>
        <class>
oracle.adf.share.security.authorization.RegionPermission</class>
        <name>dummyName</name>
        <actions>view,edit</actions>
    </permission>
</permissions>
</grant>
<grant>
    <grantee>
        <principals>
            <principal>
                <class>
oracle.security.jps.service.policystore.ApplicationRole</class>
                <name>myAppRole</name>
            </principal>
        </principals>
    </grantee>
    <permissions>
        <permission>
            <class>java.lang.XYZPermission</class>
            <name>newxyz</name>
        </permission>
    </permissions>
</grant>
<grant>
    <grantee>
        <principals>
            <principal>
                <class>
oracle.security.jps.internal.core.principals.JpsXmlEnterpriseRoleImpl</class>
                <name>test-entrole</name>
            </principal>
        </principals>
    </grantee>
    <permissions>
        <permission>
            <class>oracle.security.jps.JpsPermission</class>
            <name>newxy</name>
            <actions>view,edit</actions>
        </permission>
    </permissions>
</grant>
</jazn-policy>
</application>
</applications>
</policy-store>

```

Input Resource File res-jazn-data.xml

```

<jazn-data>
  <jazn-realm default="jazn.com">
  </jazn-realm>
  <policy-store>
    <applications>
      <application>
        <name>PolicyServlet1</name>
        <resource-types>
          <resource-type>
            <name>FileResourceType</name>
            <display-name>File Access</display-name>
            <description>Resource Type Modelling File Access</description>
            <provider-name>provider</provider-name>
            <matcher-class>oracle.security.jps.JpsPermission</matcher-class>
            <actions-delimiter>,</actions-delimiter>
            <actions>delete,write,read</actions>
          </resource-type>
        </resource-types>
        <jazn-policy>
        </jazn-policy>
      </application>
    </applications>
  </policy-store>
</jazn-policy>
</jazn-data>

```

Output Data File final-jazn-data.xml

```

<jazn-data>
  <jazn-realm>
  </jazn-realm>
  <policy-store>
    <applications>
      <application>
        <name>PolicyServlet1</name>
        <app-roles>
          <app-role>
            <name>myAppRole2</name>
            <display-name>application role myAppRole</display-name>
            <guid>4341CC10EAFB11DE9F7F17D892026AF8</guid>
            <class>
oracle.security.jps.service.policystore.ApplicationRole</class>
            <members>
              <member>
                <class>
oracle.security.jps.service.policystore.ApplicationRole</class>
                <name>myAppRole</name>
                <guid>43428F60EAFB11DE9F7F17D892026AF8</guid>
              </member>
            </members>
          </app-role>
          <app-role>
            <name>myAppRole</name>
            <display-name>application role myAppRole</display-name>
            <guid>43428F60EAFB11DE9F7F17D892026AF8</guid>
            <class>
oracle.security.jps.service.policystore.ApplicationRole</class>
            <members>

```

```

        <member>
          <class>weblogic.security.principal.WLSGroupImpl</class>
          <name>developers</name>
        </member>
      </members>
    </app-role>
  <app-role>
    <name>testrole_DATA</name>
    <display-name>application role test role</display-name>
    <guid>4342B670EAFB11DE9F7F17D892026AF8</guid>
    <class>
oracle.security.jps.service.policystore.ApplicationRole</class>
    <members>
      <member>
        <class>weblogic.security.principal.WLSGroupImpl</class>
        <name>test-entrole</name>
      </member>
    </members>
  </app-role>
  <app-role>
    <name>myAppRole_PRIV</name>
    <display-name>application role private</display-name>
    <description>app role private description</description>
    <guid>4342B671EAFB11DE9F7F17D892026AF8</guid>
    <class>
oracle.security.jps.service.policystore.ApplicationRole</class>
    <members>
      <member>
        <class>
weblogic.security.principal.WLSGroupImpl</class>
        <name>developers</name>
      </member>
      <member>
        <class>
oracle.security.jps.service.policystore.ApplicationRole</class>
        <name>myAppRole</name>
        <guid>43428F60EAFB11DE9F7F17D892026AF8</guid>
      </member>
    </members>
  </app-role>
</app-roles>
<resource-types>
  <resource-type>
    <name>FileResourceType</name>
    <display-name>File Access</display-name>
    <description>Resource Type Modelling File Access</description>
    <provider-name>provider</provider-name>
    <matcher-class>oracle.security.jps.JpsPermission</matcher-class>
    <actions-delimiter>,</actions-delimiter>
    <actions>delete,write,read</actions>
  </resource-type>
</resource-types>
<resources>
  <resource>
    <name>getClassLoader</name>
    <type-name-ref>FileResourceType</type-name-ref>
  </resource>
  <resource>
    <name>newxy</name>
    <type-name-ref>FileResourceType</type-name-ref>
  </resource>

```

```

    </resource>
  </resources>
  <jazn-policy>
    <grant>
      <grantee>
        <principals>
          <principal>
            <class>
oracle.security.jps.service.policystore.ApplicationRole</class>
            <name>myAppRole_PRIV</name>
            <guid>4342B671EAFB11DE9F7F17D892026AF8</guid>
          </principal>
        </principals>
      </grantee>
      <permissions>
        <permission>
          <class>oracle.security.jps.JpsPermission</class>
          <name>getClassLoader</name>
        </permission>
        <permission>
          <class>
oracle.adf.share.security.authorization.RegionPermission</class>
          <name>dummyName</name>
          <actions>view,edit</actions>
        </permission>
      </permissions>
      <permission-set-refs>
      </permission-set-refs>
    </grant>
    <grant>
      <grantee>
        <principals>
          <principal>
            <class>
oracle.security.jps.service.policystore.ApplicationRole</class>
            <name>myAppRole</name>
            <guid>43428F60EAFB11DE9F7F17D892026AF8</guid>
          </principal>
        </principals>
      </grantee>
      <permissions>
        <permission>
          <class>java.lang.XYZPermission</class>
          <name>newxyz</name>
        </permission>
      </permissions>
      <permission-set-refs>
      </permission-set-refs>
    </grant>
    <grant>
      <grantee>
        <principals>
          <principal>
            <class>
weblogic.security.principal.WLSGroupImpl</class>
            <name>test-entrole</name>
          </principal>
        </principals>
      </grantee>
      <permissions>

```

```
<permission>
  <class>oracle.security.jps.JpsPermission</class>
  <name>newxy</name>
  <actions></actions>
</permission>
</permissions>
<permission-set-refs>
</permission-set-refs>
</grant>
</jazn-policy>
</application>
</applications>
</policy-store>
<jazn-policy>
</jazn-policy>
</jazn-data>
```

G.2 Upgrading Policies with upgradeOpss

upgradeOpss is an offline script that updates PS1, PS2, PS3 or PS4 configurations and stores to a PS5 configuration and store.

The store to be upgraded can be file-, LDAP-, or DB-based and possibly be shared by several WebLogic domains, and the script upgrades system policies, application policies, and the file `jps-config.xml`.

The OPSS binaries and the target policy store must have compatible versions; for details, see [Section L.21, "Incompatible Versions of Binaries and Policy Store."](#)

Important Notes: upgradeOpss *must* be run on the system that hosts the administration server instance so that when the server comes up, the upgraded data is pushed to all managed servers in the cluster.

Before using it, make sure that you backup the store to be upgraded. In case of a LDAP store, backup all data under the root node of the store (which is specified as a property of the store in the configuration file). In case of an upgrade failure, restore that node entirely. For details about backing up, see the documentation for your specific LDAP store.

To upgrade from PS1, PS2, PS3 or PS4 to PS5, proceed as follows:

1. Stop the application server.
2. Install new binaries.
3. In case of upgrading a DB-based store, use Oracle Fusion Middleware Patch Set Assistant to upgrade the DB schema as follows:
 1. Navigate to the OPSS Schema page.
 2. Enter data for Connect String, DBA User Name and Password, and Schema User Name and Password and then click Next.
4. Run upgradeOpss as described in section [Command Syntax](#).
5. Restart the application server.

Note the following points:

- The offline script upgradeOpss:

- Does not change the repository type; that is, if the source policy store is of a given type, then the target policy store is of the same type.
- Applies to an existing domain, which need not be recreated.
- If the target store has already been updated to PS5, then running the script changes nothing.
- In case of a LDAP-based store, the connection parameters to the source and target stores are read from the file `jps-config.xml` or, alternatively, passed as arguments to the script.
- In case of a DB-based store, the connection parameters are passed as arguments to the script.
- In case the security store to be upgraded is shared by several domains (by the join operation), then all domains pointing to that store must install new PS5 binaries *before* the store is upgraded. Otherwise, the system may throw the exception `PolicyStoreIncompatibleVersionException` which indicates that the version of OPSS security store is later than the version of the OPSS binaries.

G.2.1 Command Syntax

To upgrade a file-, LDAP-, or DB-based store, use the syntax below; note that the connection arguments are not required in case of a file-based store; are optional in case of an LDAP-based store; and are required in case of a DB-based store:

```
upgradeOpss(jpsConfig="<full path to the old version jps config file>",
            jaznData="<full path to the new version OOTB JAZN data file>",
            [auditStore="<full path to the OOTB audit-store.xml file>"],
            [jdbcDriver="<jdbc driver>",
            url="<jdbc-ldap url>",
            user="<jdbc-ldap user>",
            password="<jdbc-ldap password>"],
```

The meaning of the arguments is as follows:

- `jpsConfig` specifies the full path to the location of the PS1, PS2, PS3 or PS4 `jps-config.xml` configuration file, which the scripts backs up in the same directory as a file with the suffix `.bak` appended to the its name; required.
- `jaznData` specifies the full path to the location of the PS5 out-of-the-box `system-jazn-data.xml` file; required.
- `auditStore` specifies the full path to the location of the PS5 out-of-the-box `audit-store.xml` file; optional; if unspecified, defaults to the file `audit_store.xml`.
- `jdbcDriver` specifies the JDBC driver to the store; optional in case of LDAP-based store; required in case of DB-based store.
- `url` specifies the JDBC URL or LDAP URL in the format `driverType:host:port:sid`; required in both DB- or LDAP-based store; if not passed, it is read from the configuration file.
- `user` specifies the JDBC user name or LDAP bind name; optional in case of LDAP-based store; required in case of DB-based store; if not passed, it is read from the configuration file. In case of LDAP-based store, the user performing the upgrade must have read and write privileges to the schema, the root node, and all nodes under `cn=OPSS, cn=OracleSchemaVersion`; in case of a DB-based store, perform the upgrade as the OPSS DB schema user.

- `password` specifies the password of the passed `user`; that is, the JDBC password, in case of a DB-based store, or the JDBC bind password, in case of a LDAP-based store; optional in case of LDAP-based store; required in case of DB-based store; if not passed, it is read from the configuration file.

This appendix contains references documentation useful to developers.

H.1 OPSS API References

The following Javadoc documents describe the various APIs that OPSS exposes:

OPSS APIs

Oracle Fusion Middleware Java API Reference for Oracle Platform Security Services

OPSS MBean APIs

Oracle Fusion Middleware MBeans Java API Reference for Oracle Platform Security Services

OPSS User and Role APIs

Oracle Fusion Middleware User and Role Java API Reference for Oracle Platform Security Services

Oracle Security Developer Tools APIs

Oracle Fusion Middleware PKI SDK CMP Java API Reference for Oracle Security Developer Tools

Oracle Fusion Middleware CMS Java API Reference for Oracle Security Developer Tools

Oracle Fusion Middleware Crypto Java API Reference for Oracle Security Developer Tools

Oracle Fusion Middleware PKI SDK LDAP Java API Reference for Oracle Security Developer Tools

Oracle Fusion Middleware Liberty 1.1 Java API Reference for Oracle Security Developer Tools

Oracle Fusion Middleware Liberty 1.2 Java API Reference for Oracle Security Developer Tools

Oracle Fusion Middleware S/MIME Java API Reference for Oracle Security Developer Tools

Oracle Fusion Middleware PKI SDK OCSP Java API Reference for Oracle Security Developer Tools

Oracle Fusion Middleware Security Engine Java API Reference for Oracle Security Developer Tools

Oracle Fusion Middleware SAML 1.0/1.1 Java API Reference for Oracle Security Developer Tools

Oracle Fusion Middleware SAML 2.0 Java API Reference for Oracle Security Developer Tools

Oracle Fusion Middleware PKI SDK TSP Java API Reference for Oracle Security Developer Tools

Oracle Fusion Middleware Web Services Security Java API Reference for Oracle Security Developer Tools

Oracle Fusion Middleware XKMS Java API Reference for Oracle Security Developer Tools

Oracle Fusion Middleware XML Security Java API Reference for Oracle Security Developer Tools

Oracle Fusion Middleware Crypto FIPS Java API Reference for Oracle Security Developer Tools

Oracle Fusion Middleware JCE Java API Reference for Oracle Security Developer Tools

OPSS Scripts

An OPSS script is either a WLST script, in the context of the Oracle WebLogic Server, or a WASAdmin script, in the context of the WebSphere Application Server. The scripts listed in this chapter apply to both platforms: WebLogic Application Server and WebSphere Application Server.

For OPSS scripts details specific to WebSphere Application Server, see *Oracle Fusion Middleware Third-Party Application Server Guide*.

The OPSS security-related scripts are described in the following sections:

- [Policy-Related Scripts](#)
- [Credential-Related Scripts](#)
- [Other Security Scripts](#)
- [Audit Scripts](#)

I.1 Policy-Related Scripts

For details on the following scripts, see [Section 9.3, "Managing Application Policies with OPSS Scripts."](#)

- [listAppStripes](#)
- [createAppRole](#)
- [deleteAppRole](#)
- [grantAppRole](#)
- [revokeAppRole](#)
- [listAppRoles](#)
- [listAppRolesMembers](#)
- [grantPermission](#)
- [revokePermission](#)
- [listPermissions](#)
- [deleteAppPolicies](#)
- [createResourceType](#)
- [getResourceType](#)
- [deleteResourceType](#)
- [createResource](#)

- [deleteResource](#)
- [listResources](#)
- [listResourceActions](#)
- [createEntitlement](#)
- [getEntitlement](#)
- [deleteEntitlement](#)
- [addResourceToEntitlement](#)
- [revokeResourceFromEntitlement](#)
- [listEntitlements](#)
- [grantEntitlement](#)
- [revokeEntitlement](#)
- [listEntitlement](#)
- [listResourceTypes](#)

I.2 Credential-Related Scripts

For details on the following scripts, see [Section 10.5, "Managing Credentials with OPSS Scripts."](#)

- [listCred](#)
- [updateCred](#)
- [createCred](#)
- [deleteCred](#)
- [modifyBootStrapCredential](#)
- [addBootStrapCredential](#)
- [exportEncryptionKey](#)
- [importEncryptionKey](#)
- [restoreEncryptionKey](#)

I.3 Other Security Scripts

- [migrateSecurityStore](#)
For details, see [Section 8.6.2, "Migrating with the Script migrateSecurityStore."](#)
- [reassociateSecurityStore](#)
For details, see [Section 9.3.29, "reassociateSecurityStore."](#)
- [upgradeSecurityStore](#)
For details, see [Section G.1, "Upgrading with upgradeSecurityStore."](#)
- [upgradeOpss](#)
For details, see [Section G.2, "Upgrading Policies with upgradeOpss."](#)

I.4 Audit Scripts

For the description of audit-related scripts, see [Section C.4, "WLST Commands for Auditing."](#)

Using an OpenLDAP Identity Store

This appendix describes the special set up required in case the identity store uses OpenLDAP 2.2.

J.1 Using an OpenLDAP Identity Store

To use OpenLDAP 2.2 as an identity store, proceed as follows:

1. Use the WebLogic Server administration console to create a new authenticator provider. For this new provider:
 - Select OpenLDAPAuthenticator from the list of authenticators.
 - Set the control flag of the OpenLDAPAuthenticator to SUFFICIENT.
 - Set the control flag of the DefaultAuthenticator to SUFFICIENT.
 - Change the order of authenticators to make the OpenLDAPAuthenticator the first in the list.
 - In the Provider Specific page for the OpenLDAPAuthenticator, enter User Base DN and Group Base DN, and set the value of the objectclass in the Group From Name Filter to something other than groupofnames.
2. From the Home directory of the OpenLDAP installation:
 - Open the file `slapd.conf` for edit.
 - In that file, insert the following line in the "include" section at the top:

```
include ../schema/inetorgperson.schema
```
 - Save the file, and restart the OpenLDAP.

The above settings make possible adding the object class `inetorgperson` to every new external role you create in the OpenLDAP; this object class is required to map the external role to an application role.

Adapter Configuration for Identity Virtualization

The identity virtualization feature, described in [Section 7.3, "Configuring the Identity Store Service"](#), requires some additional configuration to support a split profile.

This appendix describes how to create and manage the adapters used for split profiles.

- [About Split Profiles](#)
- [Configuring a Split Profile](#)
- [Deleting a Join Rule](#)
- [Deleting a Join Adapter](#)
- [Changing Adapter Visibility](#)

K.1 About Split Profiles

The Identity Virtualization feature enables you to query multiple LDAP directories through OPSS. For example, you can fetch data from both Oracle Internet Directory and Microsoft Active Directory in a single query.

The feature supports a "split profile," where an application makes use of attributes for a single identity that are stored on two different sources; for example, where the username, password, and employeeID for a single person are stored on Microsoft Active Directory, and that person's employeeID and business role are stored in Oracle Internet Directory.

For example, when a WebCenter application needs to obtain attributes for a single identity from more than one source directory, it uses the split profile to leverage the join functionality of Identity Virtualization. These joins use a standard join adapter. For details, see:

- [Understanding Oracle Virtual Directory Adapters in the *Oracle Fusion Middleware Administrator's Guide for Oracle Virtual Directory*](#)
- [Understanding the Join View Adapter in the *Oracle Fusion Middleware Administrator's Guide for Oracle Virtual Directory*](#)

The adapter configuration is stored in `adapters.os_xml`, but connection details such as host, port and credentials of a back-end directory come from OPSS.

K.2 Configuring a Split Profile

The same user occurs in both identity stores with some attributes in one store and other attributes in the other store. A query on the user record requires data from both stores. The configuration tasks are:

1. Configure the identity store service with the `virtualize` property to enable queries against multiple LDAP stores.

For details, see [Section 7.3, "Configuring the Identity Store Service."](#)

2. Connect to the Weblogic AdminServer to run WLST commands to configure the join adapter for the identity stores.

For details about how to bring up the WLST prompt, see "Getting Started Using Command-Line Tools" in the *Oracle Fusion Middleware Administrator's Guide*.

3. Create the join adapter in the primary identity store:

```
createJoinAdapter(adapterName="Join Adapter Name", root="Namespace",
primaryAdapter="Primary adapter Name")
```

4. Add the join rule to the secondary store(s):

```
addJoinRule(adapterName="Join Adapter Name", secondary="Secondary Adapter
Name", condition="Join Condition")
```

Note: If there is more than one secondary identity store, run the `addJoinRule` command for each secondary store.

5. Run the `modifyLDAPAdapter` command:

```
modifyLDAPAdapter(adapterName="AuthenticatorName", attribute="Visible",
value="Internal")
```

Note: If there is more than one secondary identity store, run the `modifyLDAPAdapter` command for each secondary ID store.

Example

In this example the same user occurs in two stores; the first store is Microsoft Active Directory and the second store is Oracle Internet Directory. In the example, we assume that Microsoft Active Directory is the primary store and Oracle Internet Directory is the secondary store.

Note: When configuring the LDAP connection parameters, the `user.create.bases` and `group.create.bases` must correspond to the primary adapter's namespace. For details about the parameters, see [Section 7.3.1, "What is Configured?."](#)

Authenticator 1

Authenticator Name: Microsoft Active Directory (AD)

User Base: `cn=users,dc=acme,dc=com`

Authenticator 2

Authenticator Name: Oracle Internet Directory (OID)

User Base: cn=users, dc=oid, dc=com

The steps to implement the split profile are as follows:

1. Create the join adapter:

```
createJoinAdapter(adapterName="JoinAdapter1", root="dc=acme,dc=com",
primaryAdapter="AD")
```

The adapter name shown here is an example; use an appropriate name in actual usage.

2. Specify the join rule:

```
addJoinRule(adapterName="JoinAdapter1", secondary="OID", condition="uid=cn")
```

"uid=cn" is the join condition in the above example which indicates that if uid value of a user in Oracle Internet Directory (secondary) matches with cn value of the Microsoft Active Directory user (primary), then the attributes are combined.

The attribute on the left side of the condition is the attribute in the secondary adapter and the attribute on the right side is the attribute in the primary adapter.

3. Modify the adapters:

```
modifyLDAPAdapter(adapterName="OID", attribute="Visible", value="Internal")
```

```
modifyLDAPAdapter(adapterName="AD", attribute="Visible", value="Internal")
```

The adapter names used here are the actual name of the authenticators. The adapter names in all the primary and secondary parameters also refer to the authenticator name. The join adapter name can be any name you choose.

4. Restart Weblogic Admin and Managed servers.

K.3 Deleting a Join Rule

You use the `removeJoinRule` command to remove a join rule from a join adapter.

Syntax

```
removeJoinRule
adapterName = "adapterName"
secondary = "Secondary Adapter associated with the JoinRule"
```

Example

```
removeJoinRule(adapterName="JoinAdapter1", secondary="OID")
```

K.4 Deleting a Join Adapter

You use the `deleteAdapter` command to delete a join adapter.

Syntax

```
deleteAdapter(adapterName = "name")
```

Example

```
deleteAdapter(adapterName="JoinAdapter1")
```

K.5 Changing Adapter Visibility

You use the `modifyLDAPAdapter` command to change the visibility of the adapters.

For example:

```
modifyLDAPAdapter(adapterName="AuthenticatorName", attribute="Visible",  
value="Yes")
```

Troubleshooting Security in Oracle Fusion Middleware

This appendix describes common problems that you may encounter when configuring and using Oracle Enterprise Manager Fusion Middleware security, and explains how to solve them. It contains the following sections:

- [Diagnosing Security Errors](#)
- [Reassociation Failure](#)
- [Server Fails to Start](#)
- [Failure to Grant or Revoke Permissions - Case Mismatch](#)
- [Failure to Connect to an LDAP Server](#)
- [Failure to Connect to the Embedded LDAP Authenticator](#)
- [User and Role API Failure](#)
- [Failure to Access Data in the Credential Store](#)
- [Failure to Establish an Anonymous SSL Connection](#)
- [Authorization Check Failure](#)
- [User Gets Unexpected Permissions](#)
- [Security Access Control Exception](#)
- [Runtime Permission Check Failure](#)
- [Permission Failure Before Server Starts](#)
- [Policy Migration Failure](#)
- [Characters in Policies](#)
- [Granting Permissions in Java SE Applications](#)
- [Troubleshooting Oracle Business Intelligence Reporting](#)
- [Search Failure when Matching Attribute in Policy Store](#)
- [Search Failure with an Unknown Host Exception](#)
- [Incompatible Versions of Binaries and Policy Store](#)
- [Incompatible Versions of Policy Stores](#)
- [Need Further Help?](#)

L.1 Diagnosing Security Errors

This section the tools available to diagnose and solve a variety of security errors. It contains the following sections:

- [Log Files and OPSS Loggers](#)
- [System Properties](#)
- [Solving Security Errors](#)

The logging support with Fusion Middleware Control is explicitly stated whenever the tool can help managing, isolating, or interpreting faults when they occur.

L.1.1 Log Files and OPSS Loggers

This section describes the various log files and OPSS loggers supported by Oracle WebLogic Server and how to configure, set logger levels, and locate and view log files with Fusion Middleware Control, in the following sections:

- [Diagnostic Log Files](#)
- [Generic Log Files](#)
- [Authorization Loggers](#)
- [Offline OPSS Scripts Loggers](#)
- [Other OPSS Loggers](#)
- [Audit Loggers](#)
- [Managing Loggers with Fusion Middleware Control](#)

L.1.1.1 Diagnostic Log Files

Each server instance in a domain writes all OPSS-based exceptions raised by its subsystems and applications to a server log file in the file system of the local host computer.

By default, this log file is located in the `logs` directory below the server instance root directory. The names of these log files have the following format:

ServerName-diagnostic.logxxxxx, where xxxxx denotes an integer between 1 and 99999.

Here are some examples of diagnostic file full names:

DomainName/servers/AdminServer/logs/AdminServer-diagnostic.log00001 (administration server log),

DomainName/servers/soa/logs/soa-diagnostic.log00013 (managed server log).

All server instances output security-related errors to diagnostic files. Server-related security errors, such as exceptions raised by issues with a subject or principal, and errors that may occur while migrating or reassociating domain security data, get written in the administration server diagnostic log. Application-related security errors, such as exceptions raised by application-specific policies or credentials, get written in the corresponding managed server diagnostic log.

L.1.1.2 Generic Log Files

In addition to diagnostic log files, Oracle WebLogic Server supports other log files for each server in a domain and for each domain in a topology.

By default and similar to diagnostic log files, server log files are located in the `logs` directory below the server instance root directory. Domain log files are located in the `logs` directory below the administration server root directory. The names of these log files have the format `ServerName.logxxxxx` and `domain.logxxxxx`, where `xxxxx` denotes an integer between 1 and 99999.

Here are some examples of server and domain log files full names:

`DomainName/servers/AdminServer/logs/AdminServer.log00001`,
`DomainName/servers/AdminServer/logs/domain1.log00033`.

Server and domain logs are files where one should look for generic errors, such as exception raised by authenticators or other domain service providers.

The domain logs duplicate some messages written to server logs (for servers in the domain), and they help determine the server where a fault has occurred in a domain that contains a large number of servers.

Note: The generation of a new log file is determined by its rotation policy; typically, the rotation is determined by file size, so when a log file exceeds a specified size, the system generates a new one with a name whose integer suffix is increased by 1.

For details about particular loggers, see [Authorization Loggers](#) and [Audit Loggers](#).

Related Documentation

For information about server log files and domain log files, see section Server Log Files and Domain Log Files in *Oracle Fusion Middleware Configuring Log Files and Filtering Log Messages for Oracle WebLogic Server*.

For information about the Oracle WebLogic Framework, see *Oracle Fusion Middleware Configuring and Using the Diagnostics Framework for Oracle WebLogic Server*.

For additional information about logging services, see *Oracle Fusion Middleware Using Logging Services for Application Logging for Oracle WebLogic Server*.

For complete details about logging in Oracle Fusion Middleware, see chapter Managing Log Files and Diagnostic Data in *Oracle Fusion Middleware Administrator's Guide*.

L.1.1.3 Authorization Loggers

OPSS provides two loggers that help troubleshooting runtime authorization failures:

- [oracle.security.jps.util.JpsAuth](#)
- [oracle.security.jps.trace.logger](#)

These two loggers, as any other OPSS logger, can be enabled and disabled dynamically, that is, without having to stop and restart the Oracle WebLogic Application Server; for details about setting the properties of a logger, see [Managing Loggers with Fusion Middleware Control](#). The level of the above two loggers must be set to `TRACE:32`.

For information about additional loggers, see [Other OPSS Loggers](#).

L.1.1.3.1 oracle.security.jps.util.JpsAuth

The logger `oracle.security.jps.util.JpsAuth` logs the start and return of the method `checkPermission`; the following snippets of a log file illustrate the entry and exit demarcations to this method in the log file:

```
[SRC_CLASS: oracle.security.jps.util.JpsAuth] [APP: JeeScenarioApp]
[SRC_METHOD: Entering checkPermission] ENTRY
(oracle.security.jps.ResourcePermission
resourceType=TaskFlowResourceType,resourceName=ResourceNameX read)

[SRC_CLASS: oracle.security.jps.util.JpsAuth] [APP: JeeScenarioApp]
[SRC_METHOD: Exiting checkPermission] RETURN
java.security.AccessControlException: access denied
(oracle.security.jps.ResourcePermission
resourceType=TaskFlowResourceType,resourceName=ResourceNameX read)
```

The following snippet illustrates a successful authorization log:

```
[JpsAuth] Check Permission
PolicyContext: [JeeScenarioApp]
Resource/Target: [getSubjectFromDomainCombiner]
Action:[null]
Permission Class: [javax.security.auth.AuthPermission]
    Result:                [SUCCEEDED]
    Subject:                [null]
    Evaluator:              [SM]
```

The following snippet illustrates an unsuccessful authorization log:

```
[JpsAuth] Check Permission
PolicyContext: [JeeScenarioApp]
Resource/Target: [resourceType=TaskFlowResourceType,resourceName=ResourceNameX]
Action:[read]
Permission Class: [oracle.security.jps.ResourcePermission]
    Result:                [FAILED]
    Evaluator:              [ACC]
    Failed
```

L.1.1.3.2 oracle.security.jps.trace.logger The logger

oracle.security.jps.trace.logger logs information about application roles, permissions, targets, principals, and granted and denied policies. Since enabling this logger can lead to a large output, it is recommended that it be used to debug a single use case only. Specifically, this logger records:

- The following information about an authorization request: the application roles granted to an enterprise role, all deny's and grant's, the permission class names, the permission targets, and the principal names.
- Member cache updates, such as when a principal is added to a role; keywords: "Principal:", "Inserted Roles:".
- Role managing information; keywords: "In App:", "Query Store for Principal:", "Number of direct app roles:".
- Calls to the method getPermissions.

L.1.1.4 Offline OPSS Scripts Loggers

When using offline OPSS scripts, such as migrateSecurityStore, OPSS loggers can be enabled by starting the JVM with the following system property:

```
-Djava.util.logging.config.file=<path>/logging.properties
```

where logging.properties is a text file with the required logger properties enabled; the format of this file is described in the documentation of the class java.util.logging.LogManager. A sample logging.properties file enabling OPSS loggers at appropriate levels is the following:

```
#The messages will be written to a file
handlers=java.util.logging.FileHandler

#The default level for all loggers is INFO
.level=INFO

#For common usage - user manager, jps config etc.
oracle.jps.common.level=FINEST

# For Migration and Upgrade
oracle.jps.upgrade.level=FINEST
oracle.jps.patching.level=FINEST
oracle.jps.policymgmt.level=FINEST

#Configure file information. %h is the user home directory (user.home)
java.util.logging.FileHandler.pattern = /tmp/opss/opss_upgrade%u.log
java.util.logging.FileHandler.limit = 50000
java.util.logging.FileHandler.count = 1
java.util.logging.FileHandler.formatter = java.util.logging.SimpleFormatter
```

L.1.1.5 Other OPSS Loggers

In addition to authorization loggers, OPSS provides the following loggers:

`oracle.jps.common` enables diagnosing issues with the OPSS JpsFilter and the OPSS JpsInterceptor.

`oracle.jps.deployment` enables diagnosing issues with OPSS artifacts packed with the application, when the application is deployed; keyword:"migration".

`oracle.jps.openaz` enables diagnosing issues with PEP API calls. Setting `oracle.jps.openaz.level` to `FINEST`, logs information about submitted requests - identity, resource, action, context - and authorization results.

L.1.1.6 Audit Loggers

There are several run-time components in the Fusion Middleware Audit Framework. This section helps you navigate the diagnostic log files for these components and explains how to interpret diagnostic messages.

The log files are located at:

`DomainName/servers/$SERVER_NAME/logs/$SERVER_NAME-diagnostic.log`

Table L-1 lists the various diagnostic log files.

Table L-1 Log Files for Audit Diagnostics

Component	Log Location	Configuring Loggers
Java EE Components using Audit APIs	DomainName/servers/\$SERVER_NAME/logs/\$SERVER_NAME-diagnostic.log	oracle.security.audit.logger (See instructions below)
OPMN Component Using Audit APIs	See the Administration Guide for the component to locate its log files.	Loggers are based on the OPMN Components's Location. Please see the corresponding component guide.
Startup Class Audit Loader	DomainName/servers/\$SERVER_NAME/logs/\$SERVER_NAME-diagnostic.log	oracle.security.audit.logger (See instructions following this table)

Table L-1 (Cont.) Log Files for Audit Diagnostics

Component	Log Location	Configuring Loggers
OPMN Audit Loader	\$ORACLE_INSTANCE/diagnostics/logs/OPMN/opmn/rmd.out	java.util.logging.config.file system property can be set to the file that contains the log level for OPMN Audit Loader
Config/Proxy Mbeans	DomainName/servers/\$SERVER_NAME/logs/\$SERVER_NAME-diagnostic.log	oracle.security.audit.logger (See instructions below)
Audit Schema Support	RCU log location (Default is \$ORACLE_HOME/rcu/log/)RCU_LOG_LOCATION can be set to change this location	RCU log level (Default is ERROR) RCU_LOG_LEVEL - [SEVERE; ERROR; NOTIFICATION; TRACE

L.1.1.6.1 Configuring the Audit Loggers

You can configure oracle.security.audit.logger using Fusion Middleware Control.

oracle.security.audit.logger can take any log level from ERROR to TRACE allowing control over the amount of information that gets logged.

You can also view these diagnostic files with Fusion Middleware Control.

See Also: For more information about the following topics, see chapter 10, *Managing Log Files and Diagnostic Data*, in *Oracle Fusion Middleware Administrator's Guide*:

- instructions for configuring the loggers
- details on viewing logs from domain, server, and each application

L.1.1.6.2 Interpreting Audit Diagnostics

The Audit diagnostic messages can be categorized into two types - errors and trace messages.

All error messages are numbered IAU-XXX. These messages are found in the Error Message Guide with a proper cause and an action that can be taken to rectify the error.

The trace messages, however, are meant to provide more information about the running components. Depending on its nature, a message may require some action on your part.

L.1.1.7 Managing Loggers with Fusion Middleware Control

Fusion Middleware Control provides several pages to manage log information. Using this tool you can:

- Configure several attributes of a log file, including the log level and rotation.
- Search the contents of all log files in a domain and group the results of a query by message ID or type.
- Correlate a given error with others by context or time span.
- Download a portion of a log file or the results of a query in one of several formats.

This section explains briefly how to configure a log file. The other three functions above are explained, also briefly, in section [Section L.1.3, "Solving Security Errors."](#)

For full details about these topics, see section *Managing Log Files and Diagnostic Data*, in the *Oracle Fusion Middleware Administrator's Guide*.

To configure a log file with Fusion Middleware Control, proceed as follows:

1. Navigate to *Server > Logs > Log Configuration*, to display the **Log Configuration** page for the selected server. This page allows you to configure the log level for both persistent loggers and active run-time loggers.
2. Click the Log File entry for the desired logger, to display the page showing the current parameter settings for that file.
3. In this page, select a row and then click the button **Edit Configuration**, to display the **Edit Log File** dialog, where you can set various parameters, including the log level and the rotation policy; typically, the logger level is set to TRACE:32.

L.1.2 System Properties

To increase the debug output, set one of the following system properties to the script that starts your Oracle WebLogic Server and restart the server:

- `jps.auth.debug`
- `jps.auth.debug.verbose`

To get debug output during the authorization process, set any of the system properties described in section [Debugging the Authorization Process](#).

Two other system properties that can be passed at server start and that can help debugging security issues are the following:

- `-DDebugOPSSPolicyLoading`, a flag that monitors the progress and setting of the OPSS policy provider.
- `-Djava.security.debug=policy`, the standard Java security debug flag that produces print information about policy files as they are parsed, including their location in the file system, the permissions they grant, and the certificates they use for signed code.

Note: A consequence of setting a high logging output is that many threads may be reported in a stuck state, specially when file loading takes place. To avoid this situation, change the time out value that Oracle WebLogic Server uses to mark a thread as stuck to a higher value.

A system property cannot be set without restarting the server. In order to set a system property the administrator must edit the `setDomainEnv.sh` shell script and add the property to the environment variable `EXTRA_JAVA_PROPERTIES` in that script.

L.1.2.1 `jps.auth.debug`

Assume that just this system property is set to true:

```
-Djps.auth.debug=true
```

Then, a permission check that fails generates an output with details illustrated in the following sample:

```
[JpsAuth] Check Permission
          PolicyContext:      [jps-wls-Demo]
          Resource/Target:    [app.monitor]
```

```

        Action:                [read,write]
        Permission Class:      [java.util.PropertyPermission]
        Evaluator:             [ACC]
        Result:                 [FAILED]
Failed ProtectionDomain:ClassLoader=weblogic.servlet.jsp.JspClassLoader@fb111c
finder: weblogic.utils.classloaders.CodeGenClassFinder@106bb21 annotation:
CodeSource=file:/C:/MyOracle/domains/base_domain/servers/AdminServer/tmp/_WL_
user/jps-wls-Demo/keqfo/jsp_servlet/test.class
Principals=total 5 of principals(
  1. weblogic.security.principal.WLSUserImpl "duane"
  2. weblogic.security.principal.WLSGroupImpl "employee"
  3. JpsPrincipal: oracle.security.jps.principals.JpsAuthenticatedRoleImpl
"authenticated-role"
  4. JpsPrincipal: oracle.security.jps.principals.JpsAnonymousRoleImpl
"anonymous-role"
  5. JpsPrincipal: oracle.security.jps.service.policystore.ApplicationRole
"appRoleEmployee")
Permissions=(
  (java.util.PropertyPermission line.separator read)
  ...
  (oracle.security.jps.service.credstore.CredentialAccessPermission
context=SYSTEM,mapName=default,keyName=* read,write))

```

A permission check that succeeds generates no output. To disable permission check messages, set this property to false; by default, it is set to true. Disabling permission check messages is not recommended in production environments.

L.1.2.2 jps.auth.debug.verbose

Assume that `jps.auth.debug` and `jps.auth.debug.verbose` are *both* set to true:

```

-Djps.auth.debug=true
-Djps.auth.debug.verbose=true

```

Then, a permission check that succeeds generates an output with details illustrated in the following sample:

```

[JpsAuth] Check Permission
        PolicyContext:        [jps-wls-Demo]
        Resource/Target:      [app.monitor]
        Action:                [read,write]
        Permission Class:     [java.util.PropertyPermission]
        Result:                 [SUCCEEDED]
        Subject:                [total 5 of principals(
  1. weblogic.security.principal.WLSGroupImpl "manager"
  2. weblogic.security.principal.WLSUserImpl "shawn"
  3. JpsPrincipal:
oracle.security.jps.internal.core.principals.JpsAuthenticatedRoleImpl
"authenticated-role" GUID=null DN=null
  4. JpsPrincipal:
oracle.security.jps.internal.core.principals.JpsAnonymousRoleImpl "anonymous-role"
GUID=null DN=null
  5. JpsPrincipal: oracle.security.jps.service.policystore.ApplicationRole
"appRoleManager" GUID=null DN=null)]
        Evaluator:             [ACC]

```

A permission check that fails generates an output with details illustrated in the following sample:

```

JpsAuth] Check Permission
        PolicyContext:        [jps-wls-Demo]

```

```

Resource/Target:      [app.monitor]
Action:              [read,write]
Permission Class:    [java.util.PropertyPermission]
Evaluator:           [ACC]
Result:              [FAILED]
Failed

ProtectionDomain:ClassLoader=weblogic.servlet.jsp.JspClassLoader@1b7682d finder:
weblogic.utils.classloaders.CodeGenClassFinder@7d32cf annotation:
CodeSource=file:/C:/Mydom/domains/domain/servers/AdminServer/jpservlet/test.class
Principals=total 5 of principals(
  1. weblogic.security.principal.WLSUserImpl "duane"
  2. weblogic.security.principal.WLSGroupImpl "employee"
  3. JpsPrincipal: oracle.security.principals.JpsAuthenticatedRoleImpl
"authenticated-role" GUID=null DN=null
  4. JpsPrincipal: oracle.security.principals.JpsAnonymousRoleImpl "anonymous-role"
GUID=null DN=null
  5. JpsPrincipal: oracle.security.jps.service.policystore.ApplicationRole
"appRoleEmployee" GUID=null DN=null)
Permissions=(
  (java.util.PropertyPermission line.separator read)
  ...
  (java.lang.RuntimePermission stopThread))
Call Stack: java.security.AccessControlException: access denied
(java.util.PropertyPermission app.monitor read,write)
java.security.AccessControlContext.checkPermission(AccessControlContext.java:323)
...
weblogic.work.ExecuteThread.run(ExecuteThread.java:173)
  ProtectionDomains for class stack:
Class[0]: class oracle.security.jps.util.JpsAuth$Diagnostic$SMSupport
ProtectionDomain: ClassLoader=sun.misc.Launcher$AppClassLoader@360be0
CodeSource=file:/C:/MyOracle/jdeveloper/modules/oracle.jps_11.1.1/jps-api.jar
Principals=total 0 of principals<no principals>
Permissions=(
  (java.io.FilePermission \C:\MyOracle\jdeveloper\modules\jps-api.jar read)
  ...
)
Class[1]: class oracle.security.jps.util.JpsAuth$Diagnostic$SMSupport

```

To disable permission check messages, set both `jps.auth.debug` and `jps.auth.debug.verbose` to false; by default, `jps.auth.debug.verbose` is set to false.

L.1.2.3 Debugging the Authorization Process

This section describes the use of several other system properties that help debugging the authorization process based on several criteria. Specifically, the following system properties:

```

oracle.security.jps.log.for.approle.substring
oracle.security.jps.log.for.permeffect
oracle.security.jps.log.for.permclassname
oracle.security.jps.log.for.permtarget.substring
oracle.security.jps.log.for.enterprise.principalname

```

generate logging messages during the following authorization phases:

- Phase 1 - The application roles that were granted to an enterprise user or to an enterprise role during the OPSS Subject computation.
- Phase 2 - The permission instances that were granted to a grantee.

- Phase 3 - The outcome of a permission check, that is, whether the grant was granted or denied.

Each of the above properties and the phases they apply are described next.

`oracle.security.jps.log.for.approle.substring` - During phases 1, 2, and 3, it logs the name of an application role that contains a specified substring; if the substring to match is unspecified, it logs all application role names.

`oracle.security.jps.log.for.permeffect` - During phase 3 and according to a specified value, it logs a grant that was granted or denied; if the value is unspecified, it logs all grants (regardless whether they were granted or denied).

`oracle.security.jps.log.for.permclassname` - During phases 2 and 3, it logs the name of the permission class that matches exactly a specified name; if the name to match is unspecified, it logs all permission class names.

`oracle.security.jps.log.for.permtarget.substring` - During phases 2 and 3, it logs the name of a permission target that contains a specified substring; if the substring to match is unspecified, it logs all permission targets.

`oracle.security.jps.log.for.enterprise.principalname` - During phases 1, 2, and 3, it logs the name of the principal (enterprise user or enterprise role) that matches exactly a specified name; if the name to match is unspecified, it logs all principal names.

The following characteristics apply to all of the above system properties:

- They are optional.
- They can be set at most once.
- The matchings (where they apply) are case insensitive

To enable the logging of any of the above system properties, proceed as follows:

1. Set the desired system properties.
2. Stop the JVM.
3. Restart the JVM.
4. Set the logger `oracle.security.jps.dbg.logger` to `TRACE:32`. For details on how to set a logger, see [Managing Loggers with Fusion Middleware Control](#).
5. Run the scenario to be debugged.
6. Examine the log output; to locate the messages output by the settings of any of the above properties, search the log file for the key word `[oracle.security.jps.dbg.logger]`.

L.1.2.3.1 Examples of Use The following examples illustrate typical settings of the above system properties.

- To log all application role names that contain the substring `myAppRole`, include the following setting:

```
-Doracle.security.jps.log.for.approle.substring=myAppRole
```

- To log all denied permission checks, include the following setting:

```
-Doracle.security.jps.log.for.permeffect=deny
```

- To log all granted permission checks, include the following setting:

```
-Doracle.security.jps.log.for.permeffect=grant
```


- To log all granted or denied permission checks, do not set `oracle.security.jps.log.for.permeffect`.
- To log all permission checks that match exactly the class name `java.util.PropertyPermission`, include the following setting:
`-Doracle.security.jps.log.for.permclassname=java.util.PropertyPermission`
- To log all target names that contain the substring `p.mon`, include the following setting:
`-Doracle.security.jps.log.for.permtarget.substring=p.mon`
- To log all authorizations involving the principal name manager, include the following setting:
`-Doracle.security.jps.log.for.enterprise.principalname=manager`
- To log application role names that match a substring or principal names that match a string, set both `oracle.security.jps.log.for.approle.substring` and `oracle.security.jps.log.for.enterprise.principalname` as indicated above.
- To log *all* application roles names and *all* principal names, set neither `oracle.security.jps.log.for.approle.substring` nor `oracle.security.jps.log.for.enterprise.principalname`.

L.1.3 Solving Security Errors

There is no generic way to resolve errors when they occur. One must search for hints and frequently follow multiple hypotheses until, hopefully, the source of the error is isolated and understood. To this end, this section describes how to search and interpret log information to resolve most common security errors. These topics are addressed in the following sections:

- [Understanding Sample Log Entries](#)
- [Searching Logs with Fusion Middleware Control](#)
- [Identifying a Message Context with Fusion Middleware Control](#)
- [Generating Error Listing Files with Fusion Middleware Control](#)

L.1.3.1 Understanding Sample Log Entries

Understanding log error output is crucial to isolate and solve an error. Let's take a closer look at a diagnostic log file to describe the information you find for an error logged in such a file. This description is best illustrated with a real-life example.

The following is an excerpt of an error in the file `AdminServer-diagnostic.log`:

```
[2009-01-07T09:15:02.393-08:00] [AdminServer] [ERROR] [JPS-00004]
[oracle.jps.admin]
[tid: [ACTIVE].ExecuteThread: '3' for queue: 'weblogic.kernel.Default
(self-tuning)'] [userId: weblogic] [ecid: 0000Hum5kxw7MAn54nU4Ui19PD8S000005,0]
Unable to add principal to the application role. Reason: Principal
"abc.xxx@myComp.com" is already a member of the application role
"BPMWorkflowAdmin"[[
java.security.PrivilegedActionException:
oracle.security.jps.service.policystore.PolicyObjectAlreadyExistsException:
Unable to add principal to the application role. Reason: Principal
"abc.xxx@myComp.com" is already a member of the application role
```

```
"BPMWorkflowAdmin"
    at java.security.AccessController.doPrivileged(Native Method)
    at oracle.security.jps.mas.mgmt.jmx.policy.JpsApplicationPolicyStoreImpl.
addRemoveMembersToRole(JpsApplicationPolicyStoreImpl.java:408)
    at oracle.security.jps.mas.mgmt.jmx.policy.JpsApplicationPolicyStoreImpl.
addMembersToApplicationRole(JpsApplicationPolicyStoreImpl.java:385)
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
...

```

The meaning of the fields in the preceding message is as follows:

- [2009-01-07T09:15:02.393-08:00]
Identifies the date and time when the error was logged.
- [AdminServer]
Identifies the name of the server where the error occurred.
- [JPS-00004]
Identifies the error code and hints to the kind of error that occurred. For a complete list of JPS error codes, see chapter 41 in *Oracle Fusion Middleware Error Messages Reference*.
- [oracle.jps.admin]
Identifies the category of the logger. The subcategories of `oracle.jps` (such as `admin` above) hint to the kind of error that occurred. For the complete list of categories under `oracle.jps`, see [Subcategories of oracle.jps](#).
- [tid: [ACTIVE].ExecuteThread: '3' for queue: 'weblogic.kernel.Default (self-tuning)']
Identifies the thread where the error occurred.
- [userId: weblogic]
Identifies the user that performed the operation that generated the error.
- [ecid: 0000Hum5kxw7MA54nU4Ui19PD8S000005,0]
Identifies the execution context id. Typically used to correlate and trace sequence of events. Ecids provide information about the flow across processes, such as, from a request, to the WebLogic server, to an Oracle Internet Directory server.
- Unable to add principal to the application role. Reason: Principal abc.xxx@myComp.com is already a member of the application role BPMWorkflowAdmin
Identifies the reason why the error was logged.
- java.security.PrivilegedActionException:
oracle.security.jps.service.policystore.PolicyObjectAlreadyExistsException: Unable to add principal to the application role. Reason: Principal abc.xxx@myComp.com is already a member of the application role BPMWorkflowAdmin
Identifies the exception that was raised and the reason for it.

Subcategories of oracle.jps

Here is the list of subcategories under `oracle.jps` and the kind of errors logged in the category:

- common - generic errors.
- config - configuration errors.

- deployment - deployment errors.
- authentication - login module errors in Java SE applications only.
- idmgmt - identity store errors.
- credstore - credential store errors.
- authorization - policy store errors at run time.
- policymgmt - policy store management errors.
- admin - JMX and WLST errors.

L.1.3.2 Searching Logs with Fusion Middleware Control

To initiate a search in the contents of all log files in a domain, select *Domain* > **Logs** > **View Log Messages**, to display the **Log Messages** page.

In this page you have several parameters that you can choose from to specify your search query; specifically, you can:

- Choose a time interval in which a message was issue, by selecting the appropriate **Date Range**.
- Display messages with a given severity error, by checking any of the **Message Types** boxes.
- Display messages satisfying further constrains, by choosing an item from the menu **Message** and entering a string in the box next to it. For example, you could query for just messages that contain the string **exception** in it.
- Add extra query fields, by clicking the button **Add Fields** and checking any of the available choices. For example, you could add the field **Host**, and then enter the appropriate query, such as **starts with** a particular string.

Once these parameters are set, click **Search** and the result of the query is displayed in the page. The result of a query can be further redisplayed by message type, message ID, or simple list of messages, by selecting an item from the menu **Show**. Moreover, the result can be automatically refreshed by choosing an item from the menu at the top right of the page (by default set to Manual Refresh).

To broaden a search to log files beyond a domain, use the button **Broaden Target Scope** at the top right of the page.

L.1.3.3 Identifying a Message Context with Fusion Middleware Control

In some situations, it is necessary to know the context in which a message has occurred. For example, it may be useful to know messages that have preceded or followed a given error message by, say, 2 minutes.

The tab **View Related Messages** provides this functionality, and you can use it as follows:

1. Display the results of a query with **Show** set to **Messages**.
2. Select a message within the result table. Note that the tabs **View Related Messages** and **Export Messages to File** become then available. Let's assume, for example, that the selected message has the time stamp Jan 21, 2009 4:05:00 PM PST.
3. Select **Time Interval** from the **Date Range** menu, and enter a **Start Date** and an **End Date**. For example, you could enter Jan 21, 2009 4:02:00 PM, as a start date, and Jan 21, 2009 4:07:00 PM.

4. Select **by Time** from the menu **View Related Messages**, to display the page with all the messages related to the selected one in the specified time span.
5. In the **Related Messages by Time** page, you can modify the time window around the time of the selected message by choosing an item from the menu **Scope**, at the right of the page.

L.1.3.4 Generating Error Listing Files with Fusion Middleware Control

In some situations, you may want to download the list of errors displayed into a separate file to forward it, for example, to a support center, or just to keep it for your records.

Whenever available, the tab **Export Messages** allows you to generate a file containing just the displayed results by choosing an item from the menu. The format of the generated file can be plain text, XML, or CSV.

The following sample, showing only the first of 29 messages, is an excerpt of a text file generated this way:

```
#
#Search Criteria
#Start Time: 2009-01-21T16:34:41.381-08:00
#End Time: 2009-01-21T16:39:41.381-08:00
#Message Types: ERROR, WARNING

#Selected Targets List
#/Farm_base_domain/base_domain/AdminServer:Oracle WebLogic Server
#/Farm_base_domain/base_domain/AdminServer/DMS Application(11.1.1.1.0):Application
Deployment
#/Farm_base_domain/base_domain/AdminServer/em:Application Deployment
#/Farm_base_domain/base_domain/AdminServer/wsdl-wls:Application Deployment
#/Farm_base_domain/base_domain/AdminServer/wsm-pm:Application Deployment
#
[2009-01-21T16:34:54.045-08:00] [AdminServer] [WARNING] []
[org.apache.myfaces.trinidad.bean.PropertyKey] [host: stacz39] [nwaddr:
140.87.5.40] [tid: 13] [userId: <anonymous>] [ecid:
0000HvvkgjVE^MT6uBj8EH19TvXj000008,0] [APP: em] [Target: /Farm_base_domain/base_
domain/AdminServer/em] [Target Type: Application Deployment] Unserializable
value:oracle.sysman.core.view.tgtctls.common.DefaultTreeModel@1fcadd2 for
key:UINodePropertyKey[value,17]
...
#
#Number of messages exported: 29
#
```

L.2 Reassociation Failure

Policy and credential reassociation from an file-based store to an LDAP-based store may fail for several reasons. This section explains three reasons why this operation may fail.

Symptom 1- Error Code 32

Reassociation fails and an error like the following is logged in the administration server diagnostic file `serverName.diagnostic.log`:

```
[LDAP: error code 32 - No Such Object]
Authentication to LDAP server ldap://myServer.com:3060 is unsuccessful.
```

Diagnosis 1

The error above identifies a problem with the target node in the LDAP server, namely, that the node specified does not exist.

It is required that the root node specified in the text box **JPS Root DN** (of the page **Set Security Provider**) be present in the LDAP directory *before* invoking the reassociation.

Solution 1

Verify that the data you enter in the box **JPS Root DN** matches the name of a node in the target LDAP directory, and then rerun the reassociation.

Symptom 2- Error Code 68

Reassociation fails and an error like the following is logged in the administration server diagnostic file `serverName.diagnostic.log`:

```
Authentication to LDAP server ldap://myServer.com:3060 is successful.
Starting to migrate policy store...
Set up security provider reassociation successfully.
Checked and seeded security store schema successfully.
null
[LDAP: error code 68 - Object already
exists]:cn=SystemPolicy,cn=domain1,cn=JPSText,cn=nb_policy
Error occurred while migrating LDAP based policy store.
```

Diagnosis 2

The error above indicates that the name specified in the box **WebLogic Domain Name** is a descendant (more precisely, a grandchild) of the **JPS Root DN** node in the target LDAP directory.

It is required that the domain specified do *not* be a descendant of the root node.

Solution 2

Verify that the name you enter in the box **WebLogic Domain Name** does not match the name of a grandchild of the specified **JPS Root DN** node, and rerun the reassociation.

Symptom 3

Reassociation, carried out with Fusion Middleware Control, fails and an error like the following is logged in the administration server diagnostic file `serverName.diagnostic.log`:

```
[2009-01-21T10:09:24.326-08:00] [AdminServer] [ERROR] [] [oracle.jps.admin] [tid
: [ACTIVE].ExecuteThread: '15' for queue: 'weblogic.kernel.Default (self-tuning)
'] [userId: weblogic] [ecid: 0000HvuOTpe7q2T6uBADUH19Tpyb000006,0] Unable to rem
ove the principal from the application role. Reason: Principal "Managers" is not
a member of the application role "test-role"[[
java.security.PrivilegedActionException: oracle.security.jps.service.policystore
.PolicyObjectNotFoundException: Unable to remove the principal from the applicat
ion role. Reason: Principal "Managers" is not a member of the application role "
test-role"
    at oracle.security.jps.mas.mgmt.jmx.policy.JpsApplicationPolicyStoreImpl
.addRemoveMembersToRole(JpsApplicationPolicyStoreImpl.java:408)...
```

Diagnosis 3

The error above points to some problem with the application role `test-role`, which is, in this case, the root of the problem.

Ensure that when entering data to perform reassociation with Fusion Middleware Control, you use the button **Test LDAP Authentication** immediately after you have completed entering all required values to connect to the target LDAP server. This test catches any problems with those values before reassociation begins.

Solution 3

In our example, a quick inspection of the file `system-jazn-data.xml` reveals that the application `test-role` is used by an application policy, but it was not defined. Here is an excerpt of that file illustrating where the required data is missing:

```
<application>
  <name>myApp</name>
  <app-roles>
    <!--! test-role should have been defined here -->
  </app-roles>
  <jazn-policy>
    <grant>
      <grantee>
        <principals>
          <principal>
            <class>
oracle.security.jps.service.policystore.ApplicationRole</class>
            <name>test-role</name>
            <guid>66368900E7E511DD9F62F9ADA4233FE2</guid>
          </principal>
        </principals>...
```

To solve this particular error, (a) fix `system-jazn-data.xml` by inserting the definition of the application `test-role`; (b) revert to file-based domain stores with the fixed file; and (c) rerun the reassociation.

L.2.1 Missing Policies in Reassociated Policy Store

Symptom

When an file-based policy store is reassociated to use an LDAP-based Oracle Internet Directory policy store, the reassociation may report that it completed successfully.

At runtime, however, the system does not behave as expected. Codebase policies, that are supposed to be present in the system policy after migration, are missing.

Diagnosis

At runtime, the server reports a stack trace that resembles the following:

```
<BEA-000000> <JspServlet: initialization complete>
###<May 4, 2009 8:32:50 AM PDT> <Error> <HTTP> <ap626atg> <WLS_Spaces>
<[ACTIVE] ExecuteThread: '3' for queue: 'weblogic.kernel.Default
(self-tuning)'\> <<WLS Kernel>> <> <> <1241451170341> <BEA-101020>
<[ServletContext@20193148[app:webcenter module:/webcenter path:/webcenter
spec-version:2.5]] Servlet failed with Exception
java.security.AccessControlException: access denied
(oracle.security.jps.service.policystore.PolicyStoreAccessPermission
context=APPLICATION,name=webcenter getApplicationPolicy)
  at
  java.security.AccessControlContext.checkPermission(AccessControlContext.java:323)
    at
  java.security.AccessController.checkPermission(AccessController.java:546)
    at
  oracle.security.jps.util.JpsAuth$AuthorizationMechanism$3.checkPermission(JpsAuth.
```

```

java:348)
    at
oracle.security.jps.util.JpsAuth$Diagnostic.checkPermission(JpsAuth.java:268)
    at
oracle.security.jps.util.JpsAuth$AuthorizationMechanism$6.checkPermission(JpsAuth.
java:372)
    at oracle.security.jps.util.JpsAuth.checkPermission(JpsAuth.java:408)
    at oracle.security.jps.util.JpsAuth.checkPermission(JpsAuth.java:431)
    at
oracle.security.jps.internal.policystore.AbstractPolicyStore.checkPolicyStoreAcces
sPermission(AbstractPolicyStore.java:246)
    at
oracle.security.jps.internal.policystore.ldap.LdapPolicyStore.getApplicationPolicy
(LdapPolicyStore.java:281)
    at
oracle.security.jps.internal.policystore.PolicyUtil.getGrantedAppRoles(PolicyUtil.
java:898)
    at
oracle.security.jps.internal.policystore.PolicyUtil.getJpsAppRoles(PolicyUtil.java
:1354)
    at
oracle.security.jps.wls.JpsWlsSubjectResolver$1.run(JpsWlsSubjectResolver.java:273
)
    at
oracle.security.jps.wls.JpsWlsSubjectResolver$1.run(JpsWlsSubjectResolver.java:270
)
    at java.security.AccessController.doPrivileged(Native Method)

```

Here the permission:

```

oracle.security.jps.service.policystore.PolicyStoreAccessPermission
context=APPLICATION,name=webcenter getApplicationPolicy

```

is granted to a code base, and the authorization is not allowed since it evaluates to false.

Solution

Check the AdminServer diagnostic logs for messages like these:

```

AdminServer-diagnostic.log:[2009-05-28T02:27:52.249-07:00] [AdminServer]
[NOTIFICATION] [JPS-00072] [oracle.jps.config] [tid: Thread-39] [ecid:
0000I66Z0KH0fplp4sm3Ui1A7_Rl00002s,1:5001] [arg: 11.1.1.1.0] [arg: 11.1.1.0.0]
Policy schema upgrade not required. Store Schema version 11.1.1.1.0 is compatible
to the seed schema version 11.1.1.0.0
AdminServer-diagnostic.log:[2009-05-28T02:28:58.012-07:00] [AdminServer]
[NOTIFICATION] [JPS-00078] [oracle.jps.config] [tid: Thread-39] [ecid:
0000I66Z0KH0fplp4sm3Ui1A7_Rl00002s,1:5001] [arg: 11.1.1.1.0] [arg: 11.1.1.0.0]
Credential store schema upgrade not required. Store Schema version 11.1.1.1.0 is
compatible to the seed schema version 11.1.1.0.0

```

A message of this type suggests that the schema was never seeded during the re-association. If the correct schema is not seeded in the Oracle Internet Directory server, the system will not work as expected.

To ensure that the schema is seeded during re-association, proceed as follows:

1. Remove the cn=OPSS container under the cn=OracleSchemaVersion container in the Oracle Internet Directory server.
2. Start with a clean working instance of an OPSS policy store using the file-based store.

3. Re-associate this file-based store to the Oracle Internet Directory server.

Check the AdminServer diagnostic logs to confirm that the OPSS LDAP schema was seeded in the LDAP server by looking for this message:

```
AdminServer-diagnostic.log:[2009-05-29T07:18:18.002-07:00] [AdminServer]
[NOTIFICATION] [JPS-00078] [oracle.jps.config] [tid: Thread-12] [ecid:
0000I61Z0MH0fplp4sm3Ui1A7_L100002s,1:5001] [arg: 11.1.1.0.0] Policy schema
version set to 11.1.1.0.0
```

If re-associating to a Release 11g Oracle Internet Directory server, the schema version should read: 11.1.1.1.0

If re-associating to a Release 10.1.4.3 Oracle Internet Directory server, the schema version should read: 11.1.1.0.0

The Policy Store schema version is set in the Oracle Internet Directory server under this container:

```
cn=PolicyStore,cn=OPSS,cn=OracleSchemaVersion
```

Similarly, the Credential Store schema version is set in the Oracle Internet Directory server under this container:

```
cn=CredentialStore,cn=OPSS,cn=OracleSchemaVersion
```

L.2.2 Unsupported Schema

This section explains a reason why reassociation to an LDAP server may fail.

Symptom

Reassociating the security store to an LDAP repository fails and the AdminServer log reports an error like the following:

```
[2011-02-09T07:01:13.884-05:00] [AdminServer] [ERROR] [] [oracle.jps.admin] [tid:
[ACTIVE].ExecuteThread: '6' for queue: 'weblogic.kernel.Default (self-tuning)']
[userId: weblogic] [ecid:
41050d66ef2ec40b:-4c1fb689:12e06cc7b6c:-8000-00000000000001e1,0] Schema seeding
failed, check the server type of the given ldap url. [[
oracle.security.jps.JpsException: Error Modifying JPS Schema, Record: dn:
cn=schema
changetype: modify
delete: objectclasses
objectclasses: ( 2.16.840.1.113894.7.2.2 NAME 'orclContainer' SUP ( top ) MUS
T ( cn ) MAY ( orclVersion $ orclServiceType ) )
-
: [LDAP: error code 32 - No Such Object]:cn=schema
```

Diagnosis

The error `LDAP: error code 32` indicates that the schema of the reassociation target LDAP repository is not supported, that is, the version of the target LDAP repository is not one of the OPSS supported LDAP stores.

Solution

Update the target LDAP repository to one of the supported LDAP stores and then try reassociating again. The version of an LDAP OID store must be 10.1.4.3 or later. For a list of supported versions, see [Section 8.2, "Using an LDAP-Based OPSS Security Store."](#)

L.3 Server Fails to Start

This section explains several reasons why the Oracle WebLogic Server may fail to start in the following sections:

- [Missing Required LDAP Authenticator](#)
- [Missing Administrator Account](#)
- [Missing Permission](#)
- [Server with NFS-Mounted Domain Directory Fails to Start](#)

L.3.1 Missing Required LDAP Authenticator

This section explains a reason why the Oracle WebLogic Server may fail to start after modifying the list of authenticators in a domain.

Symptom

After modifying the list of authenticator providers in a domain, the Oracle WebLogic Server fails to start, and the error messages output include the following:

```
java.lang.IllegalArgumentException: null KeyStore name
```

Diagnosis

One cause of this problem is that the list of authenticators in your domain does not include an LDAP authenticator.

Important: An LDAP authenticator is *required* in this list for any domain using OPSS.

Solution

Since the server cannot start, you must add one LDAP authenticator manually, as follows:

1. Open the file `DOMAIN_NAME/config/config.xml`.
2. Edit `config.xml` and include, within the element `<realm>`, an LDAP authenticator, such as the default authenticator illustrated in the following sample:

```
<realm>
  ...
  <sec:authentication-provider xsi:type="wls:default-authenticatorType">
  </sec:authentication-provider>
  ...
</realm>
```

3. Restart the server.

Once the server is back up and running, you can modify the list of providers to include the provider of your choice using the WebLogic Administration Console, but ensure that at least one of them is an LDAP authenticator provider.

To this end, use the WebLogic Administration Console as follows:

1. Navigate to the page **Create a new Authenticator Provider**.
2. Enter the authenticator name and select an authenticator type, all of which are LDAP-based:
 - ActiveDirectoryAuthenticator

- DefaultAuthenticator (this is the one inserted manually in the sample above)
- LDAPAuthenticator
- LDAPX509IdentityAsserter
- OpenLDAPAuthenticator
- OracleInternetDirectoryAuthenticator
- OracleVirtualDirectoryAuthenticator

L.3.2 Missing Administrator Account

This section explains a reason why the Oracle WebLogic Server may fail to start.

Symptom

After removing the out-of-box default authenticator and adding, say an Oracle Internet Directory authenticator, the server fails to start.

Diagnosis

Most likely, you have forgotten to enter an account member of the Administrators group in your added authenticator. The server requires that such an account be present in one domain authenticator. This account is always present in the default authenticator.

Solution

Since the server cannot start, you must add the deleted one LDAP authenticator manually, as follows:

1. Open the file `DOMAIN_NAME/config/config.xml`.
2. Edit `config.xml` and include, within the element `<realm>`, the default authenticator, as illustrated in the following sample:

```
<realm>
  ...
  <sec:authentication-provider xsi:type="wls:default-authenticatorType">
  </sec:authentication-provider>
  ...
</realm>
```

3. Restart the server.

Once the server is back up and running, proceed as follows:

1. Use the WebLogic Administration Console to create in the Oracle Internet Directory authenticator an account that is member of the Administrators group.
2. Set the Oracle Internet Directory authenticator flag to SUFFICIENT.
3. Restart the server, which it should start without problems, since it is using the account in the Administrators group provided in the default authenticator.
4. Reset the Oracle Internet Directory authenticator flag to REQUIRED and remove the default authenticator. The server should now start using the account in the Administrators group that you created in the Oracle Internet Directory authenticator.

L.3.3 Missing Permission

This section explains a reason why the Oracle WebLogic Server may fail to start.

Symptom

The server fails to start when it started with security manager is enabled (with the system property `-Djava.security.manager`).

Diagnosis

One reason why you may run into this issue is the lack of permission grants to PKI APIs in `oraclepki.jar` when the security manager is enabled at server startup.

Solution

Ensure that a grant like the following is present in the file `weblogic.policy`, or add it if it is not:

```
grant codeBase "file:${oracle.home}/modules/oracle.pki_${jrf.version}/*" {
    permission java.security.AllPermission;
};
```

The above grant is provided by default. Note that when security manager is enabled, the access to all system resources requires codebase permission grants.

For complete details about using the Java Security Manager to protect WebLogic resources, see *Oracle Fusion Middleware Programming Security for Oracle WebLogic Server*.

Note: Printing Security Manager is a WebLogic server enhancement to the Java Security Manager. Use Printing Security Manager to identify all of the required permissions for a Java application running under Java Security Manager. Unlike the Java Security Manager, which identifies needed permissions one at a time, the Printing Security Manager identifies *all* the needed permissions without intervention.

L.3.4 Server with NFS-Mounted Domain Directory Fails to Start

This section explains a reason why the Oracle WebLogic Server will fail to start.

Symptom

The domain directory `${domain.home}/config/fmwconfig` is on an NFS-mounted partition, and when the server is started an error message like the following is logged:

```
JPS-01050: Opening of wallet based credential store failed. Reason
java.io.IOException: PKI-02002: Unable to open the wallet. Check password.
```

Furthermore, when `orapki` debugging is turned on and the server is started once again, the following message is logged:

```
java.io.IOException: No locks available.
```

Note: To enable `orapki` debugging, start the server with the following property set: `-Doracle.pki.debug=true`.

Diagnosis

The real cause for the server's failure to come up is reported in the second error message above, once `orapki` has been enabled. Since OPSS requires file locking when managing security artifacts in file-based stores, that error message indicates that the file system on which the domain directory is NFS-mounted does not support file locking.

Solution

Perform either of the following:

- Upgrade from NFS v3 to NFS v4.
- Mount the remote file system with the `nolock` option enabled.
- Move files in `${domain.home}/config/fmwconfig` to a local storage

L.3.5 Other Causes

This section explains several reasons why the Oracle WebLogic Server may fail to start.

Symptom

When attempting to load and set the policy provider, the Oracle WebLogic Server fails to start and logs an exception similar to the one illustrated in the following snippet:

```
<Mar 30, 2010 3:15:54 PM EDT> <Error> <Security> <BEA-090892> <The dynamic loading
of the OPSS java security policy provider class
oracle.security.jps.internal.policystore.JavaPolicyProvider failed due to problem
inside OPSS java security policy provider. Exception was thrown when loading or
setting the JPSS policy provider.
...
<Mar 30, 2010 3:15:54 PM EDT> <Critical> <WebLogicServer> <BEA-000386> <Server
subsystem failed. Reason: weblogic.security.SecurityInitializationException: The
dynamic loading of the OPSS java security policy provider class
oracle.security.jps.internal.policystore.JavaPolicyProvider failed due to problem
inside OPSS java security policy provider. Exception was thrown when loading or
setting the JPSS policy provider.
...
weblogic.security.SecurityInitializationException: The dynamic loading of the OPSS
java security policy provider class
oracle.security.jps.internal.policystore.JavaPolicyProvider failed due to problem
inside OPSS java security policy provider. Exception was thrown when loading or
setting the JPSS policy provider.
...

```

Diagnosis

The server startup includes loading and setting the policy provider as defined in the configuration file `jps-config.xml`; if this task is not completed successfully, the Oracle WebLogic Server fails to start. As illustrated in the sample above, this type of failure is identified in the server's log by the string

Exception was thrown when loading or setting the JPSS policy provider.

To determine the root cause of a particular failure server startup, check the server's log file and inspect the logged stack trace. For details about identifying errors, see [Diagnosing Security Errors](#).

Here are some reasons why the server fails to start:

1. The path to the configuration file is incorrectly specified.

2. The default context is missing in the configuration file.
3. The XML parser is not available.
4. A code source URL is incorrectly specified in a system policy. This situation is identified by a logged exception that includes the string

```
java.net.MalformedURLException: unknown protocol.
```

Solution

A solution for each of the above cases above is explained next.

1. Ensure that the correct path is specified by the system parameter `oracle.security.jps.config`:

```
-Doracle.security.jps.config=<full-path-to-jps-config.xml>
```

Note that special characters (such as backslashes or white space characters) in the full path specification must be properly escaped. One way to verify correctness is to test using the specified full path in a command line.

2. The configuration file must include a default context. For an example of a default context configuration, see [<jpsContext>](#).
3. Make sure that the XML parser is available in your system and that the XML parser JAR file is included in the classpath.
4. Typical incorrect and corrected code source URLs are illustrated in the following two samples.

Sample 1 - Incorrect URL

```
<grantee>
  <codesource>
    <url>${my.oracle.home}/jlib/webcacheua.jar</url>
  </codesource>
</grantee>
```

Sample 1 - Corrected URL (in bold)

```
<grantee>
  <codesource>
    <url>file:${my.oracle.home}/jlib/webcacheua.jar</url>
  </codesource>
</grantee>
```

Sample 2 - Incorrect URL

```
<grantee>
  <codesource>
    <url>c:/myfolder/jlib/webcacheua.jar</url>
  </codesource>
</grantee>
```

Sample 2 - The corrected URL (in bold) is either one of the following three:

```
<grantee>
  <codesource>
    <url>file:///c:/myfolder/jlib/webcacheua.jar</url>
  </codesource>
</grantee>
```

```
<grantee>
  <codesource>
    <url>file:c:/myfolder/jlib/webcacheua.jar</url>
  </codesource>
```

```
</grantee>

<grantee>
  <codesource>
    <url>file:/c:/myfolder/jlib/webcacheua.jar</url>
  </codesource>
</grantee>
```

For details about the syntax of URL specifications in a code source (including the use of system variables), see [<url>](#).

L.4 Failure to Grant or Revoke Permissions - Case Mismatch

This section explains the likely reasons why an enterprise user or role (group) may fail to be granted or revoked permissions.

Symptom

An enterprise user or group, properly entered in a domain authenticator, is not granted or revoked the permissions defined by a grant.

Diagnosis

This problem is likely to occur when there is a case mismatch between the stored name (in a domain authenticator) and the supplied name (either actively entered by a user or obtained programmatically). For example, this mismatch would occur when the stored user name is *JdOE* and the supplied user name is *jdoe*.

Solution

There are two ways to resolve this issue.

The first solution involves setting the appropriate property in the authenticator being used in your domain. As long as both strings (the supplied and the stored) contain identical sequence of characters (irrespective of case), this setting guarantees that the user name populated in the Subject matches the user name present in a domain authenticator, even when the corresponding characters differ in case. Thus, when this setting is in place, the user names *JdOE* and *jdoe* match.

To set your domain authenticator property, proceed as follows:

1. Use the Administration Console to navigate to the page where your authenticator is configured. For example, if you are using the default authenticator, navigate to the DefaultAuthenticator page by choosing **Realms > myrealm > Providers > DefaultAuthenticator**.
2. Choose the tab **Provider Specific**.
3. Set the property **userRetrievedUserNameAsPrincipal** to true.
4. Restart the server.

The second solution considers the case where the supplied name is obtained programmatically, that is, where one must produce a principal from a user name.

To obtain the correct user or group name, either pass the name *exactly* as it is stored in the authenticator or use the sequence of calls illustrated in the following code snippet:

```
import weblogic.security.principal.WLSGroupImpl;
import weblogic.security.principal.WLSUserImpl;
```

```

// Set the context
JpsContextFactory ctxFact = JpsContextFactory.getContextFactory();
ServerContextFactory scf = (ServerContextFactory) ctxFact;
JpsContext ctx = scf.getContext(ServerContextFactory.Scope.SYSTEM);
ctx = ctxFact.getContext();

// Set the identity store
IdentityStore identityStore =
ctx.getServiceInstance(IdentityStoreService.class).getIdmStore();

// In case of a user name, search the user that matches the supplied name
User user = idStore.searchUser(IdentityStore.SEARCH_BY_NAME, suppliedUserName);

// Use the obtained object (user) to obtain the stored user name and create
// the Principal
String storedUserName = user.getName();
Principal userPrincipal = new WLSUserImpl(storedUserName);

// Similarly, in case of a role name, search the role that matches
// the supplied role name
Role role = identityStore.searchRole(IdentityStore.SEARCH_BY_NAME,
suppliedRoleName);

// Use the obtained object (role) to obtain the stored role name and create
// the Principal
String storedRoleName = role.getName();
Principal rolePrincipal = new WLSGroupImpl(storedRoleName);

```

Important: When creating a user or role principal, you must use the calls:

```

Principal userPrincipal = new
WLSUserImpl(user.getUserProfile().getName());
Principal rolePrincipal = new
WLSGroupImpl(role.getRoleProfile().getName());

```

Instead of the calls:

```

Principal userPrincipal = new WLSUserImpl(user.getName());
Principal rolePrincipal = new WLSGroupImpl(role.getName());

```

L.5 Failure to Connect to an LDAP Server

This section explains the likely reasons why a connection to an Oracle Internet Directory LDAP server can fail. This failure can also happen during reassociation.

Symptom

The migration of data from a source repository to a target LDAP server repository fails.

Diagnosis

Typically, this kind of problem is due to an incorrect set up of parameters in the target LDAP server.

For further probing into Oracle WebLogic Server log files, search any of the log files in the directories `DomainName/servers/AdminServer` or `DomainName/servers/ManagedServers` for the following strings: `<Error>`, `<Critical>`, and `<Warning>`.

For more information about identifying and solving errors, see [Section L.1, "Diagnosing Security Errors."](#)

Solution

Verify that all the target server data provided for the migration is valid. You may require the assistance of your LDAP server administrator to perform this validation.

Note: If you are using Fusion Middleware Control to reassociate to an LDAP server, ensure that you use the button **Test LDAP Authorization** before initiating the operation. Typically, this test catches incorrect supplied parameters.

L.6 Failure to Connect to the Embedded LDAP Authenticator

This section explains the likely reasons why a connection to the embedded LDAP authenticator can fail.

Symptom

The connections that client applications use to request queries to the embedded LDAP authenticator, via the User and Role API, are stored and maintained in a connection pool. By default, and out-of-the-box, this pool is the JNDI pool, as specified in the file `jps-config.xml`.

If the number of current connections in the pool exceeds the maximum allowed by the LDAP service, client applications will not be able to connect to the service or, even when they are already connected, receive a "socket closed" exception. The server log would indicate, in this case, that the number of concurrent connections allowed has been exceeded.

Diagnosis

To avoid going over the limit, one needs to adjust the maximum number of concurrent connections allowed by the LDAP service as appropriate to the application's needs. This threshold needs to be finely tuned up: a too small maximum may not be sufficient (and cause the exception mentioned above); a too large maximum may risk a denial of service (DOS) attack. The correct maximum depends on your application and the particular LDAP service the application uses.

Solution

There are two alternative ways that resolve this issue:

- Increase the maximum number of concurrent connections allowed by the authenticator:
 - If the authenticator your application is using is the WebLogic Embedded LDAP authenticator, then edit the file `DomainName/servers/MyServerName/data/ldap/conf/vde.prop`, and increase the value of the property `vde.quota.max.conpersubject` from the default 100 to, for example, 200, or any other value.
 - Otherwise, if your application is using any other authenticator, consult the authenticator's documentation to learn how to modify the maximum.
- Edit the file `DomainName/config/fmwconfig/jps-config.xml` and remove the property `CONNECTION_POOL_CLASS` from the authenticator server instance (by

default, this property has the value
`oracle.security.idm.providers.stldap.JNDIPool.`

Note that (a) these settings do not exclude each other, that is, you can carry out both of them; and (b) in any case, you must restart the server for the changes to take effect.

L.7 User and Role API Failure

This section explains some reasons why you may fail to access data in a domain authenticator with the User and Role API.

Symptom

The User and Role API fails to access data in a configured authenticator.

Diagnosis 1

The OPSS User and Role API can access data *only* in the first LDAP authenticator configured in a domain. At least one such authenticator must be present in a domain. The API access to that first LDAP authenticator fails if the target user is not present in that authenticator, even though that user is present in some other domain authenticator.

Solution 1

Enter the missing user in the first LDAP authenticator, or reorder the list of LDAP authenticators in your domain.

Diagnosis 2

Let's assume that the target user on which the API that fails *is* present in the first LDAP authenticator configured in your domain.

By default, the User and Role API uses the attribute `uid` to perform user search in an LDAP authenticator. If for some reason, a user entered in the LDAP is lacking this attribute, then the User and Role API fails.

Solution 2

Ensure that all users in the first LDAP authenticator have the attribute `uid` set.

Note: If you are developing a Java SE application (and only in this case) and want the User and Role API to employ an attribute other than the default one (`uid`) to search users, say `mail` for example, then the properties `username.attr` and `user.login.attr` must be configured in the LDAP provider instance of the identity store (in the file `jps-config-jse.xml`) as illustrated in the following code snippet:

```
<serviceInstance provider="idstore.ldap.provider"
name="idstore.ldap">
  ...
  <property name="username.attr" value="mail"/>
  <property name="user.login.attr" value="mail"/>
  ...
</serviceInstance>
```

To add properties to a provider instance with a prescribed script, see [Section E.1, "Configuring OPSS Service Provider Instances with a WLST Script."](#)

L.8 Failure to Access Data in the Credential Store

This section explains a likely reason why an application fails to access data in the domain's credential store.

Symptom

An application fails to retrieve credential data from the domain's credential store, and an error message (containing lines like the one illustrated below) is logged (text in between brackets should describe information specific to the particular failure):

```
07/07/26 18:22:22 [JpsAuth] For permission ( CredentialAccessPermission [target]
[actions]), domain that failed: ProtectionDomain
cs(file:somePath/aWarFile.war/WEB-INF/classes/), []
```

Diagnosis

If an application is to access the credential store to perform an operation (such as retrieving a user password, for example), then its code must be granted the appropriate permission to perform the secured operation; otherwise, the application runs into an error like the one described above.

Solution

To grant the permission that an application requires to access the credential store, include the appropriate `CredentialAccessPermission` in the application's `jazn-data.xml`; this grant takes effect when the application is deployed or redeployed.

To include a permission using Fusion Middleware Control, see [Section 9.2, "Managing Policies with Fusion Middleware Control."](#)

To include a permission using an OPSS script, see [Section 9.3, "Managing Application Policies with OPSS Scripts."](#)

The following fragment of the file `jazn-data.xml` illustrates how to grant all code in the application `myApp` permission to read all credentials in the folder `myAlias`:

```
<jazn-data>
  <!-- codebase policy -->
```

```

    <jazn-policy>
      <grant>
        <grantee>
          <codesource>
            <!-- This grants applies to all code in the following directory -->
            <url>${domain.home}/tmp/_WL_user/myApp/-</url>
          </codesource>
        </grantee>
        <permissions>
          <permission>

<class>oracle.security.jps.service.credstore.CredentialAccessPermission</class>
<!-- Allow read permission to all credentials under folder MY_MAP -->
          <name>context=SYSTEM,mapName=MY_MAP,keyName=*</name>
          <actions>read</actions>
        </permission>
      </permissions>
    </grant>
  </jazn-policy>
</jazn-data>

```

L.9 Failure to Establish an Anonymous SSL Connection

This section explains the likely reasons why you are not able to establish an anonymous SSL connection while reassociating policies and credentials.

Symptom

A step in the reassociation of file-based policies and credentials to an LDAP-base storage using an Oracle Internet Directory server with Fusion Middleware Control involves testing the anonymous SSL connection to the LDAP server (specifically with the button Test LDAP). This test fails.

Diagnosis

Your target LDAP server must be trusted by the Oracle WebLogic Server and the port number you are using to the LDAP server must be an SSL port.

Solution

Establishing a connection to an LDAP server requires some previous configuration on the LDAP server. For details, see [Section 8.2.2, "Prerequisites to Using an LDAP-Based Security Store."](#)

In addition, to use an anonymous SSL connection, you must enter a port that has been set for receiving secure data. If your LDAP server has not been configured with such a port, the connection fails.

Ensure that the supplied LDAP server port is an SSL port configured to listen in anonymous SSL mode, and that the supplied server name is reachable. Typically, the setting of this port involves an LDAP server administrator.

L.10 Authorization Check Failure

This section explains a reason why an authorization check has failed.

Symptom

An attempt to authorize a user by your application fails, and the system logs an error containing a line like the following:

```
Servlet failed with Exception
oracle.adf.controller.security.AuthorizationException:ADFC-0619:
Authorization check failed: '/StartHere.jspx' 'VIEW'.
```

Diagnosis

One reason that can lead to such an authorization failure is a mismatch between the run-time policy context and the policy store stripe that your application is using.

On the one hand, the application stripe (or subset of policies in the policy store) that an application uses is specified in the file `web.xml` with the parameter `application.name` within the filter configuring the `JpsFilter` (for a servlet) or the interceptor configuring the `JpsInterceptor` (for an EJB). For details and samples, see [Application Name \(Stripe\)](#). If the application stripe is not specified (or left blank), then the system picks up an application stripe based on the application name.

On the other hand, the run-time policies that your application uses are specified in the file `system-jazn-data.xml` with the element `<application.name>`.

If those two names do not match or if you have not explicitly specified the stripe to use, then, most likely, your application is accessing the wrong policy stripe and, therefore, not able to authorize your application users as expected.

Solution

Ensure that you specify explicitly your application stripe, and that stripe is the one that your application is supposed to use. In most cases, the two names specified in those two different files (as explained above) match; however, in cases where several applications share the same policy stripe, they may differ.

L.11 User Gets Unexpected Permissions

This section explains the likely reasons why a user gets permissions other than those anticipated.

Symptom

A new user or a modified user gets unexpected permissions.

Diagnosis

This issue is likely to come up in cases where a user is added with the name of a previously removed user, or an old user gets its name or uid changed. The common reason why the user may get more or less permissions than expected is that the policy store has not been properly updated before a user is removed or a user's data is changed.

Solution

Before deleting a user, revoke all permissions, application roles, and enterprise groups that had been granted to the user. If you fail to remove all security artifacts referencing a user to be deleted, they are left dangling and, potentially, inherited if another user with the same name or uid is created at a later time.

Similar considerations apply to when a user name or uid is changed: all policies (grants, permissions, roles) referring to the old data must be updated so that they work as expected with the new data.

L.12 Security Access Control Exception

This section explains a reason why your code may run into a security access control exception.

Symptom

At run time, your application outputs an error like the following one (only the first few lines are shown):

```
<Jan 20, 2009 5:45:33 PM PST> <Error> <HTTP> <BEA-101020>
<[weblogic.servlet.internal.WebAppServletContext@140cf52
- appName: 'Application2',
name: 'Application2.war',
context-path: '/Application2',
spec-version: '2.5']
Servlet failed with
Exceptionjava.lang.RuntimeException:java.security.AccessControlException:access
denied
...

```

Diagnosis

The above error means that a call in your code does not have sufficient permissions to execute a secured operation.

Solution

Your code must be granted the appropriate permissions to execute the secured operation. Depending on the scope of the permission you would like to set, you have two alternatives.

The first one is to grant permission to all application code in the application's EAR or WAR files; in this case, the call to the secured operation can be inserted anywhere in the application code.

The second one is to grant permission to just a JAR file; in this case, the call to the secured operation must be inside a privileged block.

Each of these solutions is next illustrated by an application attempting to access the credential store.

The following fragment of an application `jazn-data.xml` illustrates how to set permission to read any key within the map `MY_MAP` in the credential store to *any* code within the directory `BasicAuth`:

```
<jazn-policy>
  <grant>
    <grantee>
      <codesource>
        <url>file:${domain.home}/servers/_WL_user/BasicAuth/-</url>
      </codesource>
    </grantee>
    <permissions>
      <permission>
        <class>
          oracle.security.jps.service.credstore.CredentialAccessPermission
        </class>
        <name>context=SYSTEM,mapName=MY_MAP,keyName=*</name>
        <actions>read</actions>
      </permission>
    </permissions>
  </grant>

```

```
</jazzn-policy>
```

If the permission is to be granted to the code in a particular EAR or WAR file, the `url` specification above would have to be changed to one like the following:

```
<url>file:${domain.home}/servers/_WL_user/jpsBasicAuth/.../BasicAuth.ear</url>
```

In both above cases, the call to read the credential store can be placed anywhere in the application code.

If, however, the permission is to be granted to just the code in a particular JAR file, the `url` specification above would have to be changed to one like the following:

```
<url>file:${domain.home}/servers/_WL_user/jpsBasicAuth/myJars/Foo.jar</url>
```

In this last case, the code in the file `Foo.jar` that calls a read operation on the credential store must be placed in an `AccessController.doPrivileged` block, as illustrated in the following code snippet:

```
import oracle.security.jps.*;
import oracle.security.jps.service.credstore.*;

JpsContextFactory factory = JpsContextFactory.getContextFactory();
JpsContext jpsContext = factory.getContext();
final CredentialStore store =
jpsContext.getServiceInstance(CredentialStore.class);
Credential cred = AccessController.doPrivileged(new
PrivilegedExceptionAction<PasswordCredential>() {
    public PasswordCredential run() throws JpsException {
        return store.getCredential("MY_MAP", "anyKey");
    }
});

PasswordCredential pwdCred = (PasswordCredential)cred;
...
```

Note that since our sample grant above allows only read permission, none of the set or reset operations work, even inside a privileged block.

L.13 Runtime Permission Check Failure

This section explains a reason why a permission may fail to pass a permission check.

Symptom

At run time, your application outputs an error like the following one (only the first few lines are shown):

```
[JpsAuth] Check Permission
PolicyContext:      [null]
Resource/Target:   [test]
Action:            [null]
Permission Class:  [com.oracle.permission.SimplePermission]
Evaluator:         [ACC]
Result:            [FAILED]
Failed

ProtectionDomain:ClassLoader=weblogic.utils.classloaders.ChangeAwareClassLoader@14
061a8
finder: weblogic.utils.classloaders.CodeGenClassFinder@2dce7a8
annotation: Application2@Application2.war
CodeSource=file:/scratch/servers/AdminServer/tmp/permission/TestServlet$1.class
```

```

Principals=total 0 of principals<no principals>
Permissions=(
(oracle.security.jps.service.credstore.CredentialAccessPermission
context=SYSTEM,mapName=default,keyName=* read,write)
(java.net.SocketPermission localhost:1024- listen,resolve)
(oracle.security.jps.service.policystore.PolicyStoreAccessPermission
context=APPLICATION,name=* getApplicationPolicy)
(oracle.security.jps.service.policystore.PolicyStoreAccessPermission
context=SYSTEM getConfiguredApplications)
(com.oracle.permission.SimplePermission *)
...
java.security.AccessControlException: access denied
(com.oracle.permission.SimplePermission test)...

```

Diagnosis

When two or more applications share a permission class, that permission class must be set in the system class path so the class is loaded just once. Otherwise, only the first application loading the class passes the permission check; other ones loading the same class thereafter may fail the permission check and output an error like the one illustrated above.

Note that even though the permission class is in the permission collection (see bold text in sample output above), the check fails and the access is denied. This is because, at that point, the environment contains *several* instances of a permission class with the same name.

Solution

Ensure that if two or more applications to be run in the same domain share a permission class, then include that class in the system class path.

L.14 Permission Failure Before Server Starts

This section describes a reason why a permission check may fail before the server has completed its starting phase.

Symptom

An authorization check fails before the server has started. The server has completed its startup when it outputs the a line like the following:

```
<WebLogicServer> <BEA-000365> <Server state changed to STARTING>
```

Diagnosis

A permission check error before the server has changed status to `STARTING` usually indicates that the authorization service required to check that permission was not fully initialized at the time of the request.

Solution

To workaround this issue, proceed as follows:

1. Edit the file `weblogic.policy` to add the appropriate grant(s).
2. Start the Oracle WebLogic Server with the following two system properties set:
 - `java.security.policy` set to the location of the `weblogic.policy` file.
 - `jps.policystore.hybrid.mode` set to `true`.

L.15 Policy Migration Failure

This section describes a reason why the automatic migration of policies at application deployment may fail. Note that the deployment of an application may succeed even though the migration of policies failed.

Note: The reason why the automatic migration can fail, as explained in this section, can also lead to similar failures when reassociating domain stores.

For a failure also related to migration, see [Incompatible Versions of Policy Stores](#).

Symptom

The application is configured to migrate policies automatically at deployment. The application deployment succeeds, but the diagnostic file corresponding to the server where it has been deployed outputs a message like the following:

```
[2009-01-21T13:34:48.144-08:00] [server_soa] [NOTIFICATION] []
[oracle.jps.deployment] [tid: [ACTIVE].ExecuteThread: '2' for queue:
'weblogic.kernel.Default (self-tuning)'] [userId: weblogic]
[ecid: 0000Hvv7U_H7q2T6uBADUH19Tq0B00002I,0] [APP: JpsJdev#V2.0]
Application [JpsJdev#V2.0] is being deployed, start policy migration.
```

```
[2009-01-21T13:34:48.770-08:00] [server_soa] [WARNING] []
[oracle.jps.deployment] [tid: [ACTIVE].ExecuteThread: '2' for queue:
'weblogic.kernel.Default (self-tuning)'] [userId: weblogic]
[ecid: 0000Hvv7U_H7q2T6uBADUH19Tq0B00002I,0] [APP: JpsJdev#V2.0]
```

```
Exception in application policy migration.[[
oracle.security.jps.JpsException: application Role:
test_role not found for the application in the destination policy store
at oracle.utility.destination.apibased.JpsDstPolicy.convertAppPolicyPrincipal
(JpsDstPolicy.java:815)
at oracle.utility.destination.apibased.JpsDstPolicy.clone
(JpsDstPolicy.java:691)...
```

The above excerpt was extracted from the file `server_soa-diagnostic.log`, and the application `JpsJdev` was deployed to the managed server `server_soa`. Note that the key phrase to look for to locate such error is highlighted in the sample above. In addition, the error describes the artifact that raised the exception, the application role `test_role`.

Diagnosis

Something is wrong with the definition of this role in the application file `jazn-data.xml`. In fact, a quick look at this file reveals that the role `test_role` is referenced in a grantee, as illustrated in the following excerpt:

```
<grantee>
  <display-name>myPolicy</display-name>
  <principals>
    <principal>
      <class>oracle.security.jps.service.policystore.ApplicationRole</class>
      <name>test_role</name>
    </principal>
  </principals>
</grantee> ...
```


But the name of what is supposed to be the application role named `test_role`, however, was inadvertently misspelled to `test_rolle`:

```
<application>
  <name>JpsJdev</name>
  <app-roles>
    <app-role>
      <name>test_rolle</name>
      <class>oracle.security.jps.service.policystore.ApplicationRole</class>
      <members> ...
```

Solution

Ensure that all application roles referenced in application policies have been properly defined in the `jazn-data.xml` file. If a referenced role name cannot be matched, as in the samples above, the migration fails.

L.16 Characters in Policies

This section explains several issues related to characters used in policies, in the following sections:

- [Use of Special Characters in Oracle Internet Directory 10.1.4.3](#)
- [XML Policy Store that Contains Certain Characters](#)
- [Characters in Application Role Names](#)
- [Missing Newline Characters in XML Policy Store](#)

L.16.1 Use of Special Characters in Oracle Internet Directory 10.1.4.3

When the policy store is an LDAP-based Oracle Internet Directory 10.1.4.3 repository, then using the characters `*`, `(`, `)`, or `\` in the RFC 2252/2253 filter results in error 53 (DSA unwilling to perform). To resolve this error, apply the patch for bug number 7711351 to Oracle Internet Directory 10.1.4.3.

L.16.2 XML Policy Store that Contains Certain Characters

The issue explained in this section is relevant to XML Policy Stores only, that is, it does not apply to LDAP-based Policy Stores.

The following characters:

```
< " & $ ? * , / \ ` : ( ) ^ ' % + { }
```

are not recommended as part of an Application Role name when using an file-based policy store.

If it becomes necessary to use one of those characters to create a role, for example, then ensure that such characters are escaped in the input to API Policy Store methods like `ApplicationPolicy.searchAppRoles()`, so they return correct results.

For example, if you have an application role named `"appRole^$"` it will need to be input as `ApplicationPolicy.searchAppRoles("appRole\\^\\$")` to find the match in the policy store.

Alternatively, you could use a wild card in the search expression without including these escaped special characters, and it will also match that application role:

```
ApplicationPolicy.searchAppRoles("appRole*")
```

L.16.3 Characters in Application Role Names

An application role name is a string of printable characters other than white space, that is, it can contain alpha-numeric characters (ASCII or Unicode) and other printable characters (such as underscore or square brackets) *except* for white space. This rule applies to all three kinds of supported storage: XML, LDAP, and DB.

L.16.4 Missing Newline Characters in XML Policy Store

In an file-based policy store, a new-line character is required between the closing of a `<permission>` or `<principal>` tag and the opening of the following one.

Following are examples of strings illustrating incorrect and correct formats.

Incorrect example fragment of policy store:

```
<permission>
  <class>java.lang.RuntimePermission</class>
  <name>getClassLoader</name>
</permission> <permission>
  <class>java.io.FilePermission</class>
  <name>/foo</name>
  <actions>read,write</actions>
</permission>
```

Correct example fragment of policy store:

```
<permission>
  <class>java.lang.RuntimePermission</class>
  <name>getClassLoader</name>
</permission>
<permission>
  <class>java.io.FilePermission</class>
  <name>/foo</name>
  <actions>read,write</actions>
</permission>
```

L.17 Granting Permissions in Java SE Applications

This section describes the correct way to code a grant in Java SE applications. Even though the problem described is not an issue in Java EE applications, for maximum portability, it is recommended that this solution be used in Java EE applications too.

Symptom

The application code includes a fragment like the following, by an application creates a grant:

```
Permission p = new FilePermission(resourceName, "write");
PrincipalEntry myPrincipal2 =
InfoFactory.newPrincipalEntry("weblogic.security.principal.WLSGroupImpl",
enterpriseRoleName2);
ap.grant(new Principal[]{myPrincipal2.getPrincipal()}, null, new Permission[]{p});
```

At runtime, however, the grant is not taking effect as expected.

Diagnosis

A bit of inspection indicates that the policy store repository includes the following attribute:

```
orcljaznjavaclass=oracle.security.jps.internal.policystore.UnresolvedPrincipal+cn=enterpriseRoleName2
```

Solution

The lines of code above should be replaced by the following:

```
Permission p = new FilePermission (resourceName, "write");
PermissionEntry permEntry = InfoFactory.newPermissionEntry(p.getClassName(),
p.getName(), p.getActions());
ap.grant (new PrincipalEntry[] {myPrincipal2}, null, new PermissionEntry[]
{permEntry});
```

The solution uses the array `PrincipalEntry` instead of the array `Principal` and the array `PermissionEntry` instead of the array `Permission`.

Note: This same issue applies to the method `revoke`, which also has overloaded variants that accept `Principal []` or `PrincipalEntry []`

L.18 Troubleshooting Oracle Business Intelligence Reporting

This section describes common problems and solutions for Oracle Business Intelligence when used as a reporting tool for Oracle Fusion Middleware security. It contains the following topics:

- [Audit Templates for Oracle Business Intelligence Publisher](#)
- [Oracle Business Intelligence Publisher Time Zone](#)

L.18.1 Audit Templates for Oracle Business Intelligence Publisher

To view Oracle Fusion Middleware Audit Framework reports in Oracle Business Intelligence, you must use the appropriate audit templates.

For details, see [Section 14.1.3, "Set Up Oracle Reports in Oracle Business Intelligence Publisher"](#).

L.18.2 Oracle Business Intelligence Publisher Time Zone

You may see problems with Oracle Fusion Middleware Audit Framework reports if Oracle Business Intelligence Publisher and the database are installed in sites with different time zones.

To avoid this issue, ensure that Oracle Business Intelligence Publisher and the database are installed in the same time zone.

L.19 Search Failure when Matching Attribute in Policy Store

This section describes a reason why cataloging of an attribute is needed.

Symptom

While searching the policy store, an exception similar to the following is encountered:

```
oracle.security.jps.service.policystore.PolicyStoreOperationNotAllowedException
javax.naming.OperationNotSupportedException:
[LDAP: error code 53 - Function Not Implemented, search filter attribute
orcljpsresourcetyename is not indexed/cataloged];
```

```
remaining name 'cn=Permissions,cn=JAASPolicy,cn=IDCCS, cn=sprint6_policy_
domain,cn=JPSTContext,cn=FusionAppsPolicies'
```

Diagnosis

The error above indicates that the attribute `orcljpsresourcetypername` must be cataloged before it is used in a filter to search the policy store.

Solution

An Oracle Internet Directory attribute used in a search filter must be indexed and cataloged. Indexing and cataloging are optional operations, in general, but required for OPSS-related attributes. Attribute indexing and cataloging is automatically performed by the OPSS script `reassociateSecurityStore`.

For details about managing attribute catalogs and identifying whether an attribute is indexed, see the following sections in *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*:

To catalog attributes manually use the command `ldapmodify`, as illustrated below:

```
>ldapmodify -h <host> -p <port> -D <bind DN> -w <bind password> -v -f <catalogue
modify ldif file name>
```

To catalog, for example, the attributes `createtimestamp` and `modifytimestamp` use an LDIF file like the following:

```
dn: cn=catalogs
changetype: modify
add: orclindexedattribute
orclindexedattribute: modifytimestamp
orclindexedattribute: createtimestamp
```

The list of Oracle Internet Directory attributes that must be indexed follows:

```
OrclJpsAllResourcKeyword
OrclJpsAllResourceActionKeyword
OrclJpsEncodedAttributes
OrclJpsExtensionType
OrclJpsResourceConverter
OrclJpsResourceMatchingAlg
OrclJpsResourceMatchingAlgorithm
OrclJpsResourceNameExpression
OrclJpsRoleType
orcOesAppAttributes
orclASInstanceName
orclFarmName
orclJPSObjGUID
orclJavaApplicationEntityRef
orclJpsPolicyDomainName
orclJpsResourceActionsetMembers
orclJpsResourceExpression
orclJpsResourceLocalityRef
orclJpsResourceMatcherJavaclass
orclJpsResourceName
orclJpsResourceTypeActionAttrs
orclJpsResourceTypeActionNames
orclJpsResourceTypeName
orclJpsResourceTypeProviderName
orclJpsResourceTypeResourceAttrs
orclJpsRoleCategory
orclJpsRoleMemberExpression
```

orclJpsSuperResourceType
orclOESActCollectionName
orclOESActCollectionRfs
orclOESActionAttributes
orclOESActionConstraint
orclOESAlgorithmJavaClass
orclOESAllResourceType
orclOESAllowAdviceRef
orclOESAllowObligationRef
orclOESAttributeCategory
orclOESAttributeCollectionHandlerFunctionName
orclOESAttributeCollectionHandlerPackageName
orclOESAttributeCollectionHandlerSchemaName
orclOESAttributeCollectionName
orclOESAttributeDataType
orclOESAttributeIssuer
orclOESAttributeNamespace
orclOESAttributeType
orclOESCombinerParameter
orclOESConditionExpression
orclOESDSColumnAttrs
orclOESDSPrimKey
orclOESDataSourceCtrnt
orclOESDataSourceName
orclOESDataSourceType
orclOESDefaultPolSetRef
orclOESDenyAdviceRef
orclOESDenyObligationRef
orclOESDistributionEndTime
orclOESDistributionID
orclOESDistributionIssuer
orclOESDistributionMessage
orclOESDistributionPercentComplete
orclOESDistributionStartTime
orclOESEffect
orclOESEnvAttributes
orclOESEnvConstraint
orclOESEExecutionFrequency
orclOESEExpression
orclOESFunctionCategory
orclOESFunctionClass
orclOESFunctionParameters
orclOESFunctionReturnType
orclOESIsSensitive
orclOESIsSingleValued
orclOESMatchInfo
orclOESMaxDelegationDepth
orclOESObligationFulfillOn
orclOESPDPAddress
orclOESPDPConfigurationID
orclOESPDPInstanceName
orclOESPDPStatusSuccess
orclOESPIPTYPE
orclOESPolicyCategory
orclOESPolicyCombinerParameter
orclOESPolicyCombiningAlgorithmRef
orclOESPolicyDefaults
orclOESPolicyExtension
orclOESPolicyIssuer
orclOESPolicyRef

orclOESPolicyRuleOrder
orclOESPolicyRuleRef
orclOESPolicySetCategory
orclOESPolicySetDefaults
orclOESPolicySetRef
orclOESPolicySetType
orclOESPolicyType
orclOESPolicyVersion
orclOESPresenceRequired
orclOESPrincConstraint
orclOESPrincipalAttributes
orclOESResConstraint
orclOESResTypeCategory
orclOESResourceAttributes
orclOESResourceHirchyType
orclOESResourceNameDelim
orclOESResourceParentName
orclOESRoleMapping
orclOESRuleCombinerParameter
orclOESRuleCombiningAlgorithmRef
orclOESSQLExpression
orclOESSetCombinerParameter
orclOESSetMemberOrder
orclOESTargetExpression
orclOESXMLExpression
orclassignedpermissions
orclassignedroles
orcldistributionversion
orcljazncodebase
orcljaznjavaclass
orcljaznpermissionactions
orcljaznpermissionresourceref
orcljaznpermissionsigner
orcljaznpermissiontarget
orcljaznpermissiontargetexpr
orcljaznprincipal
orcljaznsigner
orcljpsRuleCombiningAlgorithmRef
orcljpsactionsdelim
orcljpsassignee
orclrolescope

L.20 Search Failure with an Unknown Host Exception

When searching for information in an Active Directory environment that is configured for LDAP referrals, the referrals fail if the host being referred to is in a different domain than the Active Directory server.

Symptom

When a user requests a resource, at times verification of the user's identity can fail due to an inability to validate the user's identity in the directory. This error can occur in an Active Directory environment when the user's browser runs on a non-Windows computer, or if the user's browser runs on a Windows computer that is not in the Active Directory server domain.

Diagnosis

This problem can arise due to LDAP referral chasing. An LDAP referral occurs when a domain controller does not have the section of the directory tree where a requested

object resides. The domain controller refers the client to another destination so that the client can conduct a DNS search for another domain controller. If the client is configured to chase referrals, the search can continue.

For the scenario where the user has a Windows-based computer, an issue can occur with LDAP referrals if the client's domain controller does not have a trust relationship with the Active Directory domain controller.

Solution

If you encounter this issue, add the entry for the Active Directory host's address in the following list:

```
WINDOWS_HOME_DIRECTORY\system32\drivers\etc\hosts
```

On Windows XP, the list is located here:

```
C:\WINDOWS\system32\drivers\etc\host
```

On a Unix-based system, add this entry to the `/etc/hosts` file, using the format:

```
IP_address_of_AD_host AD_host_name
```

where `AD_host_name` is the host name specified in the referral, for example:

```
123.123.123.123 my2003ad.com
```

L.21 Incompatible Versions of Binaries and Policy Store

This section describes the reason why the server would throw the exception `PolicyStoreIncompatibleVersionException`. See also [Incompatible Versions of Policy Stores](#).

Symptom

An error similar to the following is logged or issued by the server:

```
Oracle.security.jps.service.policystore. PolicyStoreIncompatibleVersionException
JPS-06100: Policy Store version 11.1.1.5.0 and Oracle Platform Security Services
version 11.1.1.4.0 are not compatible.
```

Diagnosis

The above exception indicates that the domain OPSS binaries version (11.1.1.4.0) and the policy store version (11.1.1.5.0) used by that domain have incompatible versions. The version of the policy store is established during reassociation and that version is used until the policy store is upgraded to a newer version.

OPSS domain binary versions are *backward* compatible with policy store versions used by that domain, but they are not forward compatible. Thus, the error above indicates that the policy store has version newer than the version of the OPSS binaries. PS3 OPSS binaries cannot use a newer version of the policy store.

Here are three scenarios where OPSS binaries ends up running into this incompatibility.

- Scenario 1
 - Domain1 and Domain2 point to the same policy store; Domain1, Domain2, and that policy store are all three version PS2.
 - The binaries in Domain1 are upgraded to PS3.
 - The policy store is upgraded to PS3 (using the command `upgradeOPSS`).

- When the Domain2 is brought up again, its version and the policy store version are incompatible.
- Scenario 2
 - Domain1 points to a policy store and both are version PS2.
 - An attempt to migrate the policy store to a PS3 policy store fails because the migration would render a scenario with incompatible versions.
Migration is supported only when the versions of the OPSS binaries and the policy store are same.
- Scenario 3
 - A PS2 Domain1 attempts to join a PS3 policy store (in some other domain), using the command `reassociateSecurityStore` with the join argument.
 - The operation fails because the sharing would render a scenario with incompatible versions.
Reassociation is supported only when the versions of the OPSS binaries and the policy store are same.

Solution

The solution, common to all three scenarios above, is either one of the following:

- Update the domain OPSS binaries to match the version of the policy store the domain is pointing to.
- Reassociate the domain policy store to a policy store that has version not older than the version of the domain OPSS binaries.

L.22 Incompatible Versions of Policy Stores

This section describes the reason why, while migrating the OPSS security store, the exception `PolicyStoreIncompatibleVersionException` is encountered. See also [Incompatible Versions of Binaries and Policy Store](#).

The above exception indicates that the version of the source store is *higher* than the version of the target store, an invalid combination of versions. Migration proceeds only if the version of the source is not higher than the version of the target.

The workaround is to upgrade the target store to a version compatible with the version of the source store.

L.23 Need Further Help?

You can find more solutions on My Oracle Support (formerly MetaLink) at <http://myoraclesupport.oracle.com>. If you do not find a solution to your problem, log a service request.

A

- AbstractTypedPermission, 20-14
- access control list, 8-14
- Access Server
 - cache, 17-62
- AccessGate
 - configureAccessGate tool, 17-41, 17-72
- ACL, 8-14
- add.application.roles, 21-4
- add.authenticated.role, 21-6
- addBootStrapCredential, 10-9
- addPrincipalsToAppRole, 20-4
- administration tools, 5-1
- administrative tasks, 5-5
- administrators group, 2-8
- Anonymous and Authenticated Roles
 - Properties, F-23
- anonymous role, 2-7, 2-8, 5-2
- anonymous role and authentication, 2-8
- anonymous SSL, 8-12
- anonymous user, 2-1, 2-7, 2-8
- anonymous user and role, 21-4
- app.context, 8-22
- Application Name or Stripe, 21-2
- application policy, 2-2
- application role, 2-2, 21-4
- application role hierarchy, 9-11
- application stripe, 21-2, 21-3
- application.name, 21-2, 21-3
- ApplicationRole class, 2-6
- application-specific policies and roles, 3-4
- audit data
 - bus-stop files, 13-8
 - file management, C-51
 - migrating, 13-25
 - reports, 14-1
- audit data store
 - backup and recovery, 13-27
 - configuring for Java components, 13-5
 - configuring for system components, 13-6
 - data purge, 13-27
 - de-configuring, 13-8
 - partitioning, 13-25
 - schema, 13-22
 - tiered archival, 13-27
- Audit Flow, 12-6
- audit logs, 13-21
- audit policies migration, 6-19
- audit policy, 13-11
 - event filters, 12-8
- audit report
 - example of, 14-8
- audit reports
 - attributes, 14-13
 - by component, C-29
 - custom, 14-15
 - list of standard, 14-11
 - types of, 14-6
 - viewing, 14-7
- Audit Schema, C-31
- audit service, 28-1
- audit-aware components, C-1
- auditing
 - event attributes, C-24
 - events, C-2
 - filter expression syntax, C-50
 - for Oracle Fusion Middleware components, 13-11
 - in Oracle Fusion Middleware, 12-1
 - Java components, C-1
 - manual policy management, 13-19
 - manually configure for Java components, 13-20
 - manually configure for system components, 13-20
 - Oracle Directory Integration Platform, C-2
 - Oracle HTTP Server, C-8
 - Oracle Identity Federation, C-11
 - Oracle Internet Directory, C-9
 - Oracle Platform Security Services, C-6
 - Oracle Virtual Directory, C-16
 - Oracle Web Cache, C-21
 - Oracle Web Services Manager, C-24
 - overview, 12-3
 - OWSM-Agent, C-18
 - OWSM-PM-EJB, C-19
 - policy management with Fusion Middleware Control, 13-11, 13-14
 - policy management with WLST, 13-17
 - record storage, 12-8
 - report filters, 14-4
 - report setup for Oracle Business Intelligence Publisher, 14-3

- report templates, 14-4
- Reports Server, C-2
- system components, C-2
- WLST commands, C-44
- WS-Policy Attachment, C-21
- Authenticated Role, 21-6
- authenticated role, 2-7, 5-2, 21-6
- authenticated user, 2-1
- Authentication providers, 18-12
 - DefaultAuthenticator, 16-23, 16-30, 16-36, 17-45, 17-55, 17-64, 18-12
 - LDAP Authentication, 16-19, 17-42
 - OAM, 15-4, 15-5
 - OAM Authenticator, 16-30, 17-55
 - OAM Identity Asserter, 16-23, 16-36, 17-45, 17-64
 - OID Authenticator, 16-23, 16-36, 17-45, 17-64, 18-4, 18-12
 - OSSO Identity Asserter, 18-12
 - WebLogic, 15-1
- authenticator flags, 3-3
- Authenticator for OAM, 15-5
- Authorization failure, 20-7
- authorization failure, 9-2
- Auto login, 8-19
- autologin.url, 8-22

B

- backup, 5-3
- basic security tasks, 5-2
- bootstrap credentials, 6-6, 23-3
- Bulk authorization, 23-5
- bulkload, 6-19

C

- cache
 - Access Server, 17-62
 - refresh frequency, 9-29
- cache refresh, 9-28
- caching, 9-28
- Cascading deletions, 23-5
- characters allowed in policies, L-35
- characters in security artifacts, 9-2
- checkBulkAuthorization, 20-12
- checkPermission, 20-7, 20-8, L-3
- choosing
 - the right SSO solution, 15-1
- class path, 1-7, 3-5, 8-1, 9-3, 9-11, 21-22, E-4
- class permission, 21-21
 - CredentialAccessPermission, 21-23
 - JpsPermission, 21-23
 - PolicyStoreAccessPermission, 21-22
- cloning environments, 5-3
- commands to administer credentials, 9-8, 10-6
- Complex queries, 23-5
- Compliance, 12-1
- configuration file, 21-25
- configuration of multiple authenticators, 3-3
- configureAccessGate tool, 17-41, 17-72

- configuring
 - global logout
 - Oracle Access Manager, 17-10
 - Identity Assertion
 - for single sign-on with OAM, 16-16, 17-35
 - Oracle Web Services Manager, 16-35, 17-60
 - OAM Authenticator, 17-49
 - OAM for single-sign on with OAMCfgTool, 17-39
 - OAM for SSO with OAMCfgTool, 17-39
 - OSSO, 18-1
 - providers for Oracle Web Services Manager, 16-36, 17-63
 - Single Sign-On in Oracle Fusion Middleware, 15-1, 16-1, 17-1, 18-1
- configuring domains, 5-5
- configuring resource permissions, 20-13
- configuring WebLogic domains, 5-5
- CONNECTION_POOL_CLASS, L-26
- createAppRole, 9-9, 9-10
- createCred, 10-8
- createResourceType, 9-16
- creating user accounts, 2-9
- credential migration settings, 6-5
- credential store, 2-3
- Credential Store Types, 3-5
- CredentialAccessPermission, 21-23
- CredentialMapping permission, 8-23
- CSF
 - J2EE example with LDAP store, 24-13
 - J2EE example with wallet, 24-11
 - J2SE example with wallet, 24-8
- CSIv2 identity assertion, 3-4
- custom authorization providers, 3-4
- cwallet.sso, 4-4, 6-3, 21-1, 21-19
- cwallet.sso file, 21-10

D

- DB-based credential store, 3-5
- DB-based policy store, 8-6
- DB-based security store, 4-1
- DBMS_STATS, 8-8
- debugging authorization, L-9
- DefaultAuthenticator, 16-23, 16-30, 16-36, 17-45, 17-55, 17-64, 18-12
- default.auth.level, 8-23
- deleteAppPolicies, 9-15
- deleteAppRole, 9-10
- deleteResourceType, 9-17
- deleting a role, 9-10
- deployed application, 5-4
- deploying applications, 6-2
- deploying JavaEE applications, 6-7
- deploying to a test environment, 6-6
- deployment tools, 6-2
- development mode, 21-20, 21-21
- distribute environments, 8-3
- DN, 2-10
- doAs, 20-11
- doAsPrivileged, 20-11

Dynamic authentication, 8-19

E

EAR file, 21-9, 21-10
EJB Interceptor, 21-1
ejb-jar.xml, 3-4, 21-1
ejb-jar.xml., 21-9
embedded LDAP, 3-2, 4-2
enable.anonymous, 21-4
enterprise group, 2-1
Enterprise Groups and Users Class, 21-9
enterprise user, 2-1
Enterprise-Level SSO, 15-2
entitlement-based policies, 2-2
Event Source Type, 12-8
Existing OSSO, 15-2
exportAuditConfig, C-49
EXTRA_JAVA_PROPERTIES, F-1, L-7

F

fail over support, 5-5
FAQ, 1-2
file-based policy store, 3-5
file-based security store, 4-1

G

generic credential, 10-1
Generic LDAP Properties, F-21
getAuditPolicy, C-45
getGrantedResources, 20-12
getNonJavaEEAuditMBeanName, C-45
getPermissions, L-4
getResourceType, 9-17, 9-18, 9-19, 9-20, 9-21, 9-22, 9-23, 9-24, 9-25
Global logout, 8-19
grant
 permission-based, 2-5
grantAppRole, 9-11
GrantManager class, 20-5
grantPermission, 9-13
group, 2-1
GUID, 2-10

H

Headers
 sent by Oracle HTTP Server, 18-4
host name verification, 3-3
hot deployed, 6-10

I

Identity Asserter for Single Sign-on with OAM, 15-5
identity store, 2-3
 creating provider, 25-7
 provider configuration properties, 25-8
 selecting provider, 25-6
 WebLogic, 3-2

 WebSphere, 3-4
identity store in JavaSE, 22-2
Identity Store Service, 7-1
identity store types, 3-2
identity virtualization, 7-1
idstore.type, F-16
importAuditConfig, C-49
incompatible versions, L-41, L-42
initializing an LDAP authenticator, 3-3
invoking MBeans, E-3
isCallerInRole, 1-6
isUserInRole, 1-6, 20-3

J

JAAS mode, 21-6
Java component, 2-4
javadocs
 OPSS APIs, H-1
 OPSS MBeans APIs, H-1
 OPSS User and Role APIs, H-1
JavaSE application, 23-1
java.security.policy, F-2
jazn-data.xml, 4-4, 6-3, 21-1, 21-9, 21-10
join, 9-27
JpsApplicationLifecycleListener, 21-20
jpsApplicationLifecycleListener, 21-13
jps.apppolicy.idstoreartifact.migration, 21-13
jps.auth.debug, L-7
jps.auth.debug.verbose, L-8
jps-config-jse.xml, 1-7
jps-config.xml, 21-1, A-1
jps-config.xml example, 21-25
jps-config.xml full example, 21-25
jps.credstore.migration, 21-20
jps.deployment.handler.disabled, 8-15, 21-11
JpsFilter, 21-1, 21-9, L-5
JpsInterceptor, 21-1, 21-7, 21-9, L-5
JpsPermission, 21-23
jps.policystore.applicationid, 21-12
jps.policystore.hybrid.mode, F-3
jps.policystore.migration, 21-12
jps.policystore.migration.validate.principal, 21-15
jps.policystore.removal, 21-14

K

Keys and Certificates
 managing, 11-1
Keystore Service, 11-1, 27-1
 commands, 11-3

L

large volume stores, 6-18
LDAP Credential Store Properties, F-14
LDAP Identity Store Properties, F-15
LDAP Policy Store Properties, F-4
LDAP servers, 4-1
ldapadd, 8-4
LDAP-based policy store, 3-5, 8-2

- ldapmodify, 8-14
- ldapsearch, 8-4
- LDIF file, 8-3
- ldifwrite, 6-18
- listAppRoleMembers, 9-12
- listAppRoles, 9-12
- listAuditEvents, C-48
- listPermissions, 9-15
- loggers
 - oracle.security.jps.trace.logger, L-4
 - oracle.security.jps.util.JpsAuth, L-3
- logical role, 2-2, E-11
- login.url.FORM, 8-22
- logout.url, 8-22

M

- management tools, 4-3
- managing
 - keys and certificates, 4-3
 - policies and credentials, 4-3
- managing credentials, 6-6, 6-7
- managing domain authenticators, 5-5
- managing identities, 4-3, 6-6
- managing policies, 6-6
- managing policies and credentials, 4-3
- managing system policies, 6-7
- managing users and groups, 4-2
- Manually Configuring
 - WebGate Web Server, 16-12
- mapping application roles to enterprise groups, 6-7
- mapping of application roles, 2-4
- mapping roles, 6-11
- matcher class, 20-14
- Matcher Class for a Resource Type, 20-14
- MBean
 - Administration Policy Store, E-3
 - annotations, E-11
 - Application Policy Store, E-3
 - code sample, E-4
 - Credential Store, E-3
 - Global Policy Store, E-3
 - Jps Configuration, E-3
- migrateSecurityStore, 6-9, 6-11, 8-16, 21-24, I-2
 - DB to DB, 6-15, 6-18
 - LDAP to LDAP, 6-14, 6-17
 - XML to LDAP, 6-15, 6-18
- migrating credentials example, 6-15
- Migrating Identities, 21-24
- migrating identities manually, 6-9
- migrating large stores, 6-18
- migrating other providers, 6-9
- migrating policies and credentials at deployment, 6-10
- migrating policies example, 6-11
- Migration of credentials, 3-6
- Migration of policies, 3-5
- mod_osso, 18-5, 18-22
- modifyBootStrapCredential, 10-9
- modifying a resource type, 9-17

- Monitoring, 12-2
- multiple-node server domain, 8-3

N

- name comparison logic, 2-10

O

- OAM
 - Authentication provider, 15-4, 15-5
 - parameter, 17-14
 - Troubleshooting, 17-68
 - Authenticator, 15-5, 16-30, 17-55
 - Identity Asserter, 15-5, 16-23, 16-36, 17-45, 17-64
- OAM 10g SSO solution, 17-1
- OAM 11g SSO solution, 16-1
- OAM solution, 8-19
- oamauthenticationprovider.war, 16-8, 17-5
- oamAuthnProvider.jar, 15-13, 16-8, 17-5
- OAMCfgTool, 17-2, 17-6, 17-35, 17-39
 - about using, 17-15
 - Create mode parameters, 17-18
 - host identifiers created, 17-33
 - Known Issues, 17-34
 - process overview, 17-17
 - Validate mode parameters, 17-27
- oamcfgtool.jar, 15-13, 17-5
- OID Authenticator, 16-23, 16-36, 17-45, 17-64, 18-4, 18-12
- OID patches, 8-2
- one-way SSL, 8-12
- OPSS APIs
 - User and Role, D-1
- OPSS security store, 2-3
- OPSS SSO Framework, 8-19
- OPSS System Properties, F-1
- opss_purge_changelog, 8-8
- Oracle Access Manager
 - Integration with OSSO, 15-3
- Oracle ADF security, 5-1
- Oracle Business Intelligence Publisher, 14-1
 - audit report example, 14-8
- Oracle Entitlements Server, 5-2, 5-5, 9-1, 9-31
- Oracle Fusion Middleware Audit Framework, 12-1, 12-2
 - architecture, 12-4
 - concepts, 12-4, 12-7
- Oracle Information Lifecycle Management Assistant, 13-28
- Oracle Internet Directory, 4-1
- Oracle Internet Directory 10.1.4.3 patch, 4-2
- Oracle Internet Directory tuning, 4-2
- Oracle JDeveloper 11g, 5-1
- Oracle Platform Security Services, 15-1
- OracleAS Single Sign-On solution, See Also OSSO, 18-1
- oracle.deployed.app.dir, B-70
- oracle.deployed.app.ext, B-70
- oracle.security.jps.config, 1-7, A-1

- oracle.security.jps.jaas.mode, 21-6
- oracle.security.jps.log.for.approle.substring, L-9
- oracle.security.jps.log.for.enterprise.principalname, L-9
- oracle.security.jps.log.for.permclassname, L-9
- oracle.security.jps.log.for.permeffect, L-9
- oracle.security.jps.log.for.permtarget.substring, L-9
- Oracle-specific applications, 5-1
- OSSO
 - existing implementation, 15-2
 - Identity Asserter, 18-1, 18-12
 - new users, 18-4
 - processing, 18-2
 - Tips and Troubleshooting, 18-18
 - solution, 15-1, 18-1
- OSSO Identity Asserter, 18-2

P

- packaging an J2EE application, 21-10
- Packaging Credentials, 21-10
- Packaging Policies, 21-10
- password credential, 10-1
- password validation, 2-9
- passwords, 2-9
- permission, 20-13
- permission class, 20-14
- permission classes, 3-5, 8-1, 21-22
- permission inheritance, 2-4
- permissions to anonymous role, 2-7
- permissions to authenticated role, 2-7
- PermissionSetManager class, 20-5
- policy domain
 - URL prefixes, 17-51, 17-55, 17-63
- policy migration settings, 6-4
- Policy Store, 3-4
- policy store, 2-3
- policy store cache, 9-28
- policy store removal, 3-5
- PolicyStoreAccessPermission, 21-22
- PolicyStoreIncompatibleVersionException, L-41, L-42
- polystore.refresh.interval, 9-28
- Post-installation tasks, 5-4
- principal, 2-2
- principal name comparison, 2-9, 2-10
- principal.cache.key, 23-2
- PrincipalEqualsCaseInsensitive, 2-10
- PrincipalEqualsCompareDnAndGuid, 2-10
- Procedure
 - WebGate
 - To manually configure a Web server, 16-14
- Process overview
 - OAMCfgTool, 17-17
 - Oracle Access Manager Authenticator for Web and non-Web Resources, 15-10
 - OSSO Identity Asserter, 18-3
- production environment, 5-3
- props.auth.level, 8-22
- props.auth.uri, 8-22

- props.auth.url, 8-22

R

- RCU, 8-6
- reassociateSecurityStore, 9-26, I-2
- Reassociation of credentials, 3-6
- Reassociation of policies, 3-5
- recovery of server files, 5-3
- reference integrity, 3-2
- referential integrity, 8-2
- remove.anonymous.role, 21-4
- Resource Catalog, 20-4
- resource permissions, 20-13
 - managing, 20-14
- resource type, 20-13
- resource-based policies, 2-2
- ResourceManager class, 20-5
- ResourcePermission class, 20-12
- resourcetypeenforcementmode, F-6, F-14
- ResourceTypeManager class, 20-5
- revokeAppRole, 9-12
- revokePermission, 9-14
- role category, 2-11
- role hierarchy, 2-4
- RoleCategoryManager class, 2-11

S

- SAML 1.1 identity assertion, 3-3
- SAML 2.0 identity assertion, 3-3
- scenarios, 4-4
- Security Provider Configuration, 8-10, 8-18
- Security Provider for WebLogic SSPI, 15-11
- security store, 2-3
- security-related commands, 5-5
- server restart, 4-3, F-1
- service instance update script, E-1
- Service Providers, 25-5
 - introduction, 25-5
 - understanding, 25-5
- Set Security Provider, 8-11
- setAuditPolicy, C-46
- setAuditRepository, C-47
- setDomainEnv shell script, F-1, L-7
- setPolicy, 20-7, 20-12
- Setting a Node in LDAP server, 8-3
- setting up providers
 - OAM Asserter with Oracle Web Services Manager, 16-36
 - OAM Authenticator, 16-30
 - OAM Identity Assertion, 16-23, 17-45
 - OSSO Identity Asserter, 18-12
- Single Sign-On, 8-19
- single sign-on solutions for Fusion Middleware, See Also SSO, 15-1
- split profiles, 7-7
- SPNEGO, 3-4
- SPNEGO tokens, 3-4
- SSL

- and User/Role APIs, 25-28
- anonymous, 8-12
- one-way, 8-12
- SSL to a DB, 8-9
- SSO
 - enterprise level, 15-2
 - existing 10g SSO, 15-2
 - Oracle Access Manager, 15-5
 - Synchronization Filter, 16-40, 17-66, 18-16
- SSO Logout URL, 16-39
- SSO service, 8-20
- SSO service configuration, 8-20
- sso.provider.class, 8-22
- storing policies and credentials, 4-2
- subject, 2-3, 2-8, 2-9
- supported
 - identity store types, 3-2
- synchronizing
 - user and SSO Sessions, 16-40, 17-66, 18-16
- system component, 2-4
- system-jazn-data.xml, 21-1

T

- Task overview
 - Configuring the OAM Authenticator, 16-30, 17-49
 - Deploying and configuring OAM Identity Assertion for single sign-on includes, 16-16, 17-35
 - Deploying OSSO Identity Asserter, 18-4
 - Deploying the Identity Asserter with Oracle Web Services Manager, 16-35, 17-60
 - Installing required components for OAM Authentication Provider, 16-8, 17-4
- test environments, 6-5
- token.provider.class, 8-23
- troubleshooting
 - search fails against Microsoft Active Directory, L-40
- typical security practices, 5-4

U

- Unsupported Methods in PS2, 23-4
- updateServiceInstanceProperty, E-2
- updating instance with script, E-1
- upgradeSecurityStore, G-1
- URL
 - SSO Logout URL, 16-39
- User and Role API, D-1
 - Javadoc, 25-29
 - programming tips, 25-14
- User and Role APIs
 - and WebLogic authenticators, 25-2
 - environment setup, 25-5
 - introduction, 25-1
 - programming tips, 25-13
 - summary, 25-2
- User and Role SPI
 - Javadoc, 25-39

- UseRetrievedUserNameAsPrincipal, 3-4
- user.login.attr, L-28
- username.attr, L-28

V

- virtualize, 7-3, 7-4, 7-5, F-20
- virtualized identity, 7-1

W

- WAR file, 21-1
- WebLogic
 - Authentication provider, 15-1, 16-19, 17-42
 - Authentication providers
 - Identity Assertion, 16-19, 17-42
 - J2EE applications, 15-11
 - WebLogic Administration Console, 4-2
 - WebLogic Scripting Tool (WLST), 16-20, 17-43
 - weblogic-application.xml, 21-1
 - web.xml, 3-4, 21-1, 21-9
 - WLSGroupImpl, 2-5, 9-11, 9-12, 21-9, 22-12
 - WLST
 - createAppRole, 9-9, 9-10
 - createCred, 10-8
 - createResourceType, 9-16
 - deleteAppPolicies, 9-15
 - deleteAppRole, 9-10
 - deleteCred, 10-8
 - deleteResourceType, 9-17
 - getResourceType, 9-17, 9-18, 9-19, 9-20, 9-21, 9-22, 9-23, 9-24, 9-25
 - grantAppRole, 9-11
 - grantPermission, 9-13
 - listAppRoleMembers, 9-12
 - listAppRoles, 9-12
 - listCred, 10-7
 - listPermissions, 9-15
 - reassociateSecurityStore, 9-26
 - revokeAppRole, 9-12
 - revokePermission, 9-14
 - updateCred, 10-7
 - WLSUserImpl, 2-5, 21-9, 22-12

X

- X509 identity assertion, 3-3