Oracle® Fusion Middleware

Security Guide

11*g* Release 1 (11.1.1)

E10043-04

October 2009

ORACLE®

Primary Author: Carlos Subi

Contributing Author: Vinaye Misra

Contributor: Soumya Aithal, Moushmi Banerjee, Josh Bregman, Rachel Chan, Andre Correa, Marc Chanliau, Pratik Datta, Jordan Douglas, Guru Dutt, Todd Elwood, Gail Flanegin, Vineet Garg, Vikas Ghorpade, Dheeraj Goswami, Sandeep Guggilam, Shiang-Jia Huang, Dan Hynes, Supriya Kalyanasundaram, Lakshmi Kethana, Ganesh Kirti, Ashish Kolli, Alex Kosowski, Rohit Koul, Stephen Lee, Belinda Leung, Nithya Muralidharan, Raymond Ng, Frank Nimphius, Craig Perez, Sudip Regmi, Vinay Shukla, Bhupindra Singh, Mamta Suri, Kavita Tippana, Srikant Tirumalai, Ramana Turlapati, Sirish Vepa, Jane Xu, Sam Zhou.

Contents

3 Understanding Users and Roles

4 Understanding Identities, Policies, and Credentials

5 About Oracle Platform Security Services Scenarios

Part II   Basic OPSS Administration

6 Security Administration

11   Introduction to Oracle Fusion Middleware Audit Framework

12   Configuring and Managing Auditing

13 Using Audit Analysis and Reporting

Part IV   Developing with Oracle Platform Security Services APIs

14   Overview of Developing Secure Applications with Oracle Platform Security Services

15   Manually Configuring JavaEE Applications to Use OPSS

16   Developing Authentication

17   Developing with the Credential Store Framework

18 Developing Authorization

20 Developing with Oracle HTTPClient Security

Part V Appendices

A OPSS Configuration File Reference

B   File-Based Identity and Policy Store Reference

C   Oracle Fusion Middleware Audit Framework Reference

D User and Role API Reference

E Administration with WLST Scripting and MBean Programming

F OPSS System and Configuration Properties

G Upgrading Security Data

H   References

I   Troubleshooting Security in Oracle Fusion Middleware

Index

List of Examples

List of Figures

List of Tables

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 Fusion Middleware security.

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 security auditing; parts V through VII are relevant to developers; and part VIII 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 more information, visit the Oracle Accessibility Program Web site at http://www.oracle.com/accessibility/.

Accessibility of Code Examples in Documentation

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

Accessibility of Links to External Web Sites in Documentation

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

Deaf/Hard of Hearing Access to Oracle Support Services

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

Related 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 Virtual Directory*

- *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 Identity Federation*

- 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.
italic	Italic type indicates book titles, emphasis, terms defined in text, or placeholder variables for which you supply particular values.
monospace	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 changes introduced in the 11gR1 and PS1 releases. 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.

New Features in Release 11gR1 PS1

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

- The class Resource Permission. For details, see Section 18.4, "The Class oracle.security.jps.ResourcePermission."

- Principal name comparison has been enhanced. For details, see Section 3.7, "Principal Name Comparison Logic."

- Manual settings for policy migration have been simplified. In particular, versioning the application is no longer required. For details, see Section 15.4.1, "Parameters Controlling Policy Migration," and Section 15.4.2, "Policy Parameter Configuration According to Behavior."

- The WSLT command `migrateSecurityStore` supports the embedded LDAP store as a target. For details, see Section 15.4.8, "Migrating Identities with the Command migrateSecurityStore."

- 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 WSLT command `reassociateSecurityStore` supports an existing LDAP node as a target. For details, see Section 8.4.2.11, "reassociateSecurityStore."

- Oracle Fusion Middleware pages have been improved. In particular, using these pages, one can specify the SSO service to use in a domain. For details, see the following sections:

  - Section 8.4.1.1, "Managing Application Policies"

  - Section 8.4.1.2, "Managing Application Roles"

  - Section 8.2.1, "Reassociating Domain Stores with Fusion Middleware Control"

  - Section 8.5, "Configuring Other Artifacts with Oracle Fusion Middleware Control"

New Sections in this guide

New sections added to this guide in this release include the following:

- Section 8.1.3, "Cataloging Oracle Internet Directory Attributes"
- Section 8.4.2.12, "Granting Policies to Anonymous and Authenticated Roles with WLST Commands"
- Section 8.4.2.13, "Application Stripe for Versioned Applications in WLST Commands"
- Section 8.6, "Configuring LDAP-Based Policy Stores"
- Section I.2.1, "Missing Policies in Reassociated Policy Store"
- Section I.16, "Characters in Policies"
- Section I.17, "Granting Permissions in J2SE Applications"

New Features in Release 11gR1

The features introduced in the 11gR1 release 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

This release introduces the following changes:

- 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:

1

Overview of Java Security Models

This chapter provides an overview of the security models in the Java Platform, Standard Edition (JavaSE), the Java Authentication and Authorization Service (JAAS), the Java Platform, Enterprise Edition (JavaEE), and the Java Authorization Contract for Containers (JACC).

The coverage of topics in this chapter is not comprehensive; for complete details, consult the appropriate internet site.

The presentation is divided in the following sections:

- Basic Security Concepts
- Java Security Model
- Java Authentication and Authorization Service
- Java EE Security Model
- Java Authorization Contract for Containers
- Comparing the Java Security Models

1.1 Basic Security Concepts

Authentication deals with the question "Who is trying to access services?" In any system or application it is important to ensure that the identity of the entity or caller trying to access a resource is appropriately identified. In a multitier application, the entity or caller can be a human user, a business application, a host, or one entity acting on behalf of (or impersonating) another entity.

Authentication information, such as user names and passwords, is stored in a user repository, such as an XML file, database, or directory service. When a subject attempts to access an application, such as by logging in, the security provider looks up the subject in the user repository and verifies the subject's identity. A security provider is a module that provides an implementation of a specific security service such as authentication or authorization. The Oracle Internet Directory is an example of a repository.

Authorization deals with the question "Who can perform tasks on resources?" Resources are typically expressed in terms of URL patterns for Web applications, and method permissions for EJBs. Authorization is on a per-role basis, with appropriate permissions being assigned to each defined role in an application.

1.2 Java Security Model

The Java security model is based on controlling the operations that a class can perform when it is loaded into a running environment. For this reason, this model is called code-centric or code-based.

The following sections explain the basic concepts necessary to understand how this model works:

- Permissions
- Protection Domains and Security Policies
- Security Managers and Access Controllers

For more details about the Java security model, see
`http://java.sun.com/developer/technicalArticles/Security/whitepaper/JS_White_Paper.pdf`.

1.2.1 Permissions

A *permission* is a set of permissible operations on some set of resources. Every Java class loaded into a running environment is assigned a set of permissions according to some criteria, each permission granting a specific access to a particular resource. For example, a permission can constrain the access to a database or disallow the editing of a file.

In code-based security, permissions are granted based on code characteristics, such as where the code is coming from and whether it is digitally signed (and by whom). A *codebase* is a URL indicating code location, such as the following:

- `file:` (any file on the local file system)
- `http://*.oracle.com` (any file on any host at oracle.com)
- `file:${j2ee.home}/lib/oc4j-internal.jar`

A *codesource* expands this concept to optionally include an array of certificates (stored in a Java keystore) to verify signed code originating from the location. A codesource is represented by a `java.security.CodeSource` instance, which is constructed by specifying a `java.net.URL` instance and an array of `java.security.cert.Certificate` instances.

The abstract Java class `java.security.Permission` represents a permission; concrete types, all derived from this class, include the following:

- `java.security.AllPermission`
- `java.lang.RuntimePermission` (includes only a resource target)
- `java.io.FilePermission` (includes a resource and actions)

> **Important:** The class `AllPermission` should be used with caution and only when necessary.

For further details about permissions, see
`http://java.sun.com/j2se/1.5.0/docs/guide/security/spec/security-spec.doc3.html`.

1.2.2 Protection Domains and Security Policies

A *protection domain* associates permissions with codesources. The policy currently in effect is what determines protection domains. A protection domain contains one or more codesources. It may also contain a Principal array describing who is executing the code, a classloader reference, and a permission set (`java.security.PermissionCollection` instance) representing a collection of Permission objects.

A *security policy* defines the protection domains of an environment, that is, it identifies the permissions assigned to classes from specified sources. The permissions assigned to a class by a protection domain are bound statically, when the class is loaded, or dynamically, when the executing code attempts a security-sensitive operation. Protections domains are specified in one or several policy files.

The following sample entry in a policy file illustrates how code located in the directory /home/sysadmin *and* signed by jJoe is granted read access to the file /tmp/abc:

```
grant signedby jJoe codeBase "file:/home/sysadmin/" {
    permission java.io.FilePermission "/tmp/abc", "read";
};
```

The `signedBy` clause in the preceding example is optional. If omitted, the permission is granted to all code in the specified location. Figure 1–1 illustrates the relationship between classes, protection domains, and permission sets.

Figure 1–1   Associating classes with permissions through protection domains
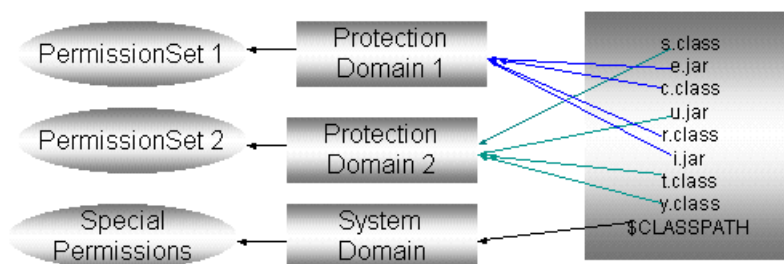
For details about policy file format and policy customization, see
http://java.sun.com/j2se/1.5.0/docs/guide/security/spec/security -spec.doc3.html#20128.

1.2.3 Security Managers and Access Controllers

A *security manager* is the component of the Java security model that enforces the permissions granted to applications by security policies. For any security-sensitive operation that an application attempts, the security manager checks the application permissions and determines whether the operation should be allowed. The Java class `java.lang.SecurityManager` represents a security manager and includes several check methods to determine whether an operation should be allowed or a given permission is in effect.

> **Important:** An application can run with or without the control of a security manager; most thin-applications (such as Web browsers) enable a security manager by default.
>
> The security manager is not mandatory for Java policies to be in effect: whether an application chooses to enforce Java policies depends on how permissions are checked by the application. For example, an application can use the method `AccessController.checkPermission` to enforce Java policies without the security manager being turned on.

An *access controller* is the object used by the security manager (or directly by an application, if the security manager is not enabled) to control operations and decisions. More specifically, an access controller:

- Decides whether access to a system resource should be allowed or denied, based on the current security policy in effect.

- Marks code as being privileged, thus affecting subsequent access determinations.

- Allows saving the current calling context so access-control decisions that consider the saved context can be made from other, different contexts.

The Java class `java.security.AccessController` represents an access controller and includes the method `checkPermission(aPerm)` that determines whether the access requested in the passed permission should be granted. The following code snippet illustrates the use of this method to allow reading the file /temp/testFile:

```
FilePermission perm = new FilePermission("/temp/testFile", "read");
AccessController.checkPermission(perm);
```

`checkPermission` evaluates according to the particular implementation of the access controller. The default implementation examines the entire call stack, the classes in it, and the permissions granted to those classes to determine whether to grant a request. The method returns silently if the request is granted or throws an exception if the request is denied (or the passed permission is invalid).

> **Note:** `AccessController.checkPermission` performs security checks within the context of the current executing thread. Sometimes, however, an application must perform a security check using a context different from the current one.
>
> The method `AccessController.getContext` takes a snapshot of the current context and returns it in an access control context. The context can then invoke its own method `checkPermission`, thus basing its evaluation on the context it encapsulates rather than the current context.

A security manager represents a central point of access control, while an access controller implements a particular access algorithm.

For further details about security manager and access controllers see http://java.sun.com/j2se/1.5.0/docs/guide/security/spec/security-spec.doc6.html.

1.3 Java Authentication and Authorization Service

The main features introduced by JAAS are the following:

- It extends the Java security model by allowing checks on the identity of the caller (a feature notably missing in the Java model), checks on the permissions granted to the code being run and subject-based authorization.

- It implements the standard Pluggable Authentication Module (PAM) framework, which decouples an application from its authentication service and supports the implementation of custom authentication modules.

The following sections explain the basic concepts necessary to understand how this model works:

- Principals and Subjects
- Login Modules
- Subjects and the Access Control Context

Oracle Platform Security Services (OPSS) includes a scalable JAAS provider, uses JAAS as a standard mechanism for fine-grained authorization, and supports JAAS authorization for both Web-based and EJB-based applications. The main OPSS extensions to JAAS are supported by the file `jazn-data.xml` which provides: (a) a light weight XML provider; (b) and LDAP provider; (c) application and anonymous users and roles for fine-grained authorization.

For details about key JAAS features, see
`http://java.sun.com/javase/6/docs/technotes/guides/security/jaas/JAASRefGuide.html`

1.3.1 Principals and Subjects

A *principal* is the identity assigned to an entity (such as a user, a group, or a system process) by an authentication process. The abstract concept of a principal is represented by the Java interface `javax.security.auth.Principal`. A concrete principal is represented by an instance of a class implementing this interface. Each principal instantiated by an application must have a distinct name, which identifies the principal.

A *subject* is a grouping of related security information that includes a collection of principals and, possibly, credentials such as passwords or cryptographic keys. The Java class `javax.security.auth.Subject` represents a subject and an instance of this class is created and populated with principals when authentication succeeds. Thus, a subject represents an authenticated entity, and the principals and credentials in a subject are used to determine what actions the authenticated entity can perform.

For further details about principals and subjects, see:

- `http://java.sun.com/j2se/1.5.0/docs/api/java/security/Principal.html`

- `http://java.sun.com/j2se/1.5.0/docs/api/javax/security/auth/Subject.html`

1.3.2 Login Modules

A *login module* is a component that authenticates users and populates a subject with principals. Login modules can be plugged in and used by applications without changes to application code. An application can use multiple login modules.

The abstract concept of a login module is represented by the Java interface `javax.security.auth.spi.LoginModule`. A concrete login module is represented by an instance of a class implementing this interface.

To authenticate users with login modules, a JavaSE application first instantiates a login context, then looks up a configuration file and loads all the login modules configured for the application, and, in turn, invokes the method `login` in each of them. Each login module adds, as appropriate, one or more principals to a subject that becomes available if, and only if, the authentication succeeds.

In JavaEE applications, however, login modules may be invoked implicitly by the container without the need for the application to instantiate a login context.

A *callback handler* is the way a login module communicates with users to obtain authentication data. It can be specified when the application instantiates a login context so that all login modules invoked by that context use the passed callback handler. This mechanism frees login module logic from user interaction.

An application can use a stack of login modules to authenticate its users, and each module, independently from the others in the stack, performs its own computations. The stack used by an application is defined in its deployment descriptor; the sequence in which login modules are enumerated in that file 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).

> **Note:** On Oracle WebLogic Server, a standard JAAS login module cannot be used as-is. For details about WebLogic Authenticator providers, see Section 4.1.1, "Oracle WebLogic Authenticators."

For further details about login module configuration, see http://java.sun.com/j2se/1.5.0/docs/api/javax/security/auth/login/Configuration.html.

1.3.3 Subjects and the Access Control Context

Once the authentication process succeeds and a subject becomes available, an application calls the method `doAs` or `doAsPriviledged` to authorize the subject to perform a sensitive action within a particular context. The difference between them is that `doAs` uses the current executing context, while `doAsPriviledged` uses a provided context that can be different from the executing one.

For further details about actions and control contexts, see:

- http://java.sun.com/j2se/1.5.0/docs/api/java/security/class-use/PrivilegedAction.html

- http://java.sun.com/j2se/1.5.0/docs/api/java/security/AccessController.html

For further details about the methods `doAs` and `doAsPrivileged`, see the following document:

- http://java.sun.com/j2se/1.4.2/docs/guide/security/jaas/JAASRefGuide.html.

1.4 Java EE Security Model

The JavaEE security model is a role-based, declarative model based on container-managed security, where resources are protected by roles that are assigned to users. This model allows decoupling an application from its underlying security infrastructure since security can be specified separate from the application logic in an application deployment descriptor. The container, where an application runs, provides security for the application according to a specifications in the deployment descriptor. This model also allows embedding security data (annotations) in the application code that can be referenced in deployment descriptors.

The following sections explain the basic concepts necessary to understand how this model works:

- Container-Based Security
- The Authentication Model
- The Authorization Model

1.4.1 Container-Based Security

A container provides two kinds of security to applications that run in it: declarative security and programmatic security.

Declarative security refers to security expressed in an application deployment descriptor file. At run time, the container uses this file to enforce access to the application resources, such as a Web module URL or an EJB bean method.

For details about EJB technology and the Web Server Security Model, see:

- http://java.sun.com/developer/technicalArticles/J2EE/jpa/
- http://docs.sun.com/app/docs/doc/820-1066/abxep?a=view

Programmatic security refers to security expressed in the code by security-aware applications. This kind of security is useful when declarative security is not sufficient to express fully an application's security requirements.

To use programmatic security, typically, an application implements the interface `javax.ejb.EJBContext` or the interface `javax.servlet.http.HttpServRequest,` and uses the `weblogic.security` APIs and the following methods to customize security decisions:

- `EJBContext.isCallerInRole(aRoleName)`
- `EJBContext.getCallerPrincipal()`
- `HttpServRequest.isUserInRole(aRoleName)`
- `HttpServRequest.getUserPrincipal()`

For further details about these interfaces, see:

- http://java.sun.com/j2ee/1.4/docs/api/javax/servlet/http/HttpSe rvletRequest.html
- http://java.sun.com/j2ee/1.4/docs/api/javax/ejb/EJBContext.html

1.4.2 The Authentication Model

A server authenticates the end-user of a Web application in the following ways: basic, form, or client-cert. They differ in the extent of security they provide, but, basically, they all obtain data from the user, such as account name and password, that is

processed by the server to create a credential or to deny access. The way a server authenticates a user is specified in the application deployment descriptor.

The server validates the supplied credential against the configured identity store and uses it to authorize further access to secured resources (in the server) or calls to enterprise bean methods (possibly running in a different server).

1.4.3 The Authorization Model

The JavaEE authorization model is based on security roles. A *security role* is a set of users that can be specified in the application deployment descriptor or in the application code (with annotations). An application controls access to its resources (such as a Web module URL or a bean method) by specifying the roles that are allowed to perform a given operation on a resource.

During deployment, application-scoped roles are mapped to security entities in the running environment, such as users, a group of users, or principals. This mapping is specified in a configuration file different from the deployment descriptor.

At run time, when a principal requests access to a resource, the container determines if the principal has associated a role allowed to access the resource, and if it does, the container transfers control to the application and the request is allowed.

This model also allows an application to specify the identity under which an enterprise Java bean or Web component must run (run-as method-based authorization) and supports transport-layer security (SSL).

For details about servlet security see
http://java.sun.com/products/servlet/whitepaper.html.

1.5 Java Authorization Contract for Containers

The Java Authorization Contract for Containers (JACC) specifies a contract between JavaEE containers and authorization modules, so that the container can provide the appropriate authorization. Specifically, JAAC performs the following tasks:

- Defines a role as a collection of permissions
- Grants principals permissions in roles
- Determines whether a principal has been granted the permissions in a role
- Maps identifiers (embedded in application code) to application-scoped roles

JACC allows the JavaEE security model fully to leverage the Java security model by translating JavaEE security constraints into Java permissions. JACC is not enabled by default.

1.6 Comparing the Java Security Models

Here are summarized the features that characterize each of the three models and differentiate them from one another.

All three models are fully supported by OPSS, a platform that supports policy stores where a permission can consist of both Java and JAAS permissions, thus enabling the combination of the code-centric and the user-centric models.

JavaSE

This model is called code-centric because it allows the specification of code security in a predefined policy file. When a class is loaded in the Java run time environment, the class loader automatically associates the following information with the class:

- The location where it was loaded from

- The entity that signed it (if any)

- A set of default permissions

Notably, the identity of the user executing the code is not available at loading time. Authentication is carried out by the application itself as appropriate. Authorization is managed by the application itself or by a security manager.

JAAS

This model extends the JavaSE code-centric model by introducing a user-centric model where both the nature of the code and the executor are taken into account in security decisions.

Authentication can be customized with login modules, and authorization is managed, as in the JavaSE model, according to a predefined policy. Since it also supports custom permissions, this model is suitable for applications where a more fine-grained, customized control of authentication is required.

JavaEE

In this model, secured resources are identified by URL patterns (in Web modules) or method names (in EJB modules); the container where the application runs enforces authentication and authorization according to specifications in the application deployment descriptor or annotations in the application code. The transport layer can use the secure protocol HTTPS.

This model is the least complicated to deploy and administer, and it is often used with the JAAS model.

1.6.1 Summary of Model Comparison

One way to distinguish these three models is to answer the question who does each model protect from accessing what?

The answers, for each model and for OPSS, are summarized in the following table:

Table 1–1   Summary of Model Comparison

Model	Who is Permitted	What is Protected
JavaSE	Code	Code, such as file systems, network resources, classloaders.
JAAS	Subject	Code, such as file systems, network resources, classloaders.
JavaEE	Roles	Resources, such as URLs and EJB methods.
JACC	Subject	Permissions on JEE resources, such as URLs and EJB methods.
OPSS	Code and Subject	All of the above.

2

Introduction to Oracle Platform Security Services

Oracle Platform Security Services (OPSS) is a security platform that can be used either with Oracle WebLogic Server or standalone. 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*.

2.1 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 standards-based 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 cryptographic key management or interfaces with user repositories 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.

2.1.1 OPSS Main Features

OPSS complies with the following standards: role-based-access-control (RBAC); Java Enterprise Edition (JavaEE); 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-specific policies
- Secure storage and access of system credentials through the Credential Store Framework
- Auditing
- Role administration and role mappings
- The User and Role API
- Security configuration and management
- SAML and XACML
- Oracle Security Developer Tools, including cryptography tools.

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.*

2.2 OPSS Architecture Overview

OPSS comprises WebLogic's security and Oracle's Fusion Middleware security. The following graphic illustrates the layered architecture that combines these two security frameworks:
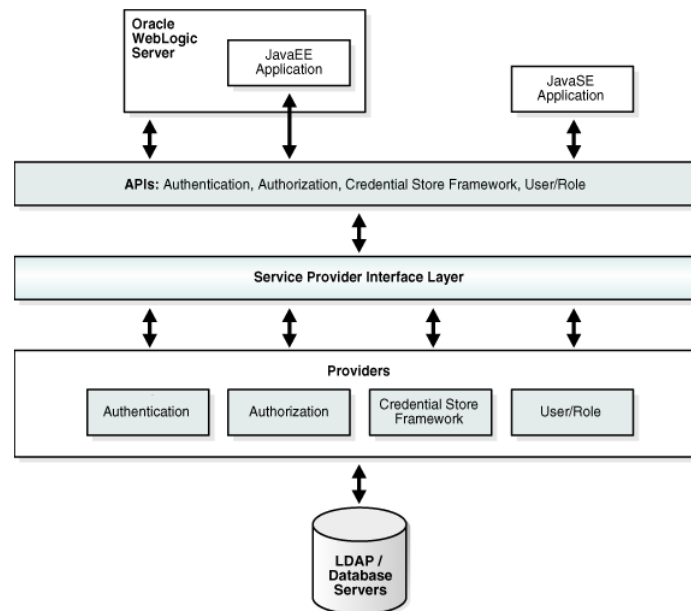
This figure depicts the various security components as layers. The uppermost layer consists of Oracle WebLogic Server and the components and Java applications running on the server; below is the API layer consisting of Authentication, Authorization, CSF, and User and Role APIs, followed by the Service Provider Interface (SPI) layer and the list of service providers. The bottom layer illustrates repositories including LDAP and database servers.

The list of providers in the above figure is not comprehensive: other providers include the role mapping provider and the audit provider.

Security Services Provider Interface

Security Services Provider Interface (SSPI) provides Java EE container security in permission-based (JACC) mode and in resource-based (non-JACC) mode. It also provides resource-based authorization for the environment, thus allowing customers to choose their own security model.

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

JAZN functionality was redesigned and expanded to include the Credential Store Framework (CSF), the Common Audit Framework (CAF), and other components, and combined with SSPI as Oracle Platform Security Services (OPSS).

OPSS includes the following services: Credential Store Framework, User and Role API, Common Audit Framework, Identity Services, and improved design-time support.

2.2.1 Benefits of OPSS

OPSS offers multiple benefits including:

- Allowing developers to focus on application and domain problems

- Support for enterprise deployments

- Verified interop testing across different LDAP servers and SSO systems

- Certified on WebLogic Server

- Pre-integration with Oracle products and technologies

- A consistent security experience for developers and administrators

- A uniform set of APIs for all types of applications

- Optimization of development time with abstraction layers (declarative APIs)

- A simplified application maintenance

- Changing security rules without affecting application code

- Ease of administration tasks

- Integration with identity management systems

- Integration with legacy and third-party security providers

OPSS combines SSPI and JPS to provide a framework where both WebLogic and Oracle applications can seamlessly run in a single environment, the Oracle WebLogic Server.

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 build security features for 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 an embedded LDAP repository. Domains can be configured, however, to use identity data in other kinds of LDAP repositories, such as Oracle Internet Directory, ActiveDirectory, Sun Java System Directory Server, Oracle Virtual Directory, Novell eDirectory, and OpenLDAP. In addition, Oracle WebLogic Server provides a generic LDAP authenticator that can be used with other LDAP servers not in the preceding list.

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

> **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.

2.3 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`.

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

2.4 OPSS for Administrators

Depending on the application type, a security feature is managed with the Oracle WebLogic Administration Console, WLST commands, or Fusion Middleware Control, as follows:

- For JavaEE applications, all security is managed with the Oracle WebLogic Administration Console or with WLST commands.

- For Oracle SOA, Oracle WebCenter, MDS, and Oracle ADF applications, authentication is managed with the Oracle WebLogic Administration Console and authorization is managed with Fusion Middleware Control.

For details about security administration, see Chapter 6, "Security Administration."

2.5 OPSS for Developers

This section summarizes the main OPSS features that developers typically implement in different kind of applications, in the following scenarios:

- Scenario 1: Enhancing Security in a JavaEE Application

- Scenario 2: Securing an Oracle ADF Application

- Scenario 3: Securing a JavaSE Application

2.5.1 Scenario 1: Enhancing Security in a JavaEE Application

A JavaEE application can be enhanced to use OPSS APIs such as the CSF, User and Role, or Policy Management: user attributes, such as such as a user's email, phone, or address, can be retrieved using 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; and authorization policy data can be managed with the Policy Management API.

JavaEE 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.

2.5.2 Scenario 2: Securing an Oracle ADF Application

Oracle Application Development Framework (ADF) is a JavaEE development framework available in Oracle JDeveloper that simplifies the development of JavaEE 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 JavaEE 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 3.3, "The Authenticated Role"

- Section 3.4, "The Anonymous User and Role"

- Section 4.2, "Policy Store Basics"

- Section 4.3, "Credential Store Basics"

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

2.5.3 Scenario 3: Securing a JavaSE Application

Most of the OPSS features that work in JavaEE applications work in JavaSE 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 JavaSE 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 `ORACLE_HOME` and `DOMAIN_HOME` and related concepts, see Glossary in *Oracle Fusion Middleware Concepts*.

Required JAR in Classpath

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

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

Login Modules

JavaSE 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 JavaSE applications is the IdentityStore login module.

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

3

Understanding Users and Roles

This chapter describes users and roles, the anonymous role, the authenticated role, and role mapping for Oracle Platform Security Services (OPSS). 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

For further details about managing users and roles programmatically, see Chapter 19, "Developing with the User and Role API."

3.1 Terminology

This section contains the definition of terms, some generic and some specific to OPSS, consistently used throughout this guide.

Users

A *user*, or *enterprise user*, is an end-user accessing a service. User information is stored in the domain identity store, typically instantiated by the WebLogic Server DefaultAuthenticator. 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 3.4, "The Anonymous User and Role."

Roles

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

A JavaEE *logical role* is a role specified declaratively or programmatically by a JavaEE 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 OPSS *application role* is a collection of users, groups, and application roles; it can be hierarchical. This role is specific to the application, defined by the application policy, and not necessarily known to the JavaEE container. Application roles are scoped in the sense that they are visible only when the application runs. They can be mapped to other application roles defined in the same application scope (and also to enterprise users or groups), and they are used in authorization decisions.

For details about the *anonymous role*, see Section 3.4, "The Anonymous User and Role." For details about the *authenticated role*, see Section 3.3, "The Authenticated Role."

Permission

A *permission* consists of a class, a resource, and a subset of the actions allowed by the type of the resource. For a permission class that can be used out-of-the-box, see Section 18.4, "The Class oracle.security.jps.ResourcePermission." Permission names are case sensitive.

Suppose, for example, that the resource type Job allows the following actions: schedule, cancel, and view. If myJob1, myJob2 are resources type Job, then one can have the following permissions:

- myJob1; schedule, view

- myJob2; cancel, view

Principal

A *principal* is the identity assigned to a requesting entity (such as a user) by an authentication process. A principal can be a user, an enterprise group, or an application role. For details about the way principal names are compared, see Principal Name Comparison Logic.

Grant

A *grant or functional policy* consists of an entitlement or a list of permissions; a code source; and a principal. A grant describes who (the principals or code) can do what (the actions in permissions) on a given resource.

OPSS Subject

An OPSS *subject* is a collection of principals and, possibly, user credentials such as passwords or cryptographic keys. WebLogic authentication populates the subject with users and groups, and then augments the subject with application roles. For details about how anonymous data is handled, see Section 3.4.1, "Anonymous Support and Subject."

Security Stores

The *Identity Store* is the repository of enterprise users and groups. Out-of-the-box the identity store is the WebLogic DefaultAuthenticator. Other types of identity stores include LDAP, RDBMS, or custom. This store is administered with the WebLogic Administration Console.

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 domain credentials. OPSS uses one logical store to keep both policies and credentials. This store is administered with Oracle Enterprise Manager Fusion Middleware Control.

For details about stores, see Chapter 4, "Understanding Identities, Policies, and Credentials."

Other Terms

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

3.2 Role Mapping

OPSS supports the mapping of application roles in the domain policy store to enterprise groups in the domain identity store. This mechanism allows users in enterprise groups to access application resources as specified by application roles. The mapping is allowed to be many-to-many.

> **Notes:** Oracle JDeveloper allows specifying this mapping when the application is being developed. Alternatively, the mapping can be also specified, after the application has been deployed, using WLST or Fusion Middleware Control as explained in Section 8.4.1.2, "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.

3.2.1 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, members of a role 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.

The following example illustrates application role management with 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>
            </membe>
            <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>

      <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>
            </jazn-policy>
        </policy-store>
```

Table 3–1 summarizes the permissions that each of the five users and roles in the above hierarchy get according the inheritance rule:

Table 3–1   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)

3.3 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 15.1, "Configuring the Servlet Filter and the EJB Interceptor."

3.4 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 15.1, "Configuring the Servlet Filter and the EJB Interceptor."

3.4.1 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.

3.5 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*.

3.6 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 I.12, "User Gets Unexpected Permissions."

3.7 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

3.7.1 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 domain 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 authorized 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.

3.7.2 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 WLST commands. 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.

4

Understanding Identities, Policies, and Credentials

Applications use the identity, policy, and credential stores configured in the domain in which they run. This chapter introduces the repository that store security data for identity, policy, and credential data. OPSS supports file- and LDAP-based policy and credential stores.

This chapter is divided into the following sections:

- Authentication Basics
- Policy Store Basics
- Credential Store Basics

For definitions of the terms used in this chapter, see Section 3.1, "Terminology."

For scenarios illustrating the use of stores, see Chapter 5, "About Oracle Platform Security Services Scenarios."

4.1 Authentication Basics

OPSS uses WebLogic 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.

Authentication providers include the DefaultAuthenticator; external LDAP stores; and DBMS to host data for enterprise applications. For a full list of authenticator providers, see chapter 4, Authentication Providers in *Oracle Fusion Middleware Developing Security Providers for Oracle WebLogic Server*.

JavaEE applications use WebLogic authentication providers; JavaSE 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, WLST commands, or the appropriate tool depending on the authentication provider(s) configured in your domain.

This section covers the following topics:

- Oracle WebLogic Authenticators
- Additional Authentication Methods
- Using an LDAP Authenticator

4.1.1 Oracle WebLogic Authenticators

OPSS includes several authentication providers. 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 LDAP authenticators, see Using an LDAP Authenticator.

> **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.*

4.1.2 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*.

4.1.3 Using an LDAP Authenticator

This section explains the initialization of the identity store service and how to access an LDAP authenticator programmatically.

Oracle WebLogic Server offers several LDAP-based authenticators. For a choice of available LDAP servers, see Section 5.1, "Supported LDAP Servers." The 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 JavaSE applications, see Section 16.2.2, "Configuring an LDAP Identity Store in JavaSE Applications."

4.1.3.1 Configuring the 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 the default values that OPPS 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.

4.2 Policy Store Basics

A Java 2 policy specifies the permissions granted to signed code loaded from a given location. An example of a Java 2 grant entry is found in Protection Domains and Security Policies.

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. An example of a JAAS grant entry is found in Principals and Subjects.

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 JavaEE 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, JavaEE application roles (configured in files `web.xml` or `ejb-jar.xml`) get mapped to enterprise users and groups and used by application-specific policies.

Policy Store Types

A policy store can be file-based or LDAP-based. A file-based policy store is an XML file, and this store is the out-of-the-box policy store provider. An LDAP-based policy store can use either of the following LDAP servers: Oracle Internet Directory or Oracle Virtual Directory (with a local store adapter, or LSA).

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 with Fusion Middleware Control, they can be automatically migrated into the domain policy store. For details about this feature, see Section 8.3.1, "Migrating Application Policies with Fusion Middleware Control." By default, the domain policy store is file-based.

Reassociation of policies is supported from a file-based store or from an LDAP-based (Oracle Internet Directory or Oracle Virtual Directory) store only. The reassociation of domain policies from an LDAP-based policy store using any other type of LDAP server provider is not supported. For details, see Section 8.2, "Reassociating the Domain Policy Store."

> **Note:** All permission classes must be specified in the system classpath.

4.3 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-based or LDAP-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. An LDAP-based credential store can use either of the following LDAP servers: Oracle Internet Directory or Oracle Virtual Directory.

Scope, Migration, and Reassociation

An application can use either the domain credential store or, only during development, its own wallet-based credential store. The domain credential store can be wallet-based (by default) or LDAP-based.

The migration of application credentials to the domain credential store can be configured to take place automatically when the application is deployed. For details, see Section 9.4.1, "Migrating Application Credentials with Fusion Middleware Control."

Domain credentials can also be reassociated from one type of store to another. For details, see Section 9.3, "Reassociating the Domain Credential Store."

5

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 supported LDAP servers, 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 Servers
- Management Tools
- Packaging Requirements
- Example Scenarios
- Other Scenarios

5.1 Supported LDAP Servers

Oracle Platform Security Services supports the following LDAP servers and repositories:

- iPlanet, WebLogic DefaultAuthenticator, Active Directory, Novell, and OpenLDAP.
- Oracle Internet Directory (versions 10.1.4.3 or 11g only) and Oracle Virtual Directory (version 11g only).

> **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=dsaconfig,cn=configsets,cn=oracle internet directory
> changetype: modify
> add: orclinmemfiltprocess
> orclinmemfiltprocess: (objectclass=orcljaznpermission)
> orclinmemfiltprocess: (objectclass=orcljazngrantee)
> EOF
> ```

WebLogic authenticators store identities (users and groups) in a store using any of the above servers. For details about LDAP authenticators and tested LDAP servers, see section Configuring LDAP Authentication Providers in *Oracle Fusion Middleware Securing Oracle WebLogic Server*.

Policies and credentials can be stored in a store using *only* Oracle Internet Directory or Oracle Virtual Directory; moreover, when both the policy and credential stores are LDAP-based, they must use the kind of server. The only supported back-end for an Oracle Virtual Directory is LSA (local store adapter).

The DefaultAuthenticator is available out-of-the-box and its use is recommended only in developing environments and for no more than a thousand entries.

5.2 Management Tools

The tools available to an administrator are the following:

- WebLogic Administration Console
- Oracle Enterprise Manager Fusion Middleware Control
- WLST commands and WLST scripting
- 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.

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 4.1.3, "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 and Credentials

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

To manage policy and credential data, use either Fusion Middleware Control as explained in Managing Policies with Fusion Middleware Control and Managing Credentials with Fusion Middleware Control, or the command-line utility WLST, as explained in Managing Policies with WLST Commands and Managing Credentials with WLST Commands.

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 or WLST

Changes to policies or credentials do not require server restart; changes to the `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 sections Migrating Application Policies with Fusion Middleware Control and Migrating Application Credentials with Fusion Middleware Control.

5.3 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 15.3, "Packaging a JavaEE Application Manually."

For information about deployment, see Chapter 7, "Deploying Secure Applications."

> **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.

5.4 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 domain 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.

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 using, configuring, and managing policies, see Chapter 8, "OPSS Authorization and the Policy Store."

For details about using, configuring, and managing credentials, see Chapter 9, "Configuring the Credential Store."

Common Scenario 1

This scenario describes a JavaEE 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 JavaEE 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 JavaEE 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 Virtual 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 3.3, "The Authenticated Role," and Section 3.4, "The Anonymous User and Role."

Common Scenario 3

This scenario describes a JavaEE 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 JavaEE 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 3.3, "The Authenticated Role," and Section 3.4, "The Anonymous User and Role."

For details about CSF APIs, see Section 17.1, "About the Credential Store Framework API."

5.5 Other Scenarios

The following scenarios differ from the common scenarios in that the application uses an authenticator othe then the DefaultAuthenticator (typically used in the application development phase) or some provided 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 19, "Developing with the User and Role API."

For details about CSF APIs, see Section 17.1, "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 8.3.1, "Migrating Application Policies with Fusion Middleware Control," and Section 9.4.1, "Migrating Application Credentials with Fusion Middleware Control."

For details about reassociation, see Section 8.2.1, "Reassociating Domain Stores with Fusion Middleware Control."

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 JavaSE application using OPPS APIs.

Authentication: The application 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 domain credential store.

Part II

Basic OPSS Administration

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

-
-

6

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 WLST Commands

For advanced administrator tasks, see Appendix E, "Administration with WLST Scripting and MBean Programming."

6.1 Choosing the Administration Tool According to Technology

The three basic tools available to a security administrator are Oracle Enterprise Manager Fusion Middleware Control, the Oracle WebLogic Administration Console, 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 an application is whether the application uses just container-managed security (JavaEE 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.

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

The recommended tool to develop applications is Oracle JDeveloper 11g. This tool helps the developer configure file-based identity, policy, and credential stores through specialized editors and UI gadgets. 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*.

6.2 Basic Security Administration Tasks

Table 6–1 lists some basic security tasks and the tool(s) used to execute them. Recall that the tool chosen to configure and manage application security depends on the type of the application: for JavaEE applications, which use just container-managed security, use the Oracle WebLogic Administration Console; for Oracle ADF applications, which use fine-grained authorization, use Fusion Middleware Control.

Manual settings without the aid of the tools listed below are not recommended. For information about using the Oracle WebLogic Administration Console to perform the tasks marked with an X, see list of links following the table.

Table 6–1   Basic Administrative Security Tasks

Task	Use Fusion Middleware Control Security Menu	Use Oracle WebLogic Administration Console
Configure WebLogic Domains		X
Configure WebLogic Security Realms		X
Manage WebLogic Domain Authenticators		X
Enable SSO for MS clients, Web Browsers, and HTTP clients.		X
Manage Domain Administrative Accounts		X
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	
Enable JAAS in Oracle ADF Application	Security Provider Configuration	
Map application to enterprise groups for Oracle ADF Application	Application Roles or Application Policies	
Manage systemwide policies for Oracle ADF Applications	System Policies	
Configure OPSS Properties	Security Provider Configuration	
Reassociate Domain 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*.

> **Note:** OPSS does not support automatic backup or cloning of server files. It is recommended that the server administrator periodically back up all server configuration files, as appropriate, using third-party tools.

6.2.1 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 17.5, Cloning a Middleware Home or an Oracle Home, in *Oracle Fusion Middleware Administrator's Guide*.

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

6.3 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 8.2.1, "Reassociating Domain Stores with Fusion Middleware Control."

- Post-installation and before deploying applications, define OPSS properties. For details, see Section 8.5, "Configuring Other Artifacts with Oracle Fusion Middleware Control."

- 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 7.3, "Deploying Oracle ADF Applications to a Test Environment"
- Section 8.3.1, "Migrating Application Policies with Fusion Middleware Control"
- Section 9.4.1, "Migrating Application Credentials with Fusion Middleware Control"

- For each application after it is deployed:
  - Manage application policies. For details, see Section 8.4.1.1, "Managing Application Policies."
  - Manage credentials; for details, see Section 9.5.1, "Managing Credentials with Fusion Middleware Control."
  - Specify the mapping from application roles to users, groups, and application roles. For details, see Section 8.4.1.2, "Managing Application Roles."

- For the domain, manage system policies; for details see Section 8.4.1.3, "Managing System Policies."

- For the domain, manage OPSS properties; for details see Section 8.5, "Configuring Other Artifacts with Oracle Fusion Middleware Control."

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.

6.4 Typical Security Practices with the Administration Console

This section addresses security-related operations and some basic administrative operations.

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 operations 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*.

6.5 Typical Security Practices with WLST Commands

WLST is a command-line interface that allows the scripting and automation of administration tasks, including domain configuration and application deployment.

The following is the complete lists of security-related WLST commands documented in this guide. For a complete list of all WLST commands, see *Oracle Fusion Middleware WebLogic Scripting Tool Command Reference*.

Policy-Related Commands

For details on the following commands, see Section 8.4.2, "Managing Policies with WLST Commands."

- createAppRole

- deleteAppRole

- grantAppRole

- revokeAppRole

- listAppRoles

- listAppRolesMembers

- grantPermission

- revokePermission

- listPermissions

- deleteAppPolicies

Credential-Related Commands

For details on the following commands, see Section 9.5.2, "Managing Credentials with WLST Commands."

- listCred

- updateCred

- createCred

- deleteCred

- modifyBootStrapCredential

Other Commands

- migrateSecurityStore. For details, see Section 8.3.2, "Migrating Policies with the Command migrateSecurityStore," and Section 9.4.2, "Migrating Credentials with the Command migrateSecurityStore."

- reassociateSecurityStore. For details, see Section 8.4.2, "Managing Policies with WLST Commands," and Section 9.5.2, "Managing Credentials with WLST Commands."

7

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, or Oracle JDeveloper.

The tool recommended to deploy it depends on the application type and whether the application is in the developing phase or in a post-development phase. The recommendations stated in this chapter apply to Oracle ADF applications and to JavaEE 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.

This chapter focuses on administrative tasks performed at deployment of an Oracle ADF or pure JavaEE application. The last section explains the packaging requirements to secure JavaEE 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 JavaEE 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 an overview of the development cycle, see Section 14.1.1, "The Development Cycle."

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 15, "Manually Configuring JavaEE Applications to Use OPSS."

7.1 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:
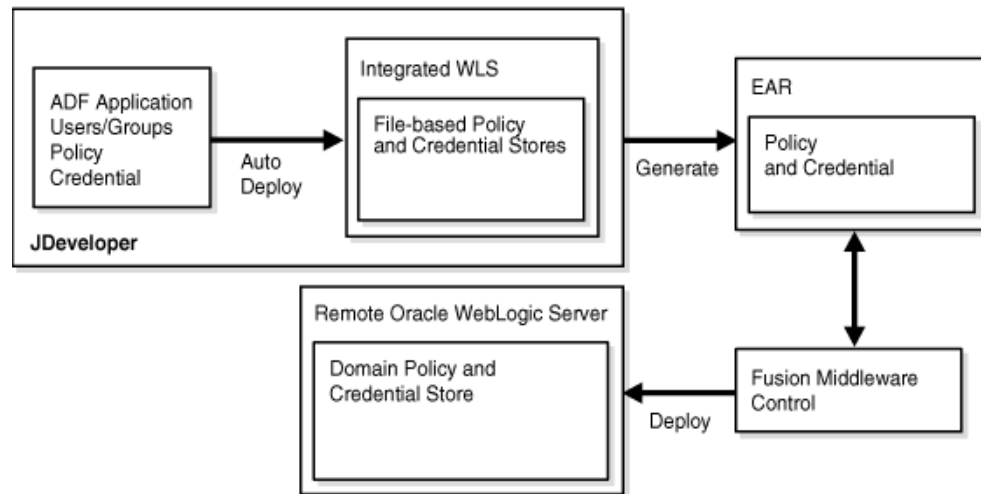
7.2 Selecting the Tool for Deployment

The types of application we consider in this chapter are JavaEE applications, which are further categorized into pure JavaEE applications and Oracle Fusion Middleware ADF applications. The distinction of these two kinds of JavaEE applications is explained in sections Section 2.5.1, "Scenario 1: Enhancing Security in a JavaEE Application," and Section 2.5.2, "Scenario 2: Securing an Oracle ADF Application."

Table 7–1 lists the tool used to deploy a developed application according to its type.

Table 7–1   Tools to Deploy Applications after Development

Application Type	Tool to Use
Pure JavaEE Application	Oracle WebLogic Administration Console, Fusion Middleware Control, or WLST command. The recommended tool is Oracle WebLogic Administration Console.
Oracle ADF Application	Fusion Middleware Control or WLST command. The recommended tool is Fusion Middleware Control.

7.2.1 Deploying JavaEE 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 JavaEE application that uses OPSS with Fusion Middleware Control, specifically, on the options you find in the page **Configure Application Security** at the third stage of the deploy settings.

The appearance of this page depends according to what is packaged in the EAR fie, 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 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 domain 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 domain 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 the following two most common scenarios:
  >
  > - Resolving inconsistent specfications 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.

- I f 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 domain 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**.

7.3 Deploying Oracle ADF Applications to a Test Environment

An Oracle ADF application is a JavaEE 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

7.3.1 Deploying to a Test Environment

The security options available at deployment are explained in Deploying JavaEE 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 domain policy store when the application is deployed.

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

Credential Management

- Application-specific credentials packed with the application are automatically migrated to the domain 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 15.4.7, "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 8.2.1, "Reassociating Domain Stores with Fusion Middleware Control."

7.3.1.1 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 8.4.1.1, "Managing Application Policies."

- Create additional application roles or customize existing ones. For details, see Section 8.4.1.2, "Managing Application Roles."

- Manage system policies. For procedure details, see Section 8.4.1.3, "Managing System Policies."

- Manage credentials. For procedure details, see Section 9.5.1, "Managing Credentials with Fusion Middleware Control."

> **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
> domain 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 it life as being packaged with an
> application, but when the application is undeployed credentials are
> *not* removed.

7.4 Deploying Standard JavaEE Applications

There are two ways to secure JavaEE applications that do not use OPSS but that use
standard Java authorization: administratively, with the Administration Console or a
WLST script (command-line or script mode); or programmatically, with deployment
descriptors.

A JavaEE 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
JavaEE Applications, in *Oracle Fusion Middleware Administrator's Guide*.

For details about deploying applications with the 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

7.5 Migrating from a Test to a Production Environment

The recommendations that follow apply only to JavaEE applications using JAAS authorization, such as Oracle Application Development Framework, Oracle SOA, and Oracle WebCenter applications, and they do not apply to JavaEE applications using standard authorization. For deploying the latter, see Deploying Standard JavaEE 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 6.2.1, "Setting Up a Brand New Production Environment."

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

7.5.1 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 configuration may include:

- The identity store configuration, including the provisioning of users (using the WebLogic Administrator Console).
- A SAML configuration of the issuer's list.
- 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*.

7.5.1.1 Migrating Identities Manually

Identity data can be migrated manually from a source repository to a target repository using the WLST command `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 command 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.

The commands listed below can be run in interactive mode or in script mode. In interactive mode, you enter the command at a command-line prompt and view the

response immediately after. In script mode, you write commands in a text file (with a py file name extension) and run it without requiring input, much like the directives in a shell script.

> **Important:** Before invoking a security-related WLST command in a shell, you must run the script `wlst.sh`, as illustrated in the following sample:
>
> ```
> > sh $ORACLE_HOME/common/bin/wlst.sh
> ```
>
> This ensures that the required JARs are added to the classpath. Failure to run the above script in a new shell renders the WLST commands unusable.
>
> Before running an online command, connect to the server as follows:
>
> ```
> >java weblogic.WLST
> >connect('servername', 'password', 'localhost:portnum')
> ```

Script and Interactive Modes Syntaxes

To migrate identities, 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 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 command 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 can be XML-based or LDAP-based backed by an Oracle Internet Directory server. No other destination is supported.

- `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 command 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.

7.5.2 Migrating Policies and Credentials at Deployment

In a production environment, it is strongly recommended that the policy and credential stores be reassociated to an LDAP-based repository; if the test policy and credential stores were also LDAP, the production LDAP is assumed to be distinct from the test LDAP. For details on how to reassociate stores, see Section 8.2.1, "Reassociating Domain Stores 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.

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 command `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 15.4.1, "Parameters Controlling Policy Migration."

7.5.2.1 Migrating Policies Manually

By default, the command `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. There is, however, a way to specify that the GUIDs be preserved by the migration. For details, see argument `preserveAppRoleGuid` in Section 8.3.2, "Migrating Policies with the Command migrateSecurityStore."

Migrating policies manually with the command `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 sources for LDAP and XML policies, and of a destination for LDAP policies:

```
<?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>Bootstrap credential provider</description>
 </serviceProvider>

 <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>
</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>

                    <!-- 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>

                    <!-- Bootstrap credentials to access source and destination LDAPs -->
               <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="LDAPdestinationContext">
               <serviceInstanceRef ref="policystore.ldap.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 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 the command `migrateSecurityStore` assume that:

- The file `t2p-policies.xml` is located on the target system in the directory where the command is run.

- The directory structure of LDAP system policies in the test and production environments should be *identical*. If this is not the case, before using the command, 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")
```

Similarly, 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="XMLsou
rceContext",dst="LDAPdestinationContext")
```

7.5.2.2 Migrating Credentials Manually

The command `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 the command `migrateSecurityStore` requires assembling a configuration file where the source and destination are specified.

Since the command `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 sources for LDAP and XML credentials, and of a destination for LDAP credentials:

```
<?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>
</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. 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="credstore.type"/>
 <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>

 <!-- 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. 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="credstore.type"/>
 <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>

 <!-- Bootstrap credentials to access source and destination LDAPs -->
 <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="LDAPdestinationContext">
<serviceInstanceRef ref="credential.ldap.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 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 the command `migrateSecurityStore` assume that the file `t2p-credentials.xml` is located on the target system in the directory where the command 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
ourceContext",dst="LDAPdestinationContext")
```

Similarly, 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="FileS
ourceContext",dst="LDAPdestinationContext")
```

7.5.2.3 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 command `migrateSecurityStore` would take an unacceptable amount of time).

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="security.principal" value="cn=orcladmin"/>
 <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*.

7.5.3 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 WLST command `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 WLST command `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 12.2.2, "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 12.5.5, "Importing and Exporting Data."

Part III

Advanced OPSS Administration

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

8

OPSS Authorization and the Policy Store

The domain policy store is the repository of system and application-specific policies. In a given domain, there is one store that stores all policies (and credentials) that all applications deployed in the domain may use.

For an introduction to the main features of the policy store, see Policy Store Basics; for details about Java 2 and JAAS, see Java Security Model and Java Authentication and Authorization Service.

This chapter is divided into the following sections:

- Configuring a Domain to Use an LDAP-Based Policy Store
- Reassociating the Domain Policy Store
- Migrating Policies to the Domain Policy Store
- Managing the Domain Policy Store
- Configuring Other Artifacts with Oracle Fusion Middleware Control
- Configuring LDAP-Based Policy Stores

For further details about JavaEE and WebLogic Security, see section J2EE and WebLogic Security in *Oracle Fusion Middleware Understanding Security for Oracle WebLogic Server*.

> **Note:** When a domain is setup to use policies based on the Policy Store, as explained in this chapter, then JACC policies and the Java Security Manager become unavailable on all managed servers in that domain.

> **Important:** All permission classes used in policies must be included in the classpath, so the policy provider can load them when a service instance is initialized.

8.1 Configuring a Domain to Use an LDAP-Based Policy Store

An LDAP-based policy store is typically used and recommended in production environments. The LDAP servers supported in this release are the Oracle Internet Directory and the Oracle Virtual Directory.

The characteristics of the domain policy store and the domain credential store are tightly correlated: both must be of the same kind (file-based or LDAP-based), and in

case of LDAP-based repositories, both must use the same type of LDAP server. Out-of-the-box, the policy and credential stores are file-based.

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

For a detail list of properties that can be specified in a service instance, see Appendix F.2.4, "Generic LDAP Properties."

The information is presented in the following sections:

- Multiple-Node Server Environments
- Prerequisites to Using an LDAP-Based Policy Store
- Cataloging Oracle Internet Directory Attributes

8.1.1 Multiple-Node Server Environments

In domains where several server instances are distributed across multiple machines, it is highly recommended that the domain policy and credential stores be LDAP-based, so that all policy and credential data is kept and maintained in one centralized store.

Typically, applications do not change policy 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, any such changes should always be performed in the domain administration server (and not by 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.

To summarize, in a multiple-node server environment, it is highly recommended that:

- Both the domain policy and credential stores be centralized in a LDAP-based store and configured in the administration server.
- Or, if they are file-based, then local changes to policy or credential data be performed *only* by the domain administration server to ensure that they are correctly propagated from the administration server to all managed servers in the domain.

8.1.2 Prerequisites to Using an LDAP-Based Policy Store

In order to ensure the proper access to an LDAP server directory (Oracle Internet Directory or Oracle Virtual Directory), you must set a node in the server directory.

Fusion Middleware Control automatically provides bootstrap credentials in the file cwallet.sso when it is used to reassociate to an LDAP-based repository. To specify them manually, see section Section 15.4.7, "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 -c -v -f
jpstestnode.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
-b "cn=jpsroot" objectclass="orclContainer"
```

4. If the LDAP server is Oracle Internet Directory, 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*.

> **Note:** If you are using an LDAP-based Oracle Internet Directory policy store, you can skip the following section (about Oracle Virtual Directory configuration) and continue on directly to reassociating the policy store as explained in Reassociating the Domain Policy Store.

Configuring an Oracle Virtual Directory Server with a Local Store Adapter

Oracle Virtual Directory provides Internet and industry-standard LDAP and XML views of existing enterprise identity information, without synchronizing or moving data from its native locations. The configuration explained in this section is performed by the Oracle Virtual Directory administrator.

To configure Oracle Virtual Directory to use a file to store the policies with an LSA, proceed as follows:

1. Start the Oracle Directory Services Manager.

2. Select the tab **Adapter**.

3. Choose to create an adapter, to display the dialog **New Adapter Wizard** illustrated in the following graphic:
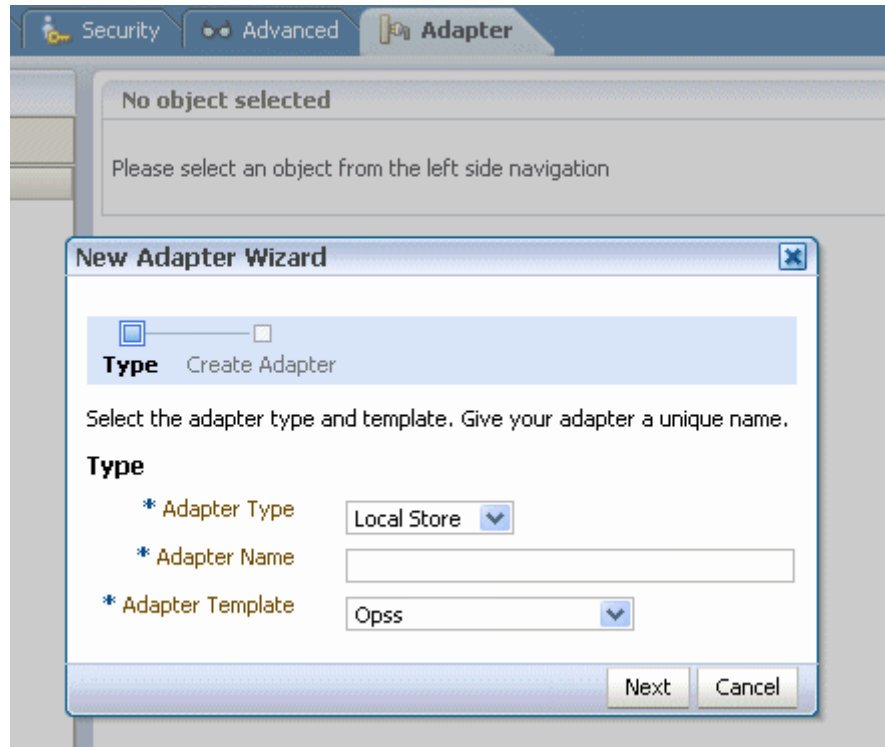
4. In the **Type** phase of that wizard, (a) set Adapter Type to **Local Store**; (b) set the Adapter Name to a unique name; and (c) set Adapter Template to **Opss**.

5. Click **Next**, to display the **Settings** phase of the **New Local Store Adapter Wizard** as illustrated in the following graphic:
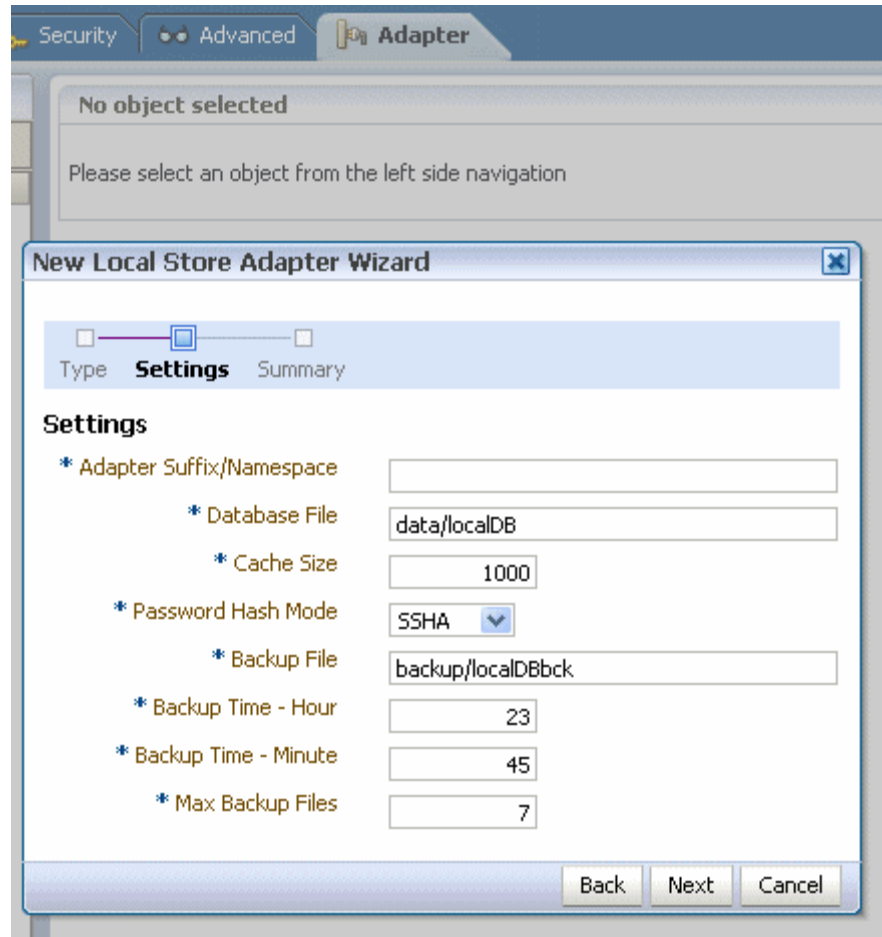
6. In the **Adapter Suffix/Namespace** text box, enter the root name for the adapter you are creating (this value should match the dn of the node in the OVD). Retain default values in all other fields.

7. Click **Next** to view all settings in the **Summary** phase of the wizard, and, when satisfied with them, commit your input.

For more details, see section 2.4, Understanding the Local Store Adapter, in *Oracle Fusion Middleware Administrator's Guide for Oracle Virtual Directory.*

8.1.3 Cataloging Oracle Internet Directory Attributes

An Oracle Internet Directory attribute used in a search filter must be indexed.

For detailed instructions on how to manage attribute catalogs and to identify whether an attribute is indexed, see the following sections in *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*:

- Adding an Index to a New Attribute by Using Oracle Directory Services Manager

- Adding an Index to an Existing Attribute by Using Oracle Directory Services Manager

- Dropping an Index from an Attribute by Using Oracle Directory Services Manager

- Indexing an Attribute for Which Data Exists by Using Oracle Directory Services Manager

■ Creating and Dropping Indexes from Existing Attributes Using Catalog

The command `ldapmodify`, who se syntax is illustrated below, can also be used to index attributes specified in an LDIF file:

```
>ldapmodify -h <host> -p <port> -D <bind DN> -w <bind password> -v -f <catalogue
modify ldif file name>
```

For example, the above command can be used with the following sample LDIF file to catalog the attributes `createtimestamp` and `modifytimestamp`:

```
dn: cn=catalogs
changetype: modify
add: orclindexedattribute
orclindexedattribute: modifytimestamp
orclindexedattribute: createtimestamp
```

Each of the following Oracle Internet Directory attributes must be indexed:

orclrolescope

orclassignedroles

orclApplicationCommonName

orclAppFullName

orclCSFAlias

orclCSFKey

orclCSFName

orclCSFDBUrl

orclCSFDBPort

orclCSFCredentialType

orclCSFExpiryTime

modifytimestamp

createtimestamp

orcljpsassignee

8.2 Reassociating the Domain Policy Store

Reassociating the policy store consists in migrating policy data from a file- or LDAP-based repository to an LDAP-based repository, that is, reassociation changes the repository preserving the integrity of the data stored. For each policy in the source policy store, reassociation searches the target LDAP directory and, if it finds a match, it updates the matching policy as appropriate. If none is found, it simply migrates the policy as is.

At any time, after a domain policy store has been instantiated, a file- or LDAP-based policy store can be reassociated into an LDAP-based policy store storing the same data. To support it, the domain has to be configured, as appropriate, to use and LDAP policy store.

Reassociation is carried out using either Fusion Middleware Control or the WLST command `reassociateSecurityStore`. This operation is typically performed when setting a domain to use the an LDAP-based domain store instead of the out-of-the-box file-based store.

> **Important:** The repositories for the policy store and the credential store must both be file-based or both LDAP-based. Moreover, in the LDAP-based case, both stores must use the same kind of LDAP server. To preserve this coherence, note that reassociating one store implies reassociating the other one.

8.2.1 Reassociating Domain Stores with Fusion Middleware Control

The reassociation of security stores migrates policies and credentials from one repository to another, thus reconfiguring those security store providers; this reconfiguration is typically performed when, for example, transitioning from a test environment to a production environment. OPSS supports LDAP- to LDAP-based or file- to LDAP-based reassociation, where the LDAP servers are Oracle Internet Directory or Oracle Virtual Directory. LDAP- to file- based reassociation, however, is not supported.

This section explains the steps you follow to reassociate policies and credentials to an LDAP-based storage using Oracle Internet Directory or Oracle Virtual Directory servers.

To reassociate domain stores manually, use the WLST command reassociateSecurityStore.

For information about other uses of the **Security Provider Configuration** page, see Configuring Other Artifacts with Oracle Fusion Middleware Control.

> **Notes:**
>
> - The procedure below reassociates *both* the policy and the credential stores to the same LDAP-based store.
>
> - Before starting a reassociation ensure that you have accomplished the requisites in Prerequisites to Using an LDAP-Based Policy Store.
>
> - If reassociation is to use a one-way SSL, follow the instructions in Setting Up a One- Way SSL Connection *before* initiating the reassociation (with either Fusion Middleware Control or the WLST command reassociateSecurityStore).
>
> - After reassociation, to secure access to the root node of the Oracle Internet Directory store, follow the instructions in Securing Access to Oracle Internet Directory Nodes.

To reassociate both the policy and the credential stores in a given domain to a new LDAP repository with Fusion Middleware Control, proceed as follows:

1. Log in to Fusion Middleware Control and navigate to *Domain* > **Security** > **Security Provider Configuration**, to display the **Security Provider Configuration** page. This page is partially shown in the following graphic:

The table underneath the button **Change Association** shows the characteristics of the current provider configured in the domain.

2. Click the button **Change Association** in the area **Policy and Credential Store Providers** to display the page **Set Security Provider**. In this page, enter details and connection information about the target LDAP server.

3. Select the type of LDAP server from the pull-down list: if Oracle Virtual Directory is configured with the local store adapter, select Oracle Virtual Directory; otherwise, select Oracle Internet Directory.

4. Enter the host name and port number of your target LDAP server.

5. 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, be sure to follow the instructions in Setting Up a One- Way SSL Connection *before* completing this step. Among other things, that set up 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.

6. In the text box **User DN**, enter the full distinguished name, a string containing between 1 and 256 characters. For example, cn=orcladmin,dc=us,dc=oracle,dc=com.

7. In the box **Password**, enter the user password, also a string containing between 1 and 256 characters.

8. 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 I.10, "Failure to Establish an Anonymous SSL Connection."

9. In the **LDAP Root Node Details** area, enter the root DN in the box **JPS Root DN**. This node denotes the top of the tree that contains the data in the LDAP repository.

The **WebLogic 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 I.2, "Reassociation Failure."

10. Optionally, enter properties for the service instances in the **Policy Store Properties** and **Credential Store Properties** areas: click **Add** and then, in the **Add New Property** 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 shown in the following excerpt:

    ```
    <serviceInstance name="myNewLDAPprovider" provider="someProvider"
      ...
      <property name="foo" value="bar"/>
      ...
    </serviceInstance>
    ```

11. 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 **Policy and Credential Store Providers** area is modified to show, as the current provider, the provider you have just specified.

12. Restart the Oracle WebLogic Server. Changes do not take effect until the server has been restarted.

Reassociation modifies the domain configuration file `$DOMAIN_HOME/config/fmwconfig/jps-config.xml`: it deletes any configuration for the old provider and inserts a configuration for the new store provider, so, when the server is restarted the new configuration takes effect.

8.2.1.1 Setting Up a One- Way SSL Connection

This section describes how to set up a one-way SSL channel between the Oracle WebLogic server and a target LDAP Oracle Internet Directory store to reassociate domain stores. This setup should be in place *before* using Fusion Middleware Control or the WLST command reassociateSecurityStore.

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.

Setting Up the WebLogic Server

To set up the server whose domain includes the stores to be reassociated, proceed as follows:

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. Use the WebLogic Administration Console to set, in the appropriate domain, a truststore that points to the file `myKeys.jks`, as follows:

   1. Navigate to **Environment > Servers**.

   2. Select the name of the administration server in your domain.

   3. Select the tab **Keystore**.

   4. Change the keystore mode from Demo Identity and Demo Trust to **Custom Identity and Custom Trust**.

   5. In the **Trust** section enter data as follows: in the field **Custom Trust Keystore**, enter the absolute path name of the keystore file **myKeys.jks**; in the field **Keystore Type**, enter JKS; in the fields **Custom Trust Keystore Passphrase** and **Confirm Custom Trust Keystore Passphrase**, enter the keystore password.

3. Modify the script (typically startWebLogic.sh) that starts your server to include a line like the following, and then restart the server:

   ```
   -Djavax.net.ssl.trustStore=<absolute path name to file myKeys.jks>
   ```

8.2.1.2 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*.

8.2.2 Reassociating Domain Stores with the Command reassociateSecurityStore

Domain stores can also be manually reassociated with the WLST command `reassociateSecurityStore`. For details about this command, see reassociateSecurityStore.

8.3 Migrating Policies to the Domain Policy Store

A domain includes one and only one policy store. Applications can specify their own policies, but these are stored as policies in the domain policy store when the application is deployed to a server. Thus, all applications deployed in a domain use a common policy store, the domain policy store. The domain 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>`.

During application development, an application specifies its own policies, and these can be migrated to the domain policy 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 domain policy store is performed by the domain administrator.

These topics are explained in the following sections:

- Migrating Application Policies with Fusion Middleware Control
- Migrating Policies with the Command migrateSecurityStore

8.3.1 Migrating Application Policies with Fusion Middleware Control

Application policies are specified in the application file `jazn-data.xml` and can be migrated to the domain 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 domain 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 7.5.2, "Migrating Policies and Credentials at Deployment."

8.3.2 Migrating Policies with the Command migrateSecurityStore

Application-specific policies or system policies can be migrated manually from a source repository to a target repository using the WLST command `migrateSecurityStore`.

This command 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 command `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 7.5.2.3, "Migrating Large Volume Policy and Credential Stores.".

For further details about WLST commands and their syntax, see section Overview of WSLT Command Categories, in *Oracle Fusion Middleware WebLogic Scripting Tool Command Reference*.

To migrate all policies (system *and* application-specific, for all applications) 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 command is run. Typically, this configuration file is created just to be used with the command and serves no other purpose. This files 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. See second example below.

- `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 command determines the locations of the source and the target repositories involved in the migration.

To migrate *just* system policies, 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, 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]
```

```
migrateSecurityStore(type="appPolicies", configFile="jpsConfigFileLocation",
src="srcJpsContext", dst="dstJpsContext", srcApp="srcAppName",
[dstApp="dstAppName"], [overWrite="trueOrfalse"],
[migrateIdStoreMapping="trueOrfalse"])
```

The meanings of the arguments `configFile`, `src`, and `dst` are identical to the previous cases. The meanings of other four arguments are 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.

If the input does not follow the syntax requirements above, the command 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.

8.3.2.1 Examples of Use

For complete examples illustrating the use of this command, see Section 7.5.2.1, "Migrating Policies Manually."

8.4 Managing the Domain Policy Store

The following sections explain how an administrator can manage policies using either Fusion Middleware Control or WLST commands. Typical managing 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.

Only a user with the appropriate permissions, such as the domain administrator, can access data in the domain policy store.

- Managing Policies with Fusion Middleware Control

- Managing Policies with WLST Commands

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 I.12, "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 I.6, "Failure to Grant or Revoke Permissions - Case Mismatch."

> **Important Point 3:** Strings in a policy store cannot contain any special ASCII character in the range [0, 15].
>
> For other issues regarding the use of characters in policy stores, see Section I.16, "Characters in Policies."

8.4.1 Managing Policies with Fusion Middleware Control

Fusion Middleware Control allows the management of 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

8.4.1.1 Managing Application Policies

This section explains the steps you follow to manage the policies of a deployed application with Fusion Middleware Control, such as, for example, to change the permission grants of an existing application role. To manage system policies, see Managing System 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 classpath.

1. Log in to Fusion Middleware Control and navigate to *Application* > **Security** > **Application Policies**, to display the **Application Policies** page for the selected Application. This page is partially shown in the following graphic:

The area **Policy Store Provider** is read-only and, when expanded, displays the policy store provider currently in use in the domain where the application is deployed.

> **Note:** If the page does not initially display policies and roles, click the blue button to display all items.

2. To display existing application policies matching a given principal or permission, expand the **Search** area, enter the data to match (a principal or a permission or both), and click the blue button. The results of the search are displayed in the table at the bottom of the page.

> **Note:** If, and only if, the selected application uses a policy stripe name *different* from the application name, the box **Select Application Stripe to Search** is displayed. In this case, check the box and select a stripe name from the pull-down list.

3. To create an application policy, click **Create** to display the **Create Application Grant** page. The top area **Grant Details** displays read-only information about the application.

   1. To add permissions to the policy being created, click **Add** in the **Permissions** area to display the **Add Permission** dialog.

      Using the **Search** to identify permissions matching a class or resource name, determine the **Permission Class** and **Resource Name** of the permission. Optionally, use the **Customize** area to further qualify the permission.

When finished, click **OK** to return to the **Create Application Grant** page. The permission you created is displayed in the table in the **Permissions** area.

2. To add a user to the policy being created, click the button **Add User** in the **Grantee** area to display the dialog **Add User**.

Using the **Search**, identify users names matching a string; the result of the query is displayed in the **Available Users** box.

Using the various buttons, move the users you want to grant permissions from the **Available Users** box to the **Selected Users** box.

When finished, click **OK** to return to the **Create Application Grant** page. The users you selected are displayed in the table in the **Grantee** area.

3. To add a role to policy being created, click the button **Add Role** in the **Grantee** area to display the dialog **Add Role**.

Using the **Search**, identify role names matching a type or name; the result of the query is displayed in the **Available Roles** box.

Using the various buttons, move the roles you want to grant permissions from the Available Roles box to the Selected Roles box.

When finished, click **OK** to return to the **Create Application Grant** page. The roles you selected are displayed in the table in the **Grantee** area.

4. At any point you can remove a selected item using the **Delete** button.

5. 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 all the permissions are automatically filled in with the data extracted from the policy you selected.

3. Modify those values and enter users or roles, as appropriate, as explained in the substeps of step 3 above, and then click **OK**.

8.4.1.2 Managing Application Roles

This section explains the steps you follow to manage the roles of a deployed application with Fusion Middleware Control. Management operations that control the access a user or group has to resources, include creating application roles, mapping an application role to users, groups, or other application roles, and managing the members in application roles.

1. Log in to Fusion Middleware Control and navigate to *Application* > **Security** > **Application Roles**, to display the **Application Roles** page. This page is partially shown in the following graphic:

The area **Policy Store Provider** is read-only and, when expanded, displays the policy store provider currently in use in the domain where the application is deployed.

> **Note:** If the page does not initially display application roles, click the blue button to display all items.

2. To display application roles matching a given name, enter the data to match in the box **Role Name**, and click the blue button. The results of a query are displayed in the table at the bottom of the page.

   To redisplay the table of all current application policies, leave the **Role Name** blank and click the blue button.

   > **Note:** If, and only if, the selected application uses a policy stripe name *different* from the application name, the box **Select Application Stripe to Search** is displayed. In this case, check the box and select a stripe name from the pull-down list.

3. To create an application role, click **Create** to display the **Create Application Role** page. Note that you must not enter data in all areas at once. For example, you could create a role by entering the role 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:

   1. The name of the role, in the text box **Role Name**.

   2. Optionally, the name to display for the role, in the text box **Display Name**.

3. 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:

1. Click **Add Role**, to display the **Add Role** dialog.

2. In this dialog, select the type of the role to be added from the menu next to **Role Type**; optionally, enter a role name in the text box **Role Name**; and then click the blue button to display, in the box **Available Roles**, the roles that match the entered criteria.

3. Select roles from the box **Available Roles**, as appropriate, and use the buttons in between the boxes to move them to the box **Selected Roles**.

4. When finished, click **OK** to return to the **Create Application Role** page. The selected roles are displayed in the table **Roles**.

In the area **Users**, specify the members of the role being created:

1. Click **Add User**, to display the **Add User** dialog.

2. The box **Available Users** displays all the available users. To display just the users with names matching a given string, enter that string in the box **User Name** and click the blue button.

3. Select users from the box **Available Users**, as appropriate, and use the buttons in between the boxes to move them to the box **Selected Users**.

4. When finished, click **OK** to return to the **Create Application Role** page. The selected users are displayed in the table **Users**.

4. Click **OK** to effect the role creation (or update) and to return to the **Application Roles** page. The role just created is displayed in the table at the bottom of that page.

5. 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 all the roles and users are 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 3.2.1, "Permission Inheritance and the Role Hierarchy."

8.4.1.3 Managing System Policies

This section explains the steps you follow to manage system policies for a domain with Fusion Middleware Control. To manage application policies, see Managing Application Policies.

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.

1. Log in to Fusion Middleware Control and navigate to *Domain* > **Security** > **System Policies** to display the page **System Policies**. This page is partially shown in the following graphic:
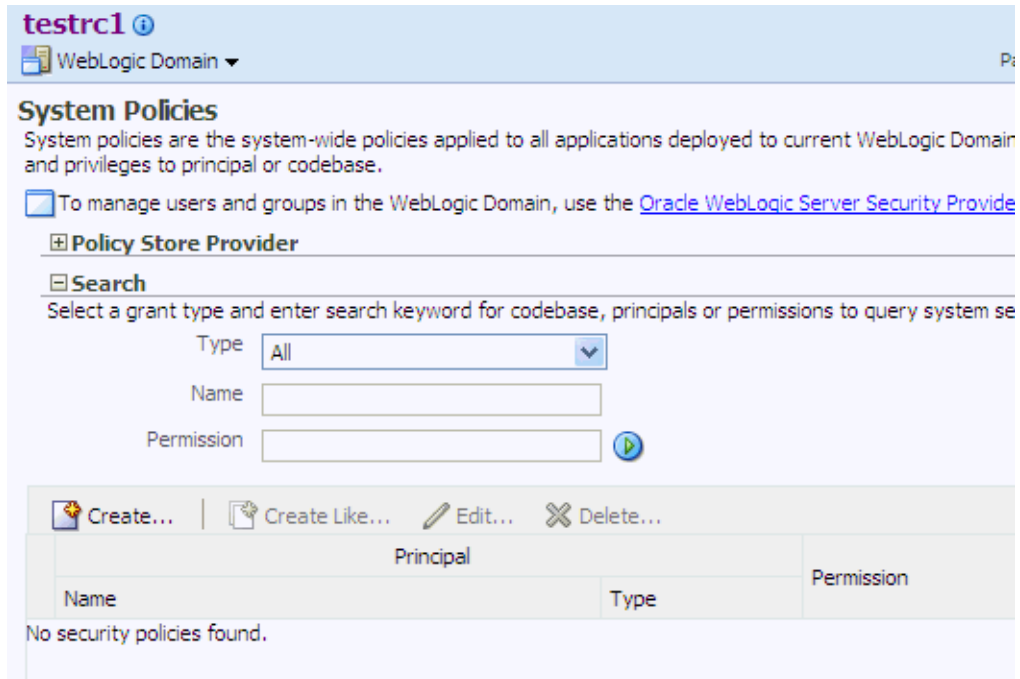
The area **Policy Store Provider** is read-only and, when expanded, displays the policy store provider currently in use in the domain.

2. To display application policies matching a given type, name, or permission, expand the **Search** area, enter the data to match, and click the blue button. The results of a query are displayed in the table at the bottom of the page.

   To redisplay the table of current application 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. In the area **Grant Details**, select type of policy to create. The valid types are Principal or Codebase. The UI differs slightly depending on the type chose. The steps below assume the selection **Principal**.

3. To add permissions to policy being created, click the button **Add** in the **Permissions** area to display the **Add Permission** dialog. In this dialog 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 query 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 table and use the button **Edit** to change the characteristics of the permission, or the button **Delete** to remove from the list.

5. To add users to the policy being created, click the button **Add User** in the **Grantee** area to display the **Add User** dialog.

   1. Use the **Search** to display user names matching a pattern. The results of the query are displayed in the box **Available Users**.

   2. Use the buttons in between the boxes to move users from the **Available Users** box to the **Selected Users** box.

   3. Click **OK** to return to the **Create System Grant** page. The users you have selected are added to the table **Grantee**.

6. To add roles to the policy being created, click the button Add Role in the Grantee area to display the **Add Role** dialog.

   1. Use the **Search** to display role names matching a type or role name. The results of the query are displayed in the box **Available Roles**.

   2. Use the buttons in between the boxes to move users from the **Available Roles** box to the **Selected Roles** box.

   3. Click **OK** to return to the **Create System Grant** page. The roles you have selected are added to the table **Grantee**.

7. Click OK to return to the System Policies page. An message at the top of the page informs you the result of the operation. If successful, the policy is added to the table at the bottom of the page.

To create a system policy based on an existing system policy:

1. Select a policy from the table.

2. Click **Create Like** to display the **Create System Grant** page. Notice that some entries in the page are filled in with data extracted from the policy selected in the previous step.

3. Follow the preceding procedure (to create a system policy) to modify those values and enter new ones, as appropriate.

8.4.2 Managing Policies with WLST Commands

If a domain administrator does not want to use Fusion Middleware Control to manage policies or wants to execute a frequent task automatically, the administrator can create a WLST script that invokes WLST security-related commands.

An online command is a command that, to perform, requires a connection to a running Oracle WebLogic Server. All commands below are **online** commands and operate on a domain policy store, regardless of whether it is file- or LDAP-based.

Read-only commands can be performed only by users in the following groups: Monitor, Operator, Configurator, or Admin. Read-write commands can be performed only by users in the following groups: Admin or Configurator. All WLST commands are available out-of-the-box with the installation of the Oracle WebLogic Server.

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

> **Important:** Before invoking a security-related WLST command in a shell, you must run the script `wlst.sh`, as illustrated in the following sample:
>
> ```
> > sh $ORACLE_HOME/common/bin/wlst.sh
> ```
>
> This ensures that the required JARs are added to the classpath. Failure to run the above script in a new shell renders the WLST commands unusable.
>
> Before running an online command, connect to the server as follows:
>
> ```
> >java weblogic.WLST
> >connect('servername', 'password', 'localhost:portnum')
> ```

OPSS supports the following **online** commands to administer policies:

- createAppRole
- deleteAppRole
- grantAppRole
- revokeAppRole
- listAppRoles
- listAppRolesMembers
- grantPermission
- revokePermission
- listPermissions
- deleteAppPolicies
- reassociateSecurityStore

All class names specified in these commands must be fully qualified path names. The argument `appStripe` refers to the application stripe (application name) and identifies the subset of domain policies pertaining a particular application.

For important information about the authenticated and the anonymous roles and WLST commands, see Section 8.4.2.12, "Granting Policies to Anonymous and Authenticated Roles with WLST Commands."

For the correct usage of the application stripe in versioned applications, see Section 8.4.2.13, "Application Stripe for Versioned Applications in WLST Commands."

8.4.2.1 createAppRole

The command `createAppRole` creates an application role in the domain 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
```

8.4.2.2 deleteAppRole

The command `deleteAppRole` removes from the domain policy store the application role with given application stripe and role name.

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
```

8.4.2.3 grantAppRole

The command `grantAppRole` adds a principal (class and name) to a role with a given application stripe and name.

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.

- `principalName` specifies the principal name.

Example of Use

The following invocation adds a principal to the role with application stripe `myApp` and name `myRole`:

```
grantAppRole.py -appStripe myApp
            -appRoleName myRole
            -principalClass com.Example.XyzPrincipal
            -principalName myPrincipal
```

8.4.2.4 revokeAppRole

The command `revokeAppRole` removes a principal (class and name) from a role with a given application stripe and name.

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 a class.

- `principalName` specifies the principal name.

Example of Use

The following invocation removes a principal from the role with application stripe `myApp` and name `myRole`:

```
revokeAppRole.py -appStripe myApp
            -appRoleName myRole
            -principalClass com.Example.XyzPrincipal
            -principalName myPrincipal
```

8.4.2.5 listAppRoles

The command `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
```

8.4.2.6 listAppRolesMembers

The command `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
```

8.4.2.7 grantPermission

The command `grantPermission` creates a permission granted to a code base 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 command 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
```

8.4.2.8 revokePermission

The command `revokePermission` removes a permission from a principal or code base 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 command 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
```

8.4.2.9 listPermissions

The command 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 command 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
```

8.4.2.10 deleteAppPolicies

The command `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 command works on just system policies.

Example of Use

```
deleteAppPolicies -appStripe myApp
```

8.4.2.11 reassociateSecurityStore

The command `reassociateSecurityStore` migrates, within a give domain, *both* the policy store and the credential store to a target LDAP server repository, and it changes the default policy and credential services to the target repository. It also allows specifying that the domain policy and credential stores be shared with those in a different domain (see optional argument `join` below).

The function of this command is equivalent to reassociating domain stores with Fusion Middleware Control.

The only kinds of server targets supported are the LDAP-based Oracle Internet Directory or Oracle Virtual Directory. The source repository can be file- or LDAP-based. This command uses and modifies the domain configuration file `jps-config.xml`, and it is supported in only the interactive mode.

Before issuing this command, ensure that you have accomplished the requisites explained in Prerequisites to Using an LDAP-Based Policy Store.

An alternate way of performing reassociation using Fusion Middleware Control is explained in Reassociating Domain Stores with Fusion Middleware Control.

If this operation is to use a one-way SSL connection to the target store, follow the procedures explained in Setting Up a One- Way SSL Connection before invoking the command.

After reassociation, to secure access to the root node of the Oracle Internet Directory store, follow the instructions in Securing Access to Oracle Internet Directory Nodes.

Interactive Mode Syntax

```
reassociateSecurityStore(domain="domainName", admin="cnSpecification",
password="passWord", ldapurl="hostAndPort", servertype="ldapSrvrType",
jpsroot="cnSpecification" [,join="trueOrfalse"])
```

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

- `domain` specifies the domain name where the reassociating takes place.
- `admin` specifies the administrator's user name on the LDAP server. The format is `cn=usrName`.

- **password** 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. The only valid types are OID (Oracle Internet Directory) or OVD (Oracle Virtual Directory).

- **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 a policy store specified in another domain. Optional. Set to true to share an existing policy store in another domain; set to false otherwise. The use of this argument allows multiple WebLogic domains to point to the same logical policy store.

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 command as follows:

```
reassociateSecurityStore(domain="otherDomain", admin="cn=adminName",
password="myPass", ldapurl="ldaps://myhost.example.com:3060", servertype="OID",
jpsroot="cn=testNode", join="true")
```

8.4.2.12 Granting Policies to Anonymous and Authenticated Roles with WLST Commands

Several WLST commands 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 commands 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 commands that required the above principal name and class specification are the following:

- `grantAppRole`

- `revokeAppRole`

- `grantPermission`

- `revokePermission`

- `listPermissions`

8.4.2.13 Application Stripe for Versioned Applications in WLST Commands

Several WLST commands 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 commands, as illustrated in the following invocation:

```
>listAppRoles myApp#v1.0
```

The list of WSLT commands that can use stripe specification are the following:

- `createAppRole`

- `deleteAppRole`

- `grantAppRole`

- `revokeAppRole`

- `listAppRoles`

- `listAppRoleMembers`

- `grantPermission`

- `revokePermission`

- `listPermissions`

- `deleteAppPolicies`

8.5 Configuring Other Artifacts with Oracle Fusion Middleware Control

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 JavaEE applications.
>
> In regards to the Keystore: a Keystore configuration should not be modified; to change any settings for the Keystore, first remove the existing configuration (uncheck the box Configure Keystore and then click OK), and then continue to specify a new one. Failure to proceed this way leads to errors and unexpected behavior.
>
> 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*.

8.5.1 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**; the page **Security Provider Configuration** appears.

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.

8.5.2 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 `<properySet>` 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**; the page **Security Provider Configuration** appears.

2. Expand, if necessary, the area **Advanced Properties**, and click **Configure** to display the page **Advanced Properties**, 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>`.

8.5.3 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

8.5.3.1 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 Chapter 10, "Configuring Single Sign-On in Oracle Fusion Middleware."

8.5.3.2 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**; the page **Security Provider Configuration** appears.

2. Expand, if necessary, the area **Single Sign-On Provider**, and click **Configure** to display the page **Single Sign-On Provider**.

3. 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 list of data changes with the provider selected).

5. Optionally, manage the provider **Custom Properties** using the buttons **Add**, **Edit**, and **Delete**, at the bottom of the page.

6. When finished, click **OK** to save the entered data.

8.5.3.3 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>
```

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:${domain.home}/servers/MyApp/-</url>
    </codesource>
  </grantee>
  <permissions>
    <permission>
      <class>oracle.security.jps.JpsPermission</class>
      <name>CredentialMapping</name>
    </permission>
  </permissions>
</grant>
```

8.6 Configuring LDAP-Based Policy Stores

The following sections provide guidelines for setting OPSS parameters and properties, and profiling LDAP-based policy stores:

- OPSS System Properties for JVM
- LDAP Policy Store Property Configuration for Maximum performance
- Profiling LDAP Policy Store APIs

8.6.1 OPSS System Properties for JVM

Table 8–1 lists some recommended special settings for OPSS system properties. These are typically specified in the script that starts Oracle WebLogic Server.

Table 8–1 OPSS System Properties for JVM

Property Name	Value	Description
domain.home	$DOMAIN _HOME	Required for code-based grants that depend on the value of this system property.
		This property *must* be set as a command-line parameter due to JVM bootstrapping restrictions.
jps.authz	ACC	Different from current out-of-the-box setting, which is DEBUG_ACC.
		Setting -Djps.auth=ACC reduces runtime and debugging overhead.
jps.policystore.hybrid.mode	false	Different from current out-of-the-box setting, which is true.
		Setting -Djps.policystore.hybrid.mode=false reduces runtime overhead.
jps.combiner.optimize	true	Different from current out-of-the-box setting, which is false.
		Setting -Djps.combiner.optimize=true improves Java2 authorization performance.
jps.combiner.optimize.lazyeval	true	Different from current out-of-the-box setting, which is false.
		Setting -Djps.combiner.optimize.lazyeval=true improves Java2 authorization performance.

8.6.2 LDAP Policy Store Property Configuration for Maximum performance

Table 8–2 lists the properties that are required to be present in any instance of an LDAP policy store service. The properties of an instance can be specified and modified with the script described in Section E.1, "Configuring OPSS Service Provider Instances with a WLST Script."

Table 8–2 LDAP Policy Store Service Instance Properties

Name	Default	Recommended	Description
oracle.security.jps.policystore.rolemember.cache.type	STATIC	STATIC	Role Member Cache Type.
oracle.security.jps.policystore.rolemember.cache.strategy	FIFO	FIFO	Role Member Cache Strategy.
oracle.security.jps.policystore.rolemember.cache.size	1000	5000	Role Member Cache Size.

Table 8–2 (Cont.) LDAP Policy Store Service Instance Properties

Name	Default	Recommended	Description
oracle.security.jps.policystore.policy.lazy.load.enable	true	true	Enable Policy Lazy Load.
oracle.security.jps.policystore.policy.cache.strategy	PERMISSION_FIFO	PERMISSION_FIFO	Permission Cache Strategy.
oracle.security.jps.policystore.policy.cache.size	1000	1000	Permission Cache Size.
oracle.security.jps.policystore.refresh.enable	true	true	Enables policy store refresh from LDAP server.
oracle.security.jps.policystore.refresh.purge.timeout	43200000	43200000	Forced policy store refresh time in milliseconds.
oracle.security.jps.ldap.policystore.refresh.interval	600000	600000	Policy store refresh polling time in milliseconds.

8.6.3 Profiling LDAP Policy Store APIs

Table 8–3 lists loggers to profile the timing of several LDAP policy store runtime APIs. These profilers are specified with Fusion Middleware Control pages, as explained in Section I.1.1.4, "Using Fusion Middleware Control Logging Support."

Table 8–3 OPSS Profilers

Category	Logger Name	Log Level	Profiles
refresh	oracle.security.jps.policystore.refresh.full.profiler	FINE	The time taken by refresh for policies.
refresh	oracle.security.jps.policystore.refresh.partial.profiler	FINE	The time taken by fine-grained various activities during policy refresh.
policy management APIs	oracle.security.jps.policystore.management.profiler	FINE	The time taken by selective policy management APIs (getAllAppRoleEntries, searchAppRoles).
policy runtime APIs	oracle.security.jps.policystore.runtime.profiler	FINE	The time taken by on-demand loading (during cache miss) of direct roles and the granted permissions for a given user.
OPSS Filter and EJB Interceptor	oracle.security.jps.policystore.runtime.profiler	FINE	The time taken to compute the application roles for the give subject by the OPSS filter and the OPSS EJB interceptor.

9

Configuring the Credential Store

A credential store is a repository of security data (credentials). A credential can hold user name and password combinations, tickets, or public key certificates. 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 credentials (user name and password) to access some external system, such as a database or an LDAP-base repository.

This chapter is divided into the following sections:

- Credential Types
- Configuring a Domain to Use an LDAP-Based Credential Store
- Reassociating the Domain Credential Store
- Migrating Credentials to the Domain Credential Store
- Managing the Domain Credential Store

9.1 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 combination of map name and key name must be unique for all entries in the credential store.

Oracle Wallet is the default credential store; in a production environment, it is recommended the use of an LDAP-based Oracle Internet Directory as the credential store. It is also recommended that the Oracle Wallet be used to store X.509 certificates. The credential store does not allow the storage of end-user digital certificates.

9.2 Configuring a Domain to Use an LDAP-Based Credential Store

An LDAP-based credential store is typically used and recommended in production environments. The LDAP servers supported in this release are the Oracle Internet Directory and the Oracle Virtual Directory (LSA).

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

The prerequisites for using an LDAP-based credential store are identical to those for using an LDAP-based policy store. For details, see Section 8.1.2, "Prerequisites to Using an LDAP-Based Policy Store."

An LDAP-based credential store is recommended in distributed environments. For details, see Section 8.1.1, "Multiple-Node Server Environments."

9.3 Reassociating the Domain Credential Store

The reassociation of both the Credential and the Policy stores is accomplished as a unit using Fusion Middleware Control or the WLST command `reassociateSecurityStore`, as explained in Section 8.2, "Reassociating the Domain Policy Store."

9.4 Migrating Credentials to the Domain Credential Store

A domain includes one credential store. Application-specific credentials are supported and migrated to credentials in the domain credential store when the application is deployed. Thus, all servers and all applications deployed in a domain use a common credential store, the domain credential store. Access to the credential store requires permission.

The following sections explain how to migrate credentials to the domain credential store when an application is deployed with either Fusion Middleware Control or a WLST command:

- Migrating Application Credentials with Fusion Middleware Control
- Migrating Credentials with the Command migrateSecurityStore

9.4.1 Migrating Application Credentials with Fusion Middleware Control

Application credentials are specified in the application file `cwallet.sso` and can be migrated to the domain credential store when the application is deployed or redeployed to a managed server in the WebLogic environment.

The migration of credentials differs with the migration of policies in several aspects: (a) credentials cannot be removed automatically when the application is undeployed; (b) the migration of credentials at deployment or redeployment with overwriting is allowed only when the WebLogic domain is operating in development mode.

For details, see procedure in Section 7.5.2, "Migrating Policies and Credentials at Deployment."

9.4.2 Migrating Credentials with the Command migrateSecurityStore

A specific credential map or all credentials in all maps can be migrated manually from a source repository to a target repository using the WLST command `migrateSecurityStore`.

This command 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.

The kinds of the repositories where the source and target data is stored is transparent to the command, and any combination of file-based and LDAP-based repositories is allowed (LDAP-repositories must use an Oracle Virtual Directory or an Oracle Internet Directory LDAP server only).

This command is typically used when transitioning from a test to a production environment, or when reproducing policy or credential data from one domain to another domain.

Note that the target LDAP server requires setting up before the command is used, as described in Section 8.1.2, "Prerequisites to Using an LDAP-Based Policy Store."

> **Note:** Since the command `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 7.5.2.3, "Migrating Large Volume Policy and Credential Stores.".

The commands listed below can be run in interactive mode or in script mode. In interactive mode, you enter the command at a command-line prompt and view the response immediately after. In script mode, you write commands 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 further details about WLST commands and their syntax, see *Oracle Fusion Middleware WebLogic Scripting Tool Command Reference*.

> **Important:** Before invoking a security-related WLST command in a shell, you must run the script `wlst.sh`, as illustrated in the following sample:
>
> ```
> > sh $ORACLE_HOME/common/bin/wlst.sh
> ```
>
> This ensures that the required JARs are added to the classpath. Failure to run the above script in a new shell renders the WLST commands unusable.
>
> Before running an online command, connect to the server as follows:
>
> ```
> >java weblogic.WLST
> >connect('servername', 'password', 'localhost:portnum')
> ```

Script and Interactive Modes Syntaxes

The command syntax varies depending on the scope of the credentials being migrated (credentials in a particular map or all credentials). Optional arguments are enclosed by square brackets.

To migrate *all* credentials use either of the script (first) or interactive (second) syntaxes (arguments are written in separate lines for clarity):

```
migrateSecurityStore.py -type credStore
```

```
                              -configFile jpsConfigFileLocation
                              -src srcJpsContext
                              -dst dstJpsContext
```

```
migrateSecurityStore(type="credStore", 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 command is run. Typically, this configuration
  file is created just to be used with the command and serves no other purpose. This
  files 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. See second example below. For complete details, see Section 15.4.7,
  "Specifying Bootstrap Credentials Manually."

- `src` specifies the name of a jps-context in the configuration file passed to the
  argument `configFile`. This context identifies the source credential repository.

- `dst` specifies the name of another jps-context in the configuration file passed to
  the argument `configFile`. This context identifies the target credential repository.

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 command determines the
locations of the source and the target repositories involved in the migration.

To migrate *just* one credential map, use either of the script (first) or interactive (second)
syntaxes (arguments are written in separate lines for clarity):

```
migrateSecurityStore.py -type folderCred
                        -configFile jpsConfigFileLocation
                        -src srcJpsContext
                        -dst dstJpsContext
                        [-srcFolder map1]
                        [-dstFolder map2]
                        [-srcConfigFile alternConfigFileLocation]
                        [-overWrite trueOrFalse]
```

```
migrateSecurityStore(type="folderCred", configFile="jpsConfigFileLocation",
src="srcJpsContext", dst="dstJpsContext", [srcFolder="map1"],
[dstFolder="map2"], [srcConfigFile="alternConfigFileLocation"],
[overWrite="trueOrFalse"])
```

The meanings of the arguments `configFile`, `src`, and `dst` are identical to the
previous case. The meanings of the last four arguments (all optional) are as follows:

- `srcFolder` specifies the name of the map containing the credentials to be
  migrated. This argument is optional. If unspecified, the credential store is assumed
  to have only one map and the value of this argument defaults to the name of that
  map.

- `dstFolder` specifies the map to where the source credentials are migrated. This
  argument is optional and, if unspecified, defaults to the map passed to
  `srcFolder`.

- `srcConfigFile` specifies the location of an alternate configuration file, and it is
  used in the special case in which credentials are not configured in the file passed to

configFile. This argument is optional. If unspecified, it defaults to the value passed to configFile; if specified, the value passed to configFile is ignored.

- overWrite specifies whether a target credential matching a source credential should be overwritten by or merged with the source credential. Set to true to overwrite the target credential; set to false to merge matching credentials. Optional. If not specified, defaults to false. When set to false, if a matching is detected, the source credential is disregarded and a warning is logged.

If the input does not follow the syntax above, the command 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.

For examples of use, see Section 7.5.2.2, "Migrating Credentials Manually."

9.5 Managing the Domain 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 FMC pages or WLST commands, and how code can access data in the CSF.

- Managing Credentials with Fusion Middleware Control
- Managing Credentials with WLST Commands

9.5.1 Managing Credentials with Fusion Middleware Control

Fusion Middleware Control allows the management of credentials in a WebLogic domain, regardless of the type of credential store provider used in the domain, as explained in Managing Credentials.

This task is performed, for example, when a credential packaged with the application is to be replaced with another credential valid in the target environment.

9.5.1.1 Managing Credentials

This section explains the steps you follow to manage credentials in a domain credential store with Fusion Middleware Control, such as creating, viewing, deleting, or updating a credential stored in the credential store.

1. Log in to Oracle Enterprise Manager and navigate to *Domain* > **Security** > **Credentials**, to display the **Credentials** page. This page is partially shown in the following graphic:
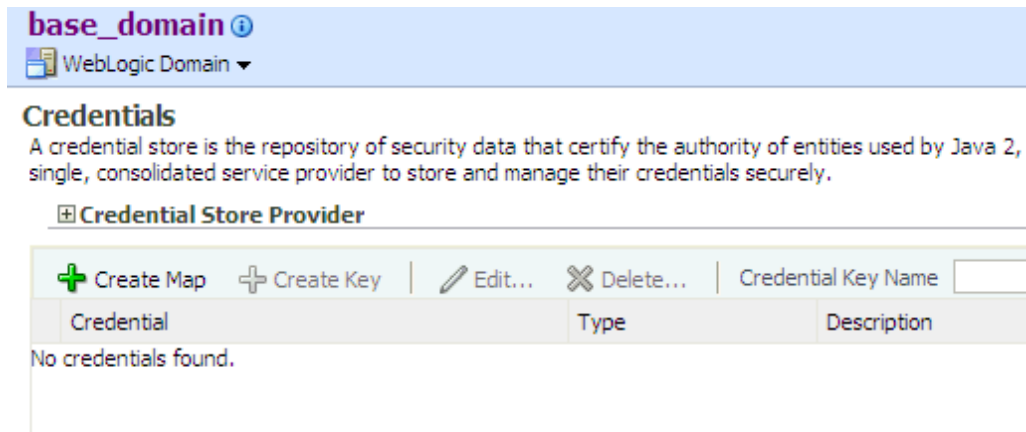
The area **Credential Store Provider** is read-only and, when expanded, displays the credential store provider currently in use in the domain.

The table below this read-only area allows creating, editing, and searching credentials.

2. At any point, use the button **Delete** to remove a selected item (key or map) in the table. Note that deleting a credential map, deletes all keys in it. Similarly, use the button **Edit** to view or modify the data in a selected item.

3. To display credentials matching a given key name, enter the string to match in the box **Credential Key Name**, and then click the blue button to the right of it. The result of the query is displayed in the table.

4. To redisplay the list of credentials after examining the results of a query, select *Domain* > **Security** > **Credentials**.

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** where the key is inserted, enter a key in the text box **Key**, select a type from the menu **Type** (the appearance of the dialog changes according to the type selected), enter the required data.

3. Click **OK** when finished to return to the **Credentials** page. The new key is shown under the map icon corresponding to the map you selected.

9.5.2 Managing Credentials with WLST Commands

If a domain administrator does not want to use Fusion Middleware Control to manage credentials or wants to execute a frequent task automatically, the administrator can create a WLST script that invokes WLST security-related commands.

An online command is a command that to operate requires the Oracle WebLogic Server to be running. All commands below operate on a domain credential store, regardless of whether it is file-based or LDAP-base, to which you connect just before the commands are available.

Read-only commands can be performed only by users in the following roles: Monitor, Operator, Configurator, or Admin. Read-write commands can be performed only by users in the following roles: Admin or Configurator. All WLST commands are available out-of-the-box with the installation of the Oracle WebLogic Server.

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

> **Important:** Before invoking a security-related WLST command in a shell, you must run the script `wlst.sh`, as illustrated in the following sample:
>
> ```
> > sh $ORACLE_HOME/common/bin/wlst.sh
> ```
>
> This ensures that the required JARs are added to the classpath. Failure to run the above script in a new shell renders the WLST commands unusable.
>
> Before running an online command, connect to the server as follows:
>
> ```
> >java weblogic.WLST
> >connect('servername', 'password', 'localhost:portnum')
> ```

WLST supports the following commands to administer credentials, all **online** commands, except for offline `modifyBootStrapCredential`:

- listCred
- updateCred
- createCred
- deleteCred
- modifyBootStrapCredential

9.5.2.1 listCred

The command `listCred` returns the list of attribute values of a credential in the domain credential store with given map name and key name. This command 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
```

9.5.2.2 updateCred

The command `updateCred` modifies the type, user name, and password of a credential in the domain credential store with given map name and key name. This command 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")
```

9.5.2.3 createCred

The command `createCred` creates a credential in the domain credential store with a given map name, key name, user name and password. This command 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")
```

9.5.2.4 deleteCred

The command `deleteCred` removes a credential with given map name and key name from the domain 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
```

9.5.2.5 modifyBootStrapCredential

The offline command `modifyBootStrapCredential` modifies the bootstrap credentials configured in the default jps context, and it is typically used in the following scenario: suppose that the domain policy and credential stores are LDAP-based, and the credentials to access the LDAP store (stored in the LDAP server) are changed. Then this command can be used to seed those changes into the bootstrap credential store.

This command must be run by a domain administrator and 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 command 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

Configuring Single Sign-On in Oracle Fusion Middleware

The chapter outlines a set of recommended single sign-on solutions for Oracle Fusion Middleware. It also provides some general guidelines and common use cases for configuring authentication and single sign-on in the Oracle Fusion Middleware environment. This chapter includes the following major sections:

- Choosing the Right SSO Solution for Your Deployment
- Deploying the Oracle Access Manager Solutions
- Deploying the OracleAS Single Sign-On (OSSO) Solution
- Synchronizing the User and SSO Sessions: SSO Synchronization Filter
- Setting Up Debugging in the WebLogic Administration Console

10.1 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 two 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

Security-enabled components, such as Oracle Service Oriented Architecture (SOA) and Oracle WebCenter, can choose either solution (Oracle Access Manager 10g or OracleAS Single Sign-On 10g). Customers using a mixed product stack 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.

> **See Also:** *Oracle Fusion Middleware Security Overview*

- **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 recommends Oracle Access Manager because it 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 number of OS platforms to provide a flexible solution.

  Oracle Access Manager is the recommended choice whether you are new to Oracle Fusion Middleware, or you are considering an enterprise-level SSO solution, or you already have Oracle Access Manager in your environment.

  > **See Also:** "Deploying the Oracle Access Manager Solutions" on page 10-3

- **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 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.

  > **See Also:** *Oracle Application Server Single Sign-On Administrator's Guide 10g (10.1.4.0.1)* on Oracle Technology Network at: http://www.oracle.com/technology/documentation/oim10 14.html

- **Portal, Forms, Reports, and Discoverer 11g** : In 10g these components relied on Oracle Single Sign-On as the SSO solution, and they continue to do so in 11g. OSSO 10g, along with Oracle Delegated Administration Services 10g, are mandatory for these components. Oracle Internet Directory 10g or 11g are also mandatory for these components.

  > **See Also:** The following topic in this chapter and other 11g Release 1 (11.1.1) manuals:
  > - "Deploying the OracleAS Single Sign-On (OSSO) Solution" on page 10-70
  > - *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 Access Manager is the recommended enterprise-wide solution. If applications mandating OracleAS Single Sign-On (Oracle Portal for example) are deployed, it is possible to delegate the authentication from OracleAS Single Sign-On (OSSO) to Oracle Access Manager. When integrating Oracle Access Manager's authorization functionality, either Oracle Access Manager or OSSO can act as the authentication engine. Such an integration also requires Oracle Internet Directory for the applications that require integrating Oracle Access Manager and OSSO

  See Also: "Integrating with Oracle Application Servers" in the 10*g* (10.1.4.3) *Oracle Access Manager Integration Guide*.

- **Windows Native Authentication for Microsoft Clients**: 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: "Configuring Single Sign-On with Microsoft Clients" in *Oracle Fusion Middleware Securing Oracle WebLogic Server*

10.2 Deploying the Oracle Access Manager Solutions

Oracle Access Manager is part of Oracle's enterprise class suite of products for identity management and security. Oracle Access Manager provides a wide range of identity administration and security functions, including a single sign-on solution.

The Oracle Access Manager Authentication provider is a new component that works with Oracle WebLogic Server. An application can use either or both of the Oracle Access Manager Authentication provider features, each of which enables a specific Oracle Access Manager function for WebLogic users:

- **Identity Asserter for Single Sign-on**

  This function uses Oracle Access Manager authentication services and also validates already-authenticated Oracle Access Manager users through the ObSSOCookie and creates a WebLogic-authenticated session. It also provides single sign-on between WebGates and portals.

  Note: Oracle Web Services Manager uses the Identity Asserter for single sign-on.

- **Authenticator**

  This function uses Oracle Access Manager authentication 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.

The information in this section presumes that you are familiar with Oracle WebLogic Server and with Oracle Access Manager. For general details about configuring and managing Oracle Access Manager, see the *Oracle Access Manager Identity and Common Administration Guide* and the *Oracle Access Manager System Administration Guide*.

The rest of this section is organized as follows:

- Scenarios for Applications with the Oracle Access Manager Authentication Provider explains how an application can use Oracle Access Manager Authentication provider features according to the current application setup.

- **Uses of the Authentication Provider for Oracle Access Manager** describes the two provider features: authentication and single sign-on.

- **Configuring Oracle Access Manager Identity Assertion for Single Sign-On** provides the procedures that you must perform to set up the provider for single sign-on.

- **Configuring the Oracle Access Manager Authenticator** provides the procedures that you must perform to set up the provider for authentication.

- **Configuring Identity Assertion for Oracle Web Services Manager** explains the procedures that you must perform to set up this scenario.

- **Configuring Global Logout for Oracle Access Manager** on page 10-57

- **Troubleshooting Tips for Provider Deployment** describes how to detect and correct issues with your implementation of the Oracle Access Manager Authentication provider.

> **See Also:**
>
> - "Synchronizing the User and SSO Sessions: SSO Synchronization Filter" on page 10-93
> - "Setting Up Debugging in the WebLogic Administration Console" on page 10-95

10.2.1 Scenarios for Applications with the Oracle Access Manager Authentication Provider

This section explains how an application can use Oracle Access Manager and the companion Authentication provider according to the current application setup:

- **Applications Using Oracle Access Manager for the First TIme**

- **Applications Migrating from Oracle Application Server to Oracle WebLogic Server**

- **Applications Using Oracle Access Manager Security Provider for WebLogic SSPI**

10.2.1.1 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**: Review "About Using Oracle Access Manager Identity Asserter for Single Sign-on" on page 10-7.

> **Note:** The Identity Asserter is also used when you have Oracle Web Services Manager protecting Web services.

- **Authenticator**: Review "About Using the Oracle Access Manager Authenticator" on page 10-9.

10.2.1.2 Applications Migrating from Oracle Application Server to Oracle WebLogic Server

If your application has been deployed on the old Oracle Application Server (OC4J), to have the application use the Oracle Access Manager Authentication provider feature with Oracle WebLogic Server, proceed as follows:

- Remove all OC4J-specific settings from the application configuration.

■ **Identity Asserter for Single Sign-On**: Review "About Using Oracle Access Manager Identity Asserter for Single Sign-on" on page 10-7.

> **Note:** The Identity Asserter is also used when you have Oracle Web Services Manager protecting Web services.

■ **Authenticator**: Review "About Using the Oracle Access Manager Authenticator" on page 10-9.

> **See Also:** *Oracle Fusion Middleware Upgrade Guide for Java EE* for information about upgrading your Java EE applications from Oracle Containers for Java EE (OC4J) 10g Release 3 (10.1.3) to Oracle WebLogic Server

10.2.1.3 Applications Using Oracle Access Manager Security Provider for WebLogic SSPI

The 10*g* (10.1.4.2.0) or earlier Oracle Access Manager Security Provider for WebLogic SSPI provides authentication, authorization, and single sign-on across J2EE applications that are deployed in the WebLogic platform. The Security Provider for WebLogic SSPI enables WebLogic administrators to use Oracle Access Manager 10*g* (10.1.4.2.0) to control user access to business applications.

> **Note:** Security Provider for WebLogic SSPI is also known as "Security Provider" in the *Oracle Access Manager Integration Guide* for release 10*g* (10.1.4.2.0).

The 10*g* (10.1.4.2.0) or earlier 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 10*g* (10.1.4.3) Oracle Access Manager Authentication provider is more easily installed and configured than the 10*g* (10.1.4.2.0) Security Provider for WebLogic SSPI. The 10*g* (10.1.4.3) Authentication provider works with all platforms supported by Oracle WebLogic Server. However, the 10*g* (10.1.4.3) Authentication provider offers just two services: authentication and single sign-on (SSO).

If your application has been using the 10*g* (10.1.4.2.0) or earlier Oracle Access Manager Security Provider for WebLogic SSPI for only authentication and SSO, the deployment is a good candidate for the latest Authentication provider when this implementation is certified. Oracle continues to certify the 10*g* (10.1.4.3) Oracle Access Manager Authentication provider with earlier WebLogic Server releases. Be sure to check the latest certification matrix for more information. However, if your application relies on features other than those offered by the latest Oracle Access Manager Authentication provider, you should continue to use the Oracle Access Manager Security Provider for WebLogic SSPI.

> **See Also:** "Uses of the Authentication Provider for Oracle Access Manager"

10.2.2 Uses of the Authentication Provider for Oracle Access Manager

Applications can use the Authentication provider for Oracle Access Manager for authentication or single sign-on (or both). This section outlines the common components required by these functions as well as how these work:

- Required Components and Files

- About Using Oracle Access Manager Identity Asserter for Single Sign-on

- About Using the Oracle Access Manager Authenticator

10.2.2.1 Required Components and Files

Regardless of the functionality you intend to use (Authentication or Identity Assertion for single sign-on), a number of components and files are required. The following list provides a brief overview of the required components and files:

> **See Also:**   "Installing and Setting Up Required Components for Oracle Access Manager Providers" on page 10-10

- An enterprise directory server (Oracle Internet Directory or Sun One directory server is used by the Oracle Access Manager Access Server 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) includes the Oracle Access Manager Authentication provider and OAMCfgTool JAR files.

- Authentication provider for Oracle Access Manager jar files are available when you install an Oracle Fusion Middleware product (Oracle Identity Management, Oracle SOA Suite, or Oracle WebCenter).

  > **Note:**   If you have a stand-alone Oracle WebLogic Server with no Oracle Fusion Middleware application, you obtain the JAR files as described in 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.

  - **oamcfgtool.jar**: Provides the platform-agnostic OAMCfgTool and scripts that 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.

- OHS 11g must be configured as a reverse proxy for the 10*g* (10.1.4.3) WebGate required by the Oracle Access Manager Identity Asserter

- Oracle Access Manager 10*g* (10.1.4.3) components:

  - Identity Server

  - WebPass

  - Policy Manager

- – Access Server

- – WebGate (use for Identity Assertion for single sign-on only)

  WebGate is Web server plug-in access client that intercepts HTTP requests for Web resources and forwards them to the Access Server for authentication and authorization.

- – AccessGate (use with the Authenticator or when you have Oracle Web Services Manager protecting Web services)

  The custom AccessGate is available in oamAuthnProvider.jar. No Web server is required for this component.

  > **See Also:** "Installing and Setting Up Required Components for Oracle Access Manager Providers" on page 10-10

10.2.2.2 About Using Oracle Access Manager Identity Asserter for Single Sign-on

This topic describes and illustrates the use of the Oracle Access Manager Identity Asserter for single sign-on.

The Authentication provider for Oracle Access Manager can be configured as the Identity Asserter for single sign-on. In this case, the provider protects Web resources only.

This Identity Asserter for single sign-on uses perimeter authentication performed by WebGate on the Web Tier and the ObSSOCookie to assert the identity of users who try to access protected WebLogic resources.

All requests are routed to a reverse proxy Web Server; requests are intercepted by WebGate. The user is challenged for credentials based on the authentication scheme that is configured within Oracle Access Manager. The recommended scheme is Form (form-based login).

If authentication succeeds, WebGate generates an ObSSOCookie, the Web server mod_ weblogic module forwards the request to Oracle WebLogic Server, which, in turn, invokes Oracle Access Manager Identity Asserter for single sign-on (with the request and the cookie) for validation.

> **Note:** The user is challenged for credentials based on the authentication scheme that is configured within Oracle Access Manager. You can set up this scheme using OAMCfgTool, as described later.
>
> In text, 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 Security Service invokes Oracle Access Manager Identity Asserter for single sign-on, which gets the ObSSOCookie from the incoming request, and populates the subject with WLSUserImpl principal. The Identity Asserter for single sign-on also adds the WLSGroupImpl principal that corresponds to the user's groups, if any. Oracle Access Manager validates the cookie.

Figure 10–1 illustrates the distribution of components and the flow of information when Oracle Access Manager Identity Asserter for single sign-on is configured with Oracle Fusion Middleware. A detailed description follows the figure.

Figure 10–1   Oracle Access Manager Single Sign-On Solution for Web Resources Only
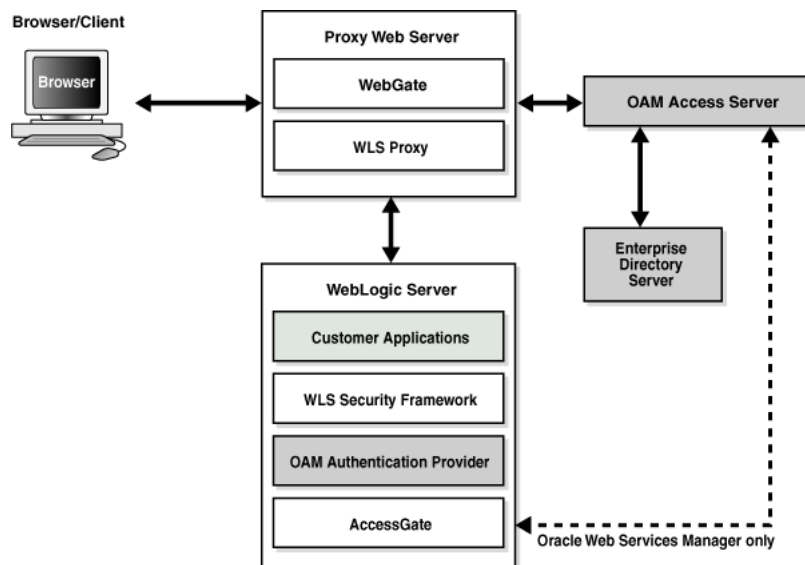
The following process overview describes the processing that occurs between components when the Identity Asserter for single sign-on is used with Web-only applications. This implementation handles nearly all SSO use cases. The exception is when you have Oracle Web Services Manager protected Web services. In this case, there is no trusted WebGate. Instead the AccessGate provided with the Identity Asserter is contacted and interacts with the Access Server; all other processing is essentially the same.

> **Note:**   The Identity Asserter for single sign-on processing is the same whether the implementation relies on a trusted WebGate or the custom AccessGate provided in oamAuthnProvider.jar.

Process overview: Oracle Access Manager Identity Asserter with 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 Oracle Access Manager Access Server to check if 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 Access Server.

5. Access Server validates the user credentials against those stored in user directory and returns the response back to WebGate. Processing continues based on:

    ■   **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. Access Server generates the session token and sends it to the WebGate. WebGate sets the ObSSOCookie and value as that returned from Access Server. 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.

7. WebLogic Server security service invokes the Oracle Access Manager Identity Asserter for single sign-on which is configured to accept the tokens of type "ObSSOCookie". The Identity Asserter initializes a CallbackHandler with the ObSSOCookie. 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.

> **See Also:** Set up details for the implementation you need in:
>
> - "Configuring Oracle Access Manager Identity Assertion for Single Sign-On" on page 10-16
> - "Configuring Identity Assertion for Oracle Web Services Manager" on page 10-50

10.2.2.3 About Using the Oracle Access Manager Authenticator

This topic describes and illustrates use of the Oracle Access Manager Authenticator.

Oracle Access Manager can be configured to protect access to Web and non-Web resources and to use the Authentication provider as the Authenticator.

> **Note:** The Authenticator function does not provide single sign-on.

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.

> **Note:** As the authenticator, the Authentication provider 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.

Figure 10–2 illustrates the distribution of components and flow of information for Oracle Access Manager authentication for Web and non-Web resources. Details follow the figure. Standard Oracle Access Manager Identity System (Identity Server and WebPass), Policy Manager, and WebGate components are present but not illustrated because the Authenticator communicates with the Access Server through a custom AccessGate.

Figure 10–2   Oracle Access Manager Authentication for Web and non-Web Resources
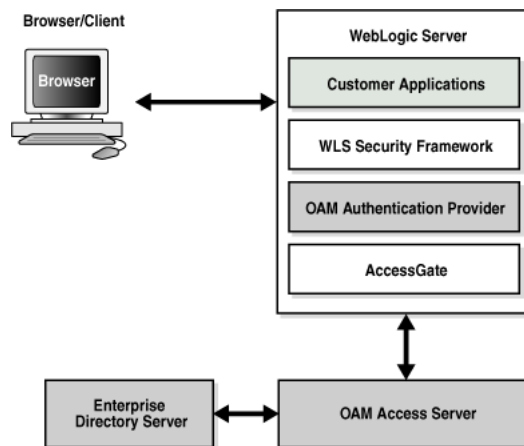
Process overview: Oracle Access Manager Authenticator for Web and non-Web Resources

1. A user attempts to access a J2EE 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 NAP library to communicate with the 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 Oracle Access Manager Authenticator" on page 10-38 for set up details

10.2.3 Installing and Setting Up Required Components for Oracle Access Manager Providers

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 Installation and Setup

- Installing Components and Files

■ Creating Resource Types in Oracle Access Manager

10.2.3.1 About Oracle Access Manager Installation and Setup

This topic provides a brief installation and setup overview if you are new to Oracle Access Manager.

Identity System: The Identity System consists of at least one Identity Server and one WebPass. After installing at least one Identity Server and one WebPass, you must set up the Identity System. During Identity System setup, you specify a Master Administrator (the highest level administrator). A Master Administrator can specify other Master and Delegated Administrators for Identity and Access Systems.

Policy Manager: After Identity System setup, you can install and set up a Policy Manager. During Policy Manager setup, ensure that the default policy protecting the Policy Manager, /access, is created and enabled.

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 WebGate/AccessGate Profiles and Policy Domains

This topic introduces the WebGate/AccessGate profiles, policy domains, and the methods you can use the 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 10-12
>
> - "About Using OAMCfgTool" on page 10-20
>
> - "Configuring WebGates and Access Servers" in the *Oracle Access Manager System 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:**

■ **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"
>
> ■ "Configuring WebGates and Access Servers" in the *Oracle Access Manager System Administration Guide*

10.2.3.2 Installing Components and Files

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 Authentication Provider

1. An Oracle Internet Directory or 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), and 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
```

> **Note:** With a stand-alone Oracle WebLogic Server with no Fusion Middleware application, you obtain and store the JAR files as described when you perform tasks later in this chapter.

4. Install OHS 11g for the Oracle Access Manager 10*g* (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 10*g* (10.1.4.3) components and perform initial setup as follows:

   > **See Also:** "About Oracle Access Manager Installation and Setup" on page 10-11

   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:** "About Using OAMCfgTool" on page 10-20

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 WebGate/AccessGate Profiles and Policy Domains" on page 10-11

- 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:

- **Create Resource Types**: "Creating Resource Types in Oracle Access Manager" on page 10-15 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 Oracle Access Manager Identity Assertion for Single Sign-On" on page 10-16.

- **Authenticator**: Perform tasks in "Configuring the Oracle Access Manager Authenticator" on page 10-38.

- **Oracle Web Services Manager**: Perform tasks in "Configuring Identity Assertion for Oracle Web Services Manager" on page 10-50.

- "Troubleshooting Tips for Provider Deployment" on page 10-61.

10.2.3.3 Creating Resource Types in Oracle Access Manager

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. 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

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 Oracle Access Manager Authenticator" on page 10-38

   - **Oracle Web Services Manager**: "Configuring Identity Assertion for Oracle Web Services Manager" on page 10-38

10.2.4 Configuring Oracle Access Manager Identity Assertion for Single Sign-On

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 Required Components for Oracle Access Manager Providers" on page 10-10 should be performed, including:

- Installing Components and Files

Note: You can add WebGate profiles 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 Oracle Access Manager Identity Asserter

6. Testing Oracle Access Manager Identity Assertion for Single Sign-on

7. Configuring Global Logout for Oracle Access Manager

10.2.4.1 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:

- Setting Up the Application Authentication Method for Identity Asserter for Single Sign-On

- Confirming mod_weblogic for Oracle Access Manager Identity Asserter

- Establishing Trust between Oracle WebLogic Server and Other Entities

10.2.4.1.1 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.

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 Oracle Access Manager Identity Asserter

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".

10.2.4.1.2 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 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".

10.2.4.1.3 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. 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.

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 10–12 provides a description of each parameter in a connection filter.

Table 10–1   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".

10.2.4.2 Configuring the Authentication Scheme for the Identity Asserter

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.

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.

For more information, see:

- About Using OAMCfgTool
- OAMCfgTool Process Overview
- Using OAMCfgTool to Create an Authentication Scheme, Policy Domain, and a WebGate Profile for Identity Assertion

10.2.4.2.1 About Using OAMCfgTool This topic introduces OAMCfgTool, which can be used only if you are deploying the Oracle Access Manager 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. 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
  ```

Unless you specify an LDIF output file, configuration changes are written directly in the LDAP directory server that is configured with Oracle Access Manager. 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. Without an LDIF file, Oracle Access Manager cache flush is triggered so that changes are implemented immediately.

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.

Table 10–2 provides both required and optional OAMCfgTool parameters and values for CREATE mode. You can specify multiple parameters at one time. See also, "Known Issues: JAR Files and OAMCfgTool" on page 10-61.

> **Note:**
>
> ■ By default, OAMCfgTool will prompt for required passwords on the command line in a secure way if you do not enter them. For example:
>
> ```
> Enter app_agent_password:
> Enter ldap_userpassword:
> Enter oam_aaa_passphrase:
> Processed input parameters
> ```
>
> ■ A -noprompt option is available to avoid the password prompt

Table 10–2   OAMCfgTool CREATE Mode Parameters and Values

Parameters	CREATE Mode Values
Required Parameters	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.
app_agent_password	Password to be provisioned for the WebGate. In the AccessGate Profile within the 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.
ldap_host	DNS name of the computer hosting the LDAP directory server for Oracle Access Manager **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.
ldap_userpassword	Password of LDAP administrative user. Passwords appear in clear text but are not captured in a log file.
oam_aaa_host	DNS name of the computer hosting an accessible Access Server. OAMCfgTool does not recognize primary and secondary Access Server designations. Set up the WebGate profile with a single accessible Access Server. Oracle Access Manager can later associate the profile with designated Access Servers.
oam_aaa_port	Listening port on the accessible Access Server
Optional	**Values**
-help	Provides a list of parameters and descriptions.

Table 10–2   (Cont.) OAMCfgTool CREATE Mode Parameters and Values

Parameters	CREATE Mode Values
web_domain	**Fresh Web Tier**: Omit web_domain to create a new WebGate profile. The profile is populated with a WebGate name, Host name, and Preferred HTTP host all using the same app_domain value as follows: ▪ app_domain=*ABC* (without web_domain) ▪ WebGate name: *ABC*_AG (here _AG is appended to the app_domain) ▪ Host: *ABC* ▪ Preferred HTTP Host: *ABC* **Existing Web Tier**: Include web_domain to specify the name of an existing host identifier in Oracle Access Manager to tie new policies to an existing host ID. For example: web_domain=*existing_host_Identifier* When WebGate intercepts a request, it checks the request for an address. If the address is on the host identifiers list, this address is mapped to the official host name, and the Access System can apply the rules and policies that protect the resource. If virtual Web hosting is supported, you supply a reserved name in the Preferred HTTP Host field instead of a host name variation. For more information, see *Oracle Access Manager System Administration Guide*.
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 identity public URIs by providing a comma separated list: "uri1,uri2,uri3", for example. The default URL pattern that is part of the URIs is created: /.../public/.../. For example "uri1,uri2,uri3" creates 3 public URLs which include "/.../public/.../" as a default URL pattern. Users can log in to Policy Manager and change the URL patterns, if needed.
ldap_base	Base from which all LDAP searches are performed. **Note**: If Oracle Access Manager user data and configuration data are stored in different directory servers, the following information is required for each: ▪ ldap_host= ▪ ldap_port= ▪ ldap_userdn= ▪ ldap_userpassword= ▪ ldap_base=
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.
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).
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.

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 10–3 provides both required and optional OAMCfgTool parameters and values for VALIDATE mode.

Table 10–3   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.
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
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.
	Note: If Oracle Access Manager user data and configuration data are stored in different directory servers, the following information is required for each:
	■ ldap_host=
	■ ldap_port=
	■ ldap_userdn=
	■ ldap_userpassword=
	■ ldap_base=
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).

10.2.4.2.2 OAMCfgTool Process Overview This topic describes the processing that occurs when you use OAMCfgTool with various parameters and values for your environment.

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. If no existing access policy uses this host identifier, a new policy is set up.

> **Note:** In a fresh Web Tier the web_domain parameter should be omitted in CREATE mode to create new WebGate profile. In this case, as described in Table 10–2:
>
> - app_domain specifies the WebGate name, host name, and Preferred HTTP Host
> - app_agent_password specifies the AccessGate password
> - cookie_domain specifies the Primary HTTP Cookie Domain
> - oam_aaa parameters associate the profile with the Access Server

3. The protected_uris parameter defines application-specific URL's to protect HTTP resources using the host identifier (or the new WebGate/AccessGate profile created in Step 2).

4. The public_uris parameter creates a Public_URI_Policy to unprotect certain URIs for http resources (GET and POST operations) in app_domain *name*.

5. The LDAP parameters specify the directory server used by Oracle Access Manager to identity the searchbase from which all LDAP queries are performed. For more information, see Table 10–2

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

10.2.4.2.3 Sample Policy Domain and AccessGate Profile Created with OAMCfgTool This topic describes and illustrates the results of running OAMCfgTool 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 OAMCfgTool

Policy Domain, General Tab

[Figure 10–3](#) illustrates the General tab in a sample policy domain created with OAMCfgTool. The Description is provided automatically.

Name: *app_domain* value specified with OAMCfgTool
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 10–3 Sample OAMCfgTool Policy Domain General Tab
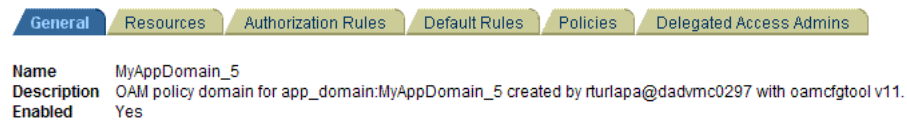
Policy Domain, Resources Tab

[Figure 10–4](#) 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 10–4 Sample OAMCfgTool Policy Domain Resources Tab
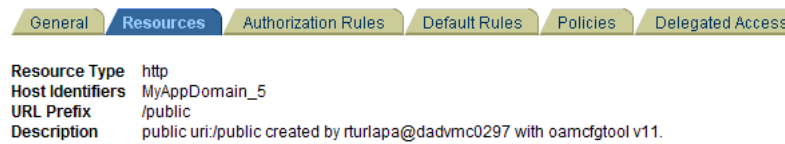
Policy Domain, Authorization Rules Tab

[Figure 10–5](#) 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 10–5   Sample OAMCfgTool Policy Domain Authorization Rules Tab

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 10–6 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 10–6.
> **Actions**: There are no actions defined.

Figure 10–6   Sample OAMCfgTool Policy Domain Default Rules Tab
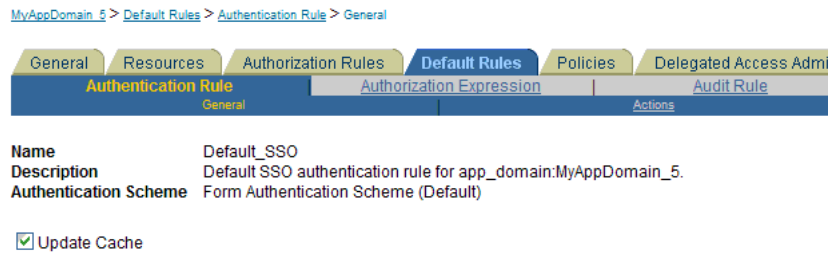
Authorization Expression
Authorization Expression: Default_Authorization
> **Duplicate Actions**: No policy defined for this Authorization Expression. The Access System level default policy for dealing with duplicate action headers are employed.

> **Actions**
> Authorization Success

Return	Type	Name	Attribute
	HeaderVar	REMOTE_USER	uid
	HeaderVar	OAM_REMOTE_USER	uid

Policy Domain, Policies Tab

Figure 10–7 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 10–7   Sample OAMCfgTool Policy Domain Policies Tab

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 10–8 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 10–8   OAMCfgTool Policy Domain Delegated Access Admins Tab

>   **See Also:**   "Protecting Resources with Policy Domains" in the *Oracle Access Manager System 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 10–9 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 10–9   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

AccessGate Profile

Figure 10–10 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 10–10   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

10.2.4.2.4   Using OAMCfgTool to Create an Authentication Scheme, Policy Domain, and a WebGate Profile for Identity Assertion   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. **Applications Deployed on a Stand Alone WebLogic Server**: Obtain OAMCfgTool, as follows.

   a. Log in to Oracle Technology Network at:

      http://www.oracle.com/technology/software/products/middleware/htdocs/111110_fmw.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 10–2. For example:

```
java -jar oamcfgtool.jar mode=CREATE app_domain=IASSO_App1
protected_uris=/myapp/login
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>
```

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 10–3). For example:

```
java -jar oamcfgtool.jar mode=VALIDATE app_domain=IASSO_App1
protected_uris=/myapp/login
ldap_host=wxyz
ldap_port=6633
ldap_userdn=orcladmin
ldap_userpassword=<ldap_userpassword>
oam_aaa_host=abcd
oam_aaa_port=7789
log_file=OAMCfg_date.log
log_level=INFO
test_username=gcf
test_userpassword=<test_userpassword>
```

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 System 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 System Administration Guide.

11. Repeat Steps 3 through 8 for each application that you are protecting.

12. Proceed to "Configuring Providers in the WebLogic Domain".

10.2.4.3 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

10.2.4.3.1   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

10.2.4.3.2 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

10.2.4.3.3 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 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 10–4 defines the expected value for each argument in the addOAMSSOProvider command line.

Table 10–4   addOAMSSOProvider Command-line Arguments

Argument	Definition
loginuri	Specifies the URI of the login page
logouturi	Specifies the URI of the logout page
	Note: For ADF security enabled applications, the logouturi value should be empty (or the word None) to ensure that cookies that could prevent the user from logging out (ObSSOCookie, for instance) are cleared. If a logout URI is specified with ADF security enabled, cookies are not cleared.
autologinuri	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=None, 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 Oracle Access Manager Identity Asserter

   ■ Testing Oracle Access Manager Identity Assertion for Single Sign-on

   ■ Configuring Global Logout for Oracle Access Manager

6. Run the application.

10.2.4.3.4 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 10-31

The following procedure uses the WebLogic Administration Console.

> **Note:** If you have 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. **Applications Deployed on a Stand Alone WebLogic Server**: Obtain the Oracle Access Manager provider:

   a. Log in to Oracle Technology Network at:

   http://www.oracle.com/technology/software/products/middleware/htdocs/111110_fmw.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. Log in to the WebLogic Administration Console.

3. Click **Security Realms**, *Default Realm Name*, and click **Providers**.

4. **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

5. **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.

6. **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.

7. 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

8. **Activate** Changes: In the Change Center, click Activate Changes

9. Reboot Oracle WebLogic Server.

10. Proceed as follows:

- Successful: Go to "Setting Up the Login Form for the Oracle Access Manager Identity Asserter".

- 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 "Required Components and Files" on page 10-6.

10.2.4.4 Setting Up the Login Form for the Oracle Access Manager Identity Asserter

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 10–11 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 10–11   Default Login Form for Single Sign-On

> **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 /oamsso "/oam/webgate/access/oamsso"
<LocationMatch "/oamsso/*">
Satisfy any
</LocationMatch>
```

The following procedure guides as you set up the login form for your environment.

To set up the login form for Oracle Access Manager Identity Assertion

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/oamsso/login.html

2. From your browser, go to the following URL:

   http://*WebGatehost:port*/oamsso/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 "Testing Oracle Access Manager Identity Assertion for Single Sign-on".

10.2.4.5 Testing Oracle Access Manager Identity Assertion for Single Sign-on

The following procedure describes how to test your Oracle Access Manager Identity Assertion setup.

Alternatively, you can run Access Tester in Oracle Access Manager to test your policy domain, as described in the *Oracle Access Manager System Administration Guide*.

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 for Provider Deployment" on page 10-61.

3. Proceed to "Configuring Global Logout for Oracle Access Manager" on page 10-57.

10.2.5 Configuring the Oracle Access Manager Authenticator

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 Required Components for Oracle Access Manager Providers" on page 10-10 must be completed:

■ Installing Components and Files which includes manually creating AccessGate profiles in the Access System Console and accepting defaults during Policy Manager setup

> **See Also:**
>
> ■ "About Oracle Access Manager Installation and Setup" on page 10-11
>
> ■ "About WebGate/AccessGate Profiles and Policy Domains" on page 10-11

■ Creating Resource Types in Oracle Access Manager

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

10.2.5.1 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.

> **Note:** You must perform this task in the Access System Console of Oracle Access Manager. You cannot use OAMCfgTool for this task.

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.

This topic is divided as follows:

- About Oracle Access Manager Authentication Schemes

- Creating an Authentication Scheme in Oracle Access Manager

10.2.5.1.1 About Oracle Access Manager Authentication Schemes Each policy domain in Oracle Access Manager must include at least one authentication rule that includes one Oracle Access Manager authentication scheme and authentication actions. The Oracle Access Manager Authenticator performs two functions: credential authentication and role-fetching based on the username.

As you create the Oracle Access Manager authentication scheme, you give this scheme a unique name and optional description. The level of the scheme is a number that corresponds to the relative security level for this scheme. Higher levels are considered more secure.

The Oracle Access Manager Authenticator performs two functions: credential authentication and role-fetching based on the username. With the Authenticator, the scheme's challenge method is specified as None, with no challenge parameters. However, each scheme includes one or more steps, each of which can include one or more plug-ins that perform part of the authentication process. A single-step scheme, or a multi-step (chained) scheme, requires an authentication flow.

For more information about authentication schemes, see the chapter on configuring user authentication in the *Oracle Access Manager System Administration Guide*.

10.2.5.1.2 Creating an Authentication Scheme in Oracle Access Manager You can use the following procedure to create an authentication scheme in the Access System Console.

To create an authentication scheme for the Oracle Access Manager Authenticator

1. From the Access System Console, click Access System Configuration, Authentication Management, Add.

2. Create the authentication scheme for the policy domain used by the Authenticator, as follows:

a. **General** tab: Enter and Save the following information for use with Authenticator.

Name: *Username Resolution*

Description: *AuthN Scheme for the Authenticator*

Level: 1

Challenge Method: None

Challenge Parameter:

SSL Required: No

Challenge Redirect: (Leave blank)

Enabled: (Leave as is)

b. **Plugins** tab: Use the `credential_mapping` plug-in from existing Oracle Access Manager authentication schemes.

Table 10–5 Plug-ins for the Authentication Scheme

Plug-in Name	Plugi-in Parameters
credential_mapping	`obMappingBase="o=company,c=us",obMappingFilter="(&(&(objectclass=inetOrgPerson)(uid=%userid%))(|(!(obuseraccountcontrol=*))(obuseraccountcontrol=ACTIVATED)))"`

obMappingBase: The base DN in the user search in the LDAP directory server.

obMappingFilter: The LDAP filter used to search for a user with a given userID. The directory login attribute is an attribute defined in the Identity System using a Semantic login type.

Note: No spaces are allowed in the filter. The Policy Manager does not validate the credential_mapping filter string. If you make a mistake and enter an erroneous filter, no error occurs while saving. However, the filter can fail and the plug-in returns "Authentication Failed" each time it is run.

After you create at least one plug-in, default steps and a default authentication flow are created automatically.

3. Enable the authentication scheme: Click the General tab, click Modify; beside Enable, click Yes, and then click Save.

4. Restart the Access Server.

5. Proceed to "Creating a Policy Domain and Access Policies for the Authenticator".

10.2.5.2 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 user 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 10–12.

Figure 10–12   Create Policy Domain Page in the Oracle Access Manager Policy Manager
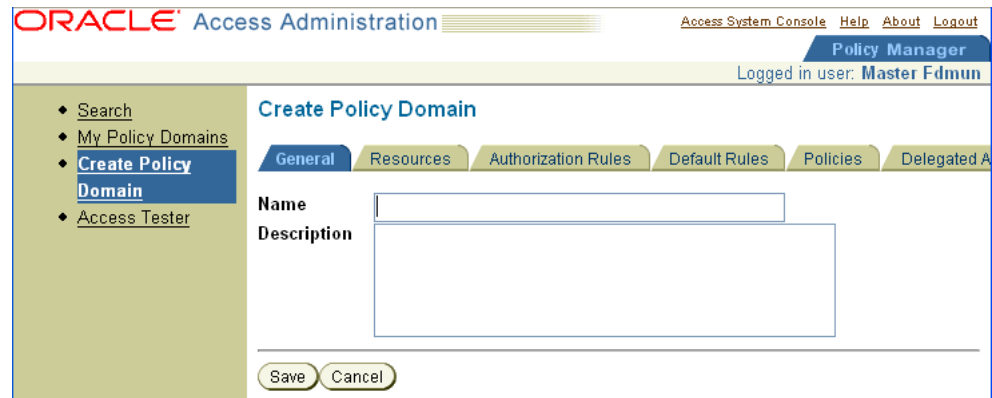
For more information, see the following topics:

- About Creating a Policy Domain

- Creating a Policy Domain and Access Policies for the Authenticator

10.2.5.2.1   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.

  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 System Administration Guide*:
>
> - "Creating an Authentication Rule for a Policy Domain"
> - "Creating an Audit Rule for a Policy Domain"

10.2.5.2.2 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" on page 10-15

- Request user credentials using the Oracle Access Manager authentication scheme created earlier. See also, "Creating an Authentication Scheme in Oracle Access Manager" on page 10-39.

> **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 10-48.

- 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.

> **Note:** The Authenticator does not perform authorization. Therefore 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/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.

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. **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.

8. **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 System Administration Guide*, "Delegating Policy Domain Administration"

9. Proceed with "Configuring Providers for the Authenticator in a WebLogic Domain".

10.2.5.3 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 10-31
>
> - *Oracle Fusion Middleware Oracle WebLogic Scripting Tool*
>
> - *Oracle Fusion Middleware WebLogic Scripting Tool Command Reference*

> **Note:** If you have an Oracle Fusion Middleware application installed, you can skip Step 1.

To configure providers for the Oracle Access Manager Authenticator in a WebLogic domain

1. **Applications Deployed on a Stand Alone WebLogic Server**: Obtain the Oracle Access Manager provider as follows.

    a. Log in to Oracle Technology Network at:

    http://www.oracle.com/technology/software/products/middleware/ht
docs/111110_fmw.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. Click **Lock & Edit**, if desired.

4. **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 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 10-59 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.

5. In the Change Center, click **Activate Changes**.

6. **DefaultAuthenticator**: Under the Providers tab, select **DefaultAuthenticator**, which changes its control flag to SUFFICIENT.

7. **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.

8. **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.

9. Reboot the WebLogic Server.

10. 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 10-67.

11. Proceed with "Configuring the Application Authentication Method for the Authenticator".

10.2.5.4 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".

10.2.5.5 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.

10.2.5.6 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 System 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 Provider Deployment" on page 10-61

10.2.6 Configuring Identity Assertion for Oracle Web Services Manager

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 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 Installation and Setup" on page 10-11
  >
  > - "About WebGate/AccessGate Profiles and Policy Domains" on page 10-11

- Creating Resource Types in Oracle Access Manager

Task overview: Deploying the Identity Asserter with Oracle Web Services Manager includes

1. Ensuring that all prerequisite tasks have been performed

2. Creating a Policy Domain for Use with Oracle Web Services Manager

3. Configuring Oracle Web Services Manager Policies for Web Services

4. Configuring Providers in a WebLogic Domain for Oracle Web Services Manager

5. Testing the Identity Asserter with Oracle Web Services Manager

10.2.6.1 Creating a 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.

> **See Also:** "About Creating a Policy Domain" on page 10-41

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 `wl_authen`, which were defined in "Creating Resource Types in Oracle Access Manager" on page 10-15

- 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: `wl_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 System 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 System 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 Oracle Web Services Manager Policies for Web Services".

10.2.6.2 Configuring Oracle Web Services Manager Policies for Web Services

This section provides an overview of configuring Oracle Web Services Manager policies to protect Web services.

To use the Identity Asserter with Oracle Web Services Manager, you must set up a Web service with the oracle/wss_oam_token_service_policy and a corresponding client with the oracle/wss_oam_token_client_policy in Oracle Web Services Manager.

> **See Also:** "About Using Oracle Access Manager Identity Asserter for Single Sign-on" on page 10-7

About oracle/wss_oam_token_service_policy

This Oracle Web Services Manager policy contains the policy assertion `oracle/wss_oam_token_service_template`. This template uses the credentials in the WS-Security header's binary security token to authenticate users against the Oracle Access Manager identity store.

The Oracle Access Manager Identity Asserter uses the ObSSOCookie token to assert the identity of users who try to access a Web service protected by the `oracle/wss_oam_token_service_policy` policy. A Web service that is protected by this policy must be presented with an ObSSOCookie token in a SOAP header. That is, the Web service consumes the ObSSOCookie token; it is not involved in how the token is generated. Specifically, the WebLogic Server security service detects the token type and invokes the Oracle Access Manager Identity Asserter. The Oracle Access Manager Identity Asserter then validates the ObSSOCookie token against the Oracle Access Manager Access Server and obtains the username. The username is populated as the principal in the authenticated subject.

The Web service client, for example the Web application, must obtain the ObSSOCookie token to send it to the Web service. This is typically done using an AccessGate. AccessGate challenges the Web service client user for credentials (depending on the authentication scheme configured in Oracle Access Manager) and authenticates the user. The WebGate sends the ObSSOCookie to the user's browser upon successful authentication

The Web service client then sends the ObSSOCookie token in the SOAP request to the Web service.

> **Note:** Settings for the `wss_oam_token_service_template` are identical to the client version of the assertion: `wss_oam_token_client_template`. Identity store configuration for the service template is identical to the client version of the assertion.

About oracle/wss_oam_token_client_policy

This Oracle Web Services Manager policy contains the following policy assertion: `oracle/wss_oam_token_client_template`. This template inserts Oracle Access Manager credentials into the WS-Security header as part of the binary security token.

`oracle/wss_oam_token_client_policy` is the analogous client policy to the `oracle/wss_oam_token_service_policy` service endpoint policy. This policy can be enforced on any SOAP-based endpoint.

The following task overview outlines the procedures you must perform.

Task overview: Setting policies in Oracle Web Services Manager

1. Using Oracle Web Services Manager, set up a Web service with the `oracle/wss_oam_token_service_policy` policy.

2. Using Oracle Web Services Manager, set up a corresponding client for the Web service with the `oracle/wss_oam_token_client_policy` policy.

3. Configuring Providers in a WebLogic Domain for Oracle Web Services Manager.

See Also:

- "About Using Oracle Access Manager Identity Asserter for Single Sign-on" on page 10-7

- *Oracle Fusion Middleware Security and Administrator's Guide for Web Services*

  "Configuring Policies"

  "Predefined Assertion Templates"

10.2.6.3 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 setting up providersAuthentication 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 10-31

- *Oracle Fusion Middleware Oracle WebLogic Scripting Tool*

- *Oracle Fusion Middleware WebLogic Scripting Tool Command Reference*

Note: If you have an Oracle Fusion Middleware application installed, you have the required provider JAR file. Skip Step 1.

To set up providers in a WebLogic domain

1. **Applications Deployed on a Stand Alone WebLogic Server**: Obtain the Oracle Access Manager provider, as follows.

   a. Log in to Oracle Technology Network at:

   http://www.oracle.com/technology/software/products/middleware/htdocs/111110_fmw.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 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: *mmmm*

    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 "Required Components and Files" on page 10-6.

10.2.6.4 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 Provider Deployment" on page 10-61.

10.2.7 Configuring Global Logout for Oracle Access Manager

In Oracle Access Manager, global logout or SLO (single log out) is handled in various ways. This section describes the options supported by Oracle Access Manager and different paths to integrate these.

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:

- Zero Configuration SLO
- Configuring the LogoutURLs Parameter in WebGate/AccessGate Profiles
- Application-Managed SLO

10.2.7.1 Zero Configuration SLO

This is a form of WebGate-managed SLO. In this mode, WebGate logs out any request for 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.

The requirement is to include a logout.jsp or logout.html as part of the application resource. No special policies are required to "unprotect" this URL. There is no need to include any logic within the page to delete any cookies.

To implement zero-configuration SLO

1. Include a logout.jsp or logout.html as part of the application resource

2. Optionally: The logout page could simply redirect back to the application Home page.

10.2.7.2 Configuring the LogoutURLs Parameter in WebGate/AccessGate Profiles

This is another form of WebGate-managed SLO. You can designate a set of logout URLs and set them as LogoutURLs parameter in the WebGate/AccessGate profile.

Caveat: WebGate matching of these URLs does not include query parameters. There is no requirement to have any logic in the logout page to clear the cookie explicitly. For example, a valid URL is illustrated in the first line while the second line has an example of an entry that cannot be matched as a LogoutURL:

```
/myapp/myapp.action/signout
/myapp/myapp.action/do?logout=true
```

> **Note:** Once the LogoutURLs parameter is specified, zero configuration SLO (matching of "logout." string) functionality ceases. Essentially, you can use either zero configuration SLO or specify LogoutURLs but not both.

10.2.7.3 Application-Managed SLO

If you do not want the application to rely on WebGate managed SLO, you can handle SLO in an application-specific manner. In this mode, the application has a logout page resource defined. The page logic includes clearing the ObSSOCookie.

A logout.html template is shipped with every WebGate. Applications can use the template as reference. In this mode, it is also essential for the application to "unprotect" access to the logout resource using Oracle Access Manager policies. The application can also "brand" the page and also handle redirects to its home page as part of logout logic.

To implement application-managed SLO

1. Ensure that your application references the WebGate logout.html template.

2. Ensure that no Oracle Access Manager policies for the application protect access to the logout resource.

3. Optional: Allow the application to branch the page and handle redirects to the application Home page as part of logout logic.

10.2.8 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 10–6, " Oracle Access Manager Authentication Provider Common Parameters"

- Table 10–7, " Provider-Specific Parameters for Identity Asserter for Single Sign-On"

- Table 10–8, " Provider-Specific Parameters: Oracle Access Manager Authenticator"

Table 10–6   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 to ObSSOCookie.
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 10–7 lists the provider-specific parameters for Oracle Access Manager Identity Asserter.

Table 10–7   Provider-Specific Parameters for Identity Asserter for Single Sign-On

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.

Table 10–7   (Cont.) Provider-Specific Parameters for Identity Asserter for Single Sign-On

Parameter Name	Parameter Description
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 *host:port*. Required. See "Installing and Setting Up Required Components for Oracle Access Manager Providers" on page 10-10.
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 *host:port*. See "Installing and Setting Up Required Components for Oracle Access Manager Providers" on page 10-10.
Key Store	The absolute path of JKS key store used for SSL communication between the provider and the Oracle Access Manager Access Server.

Table 10–8 lists provider-specific parameters for the Oracle Access Manager Authenticator.

Table 10–8   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 *host:port*. See "Installing and Setting Up Required Components for Oracle Access Manager Providers" on page 10-10.

Table 10–8   (Cont.) Provider-Specific Parameters: Oracle Access Manager Authenticator

Parameter Name	Parameter Description
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 *host:port*. Required. See "Installing and Setting Up Required Components for Oracle Access Manager Providers" on page 10-10.

10.2.9 Known Issues: JAR Files and OAMCfgTool

Table 10–9 identifies known issues with this release. For more information about the tool, parameters, and values, see "About Using OAMCfgTool" on page 10-20.

Table 10–9   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 and Validate options. It does not provide Delete or Overwrite options.
8362039	OAMCfgTool does not provide explicit options to specify the Web Tier host and port. Instead, without web_domain specified the app_domain value specifies the WebGate name, host, and Preferred HTTP Host. For example: ■ app_domain=*ABC* (without web_domain specified) ■ AccessGate Name: *ABC*_AG ■ Hostname: *ABC* ■ Port: Not specified ■ Preferred HTTP Host: *ABC*
n/a	With OAMCfgTool, if web_domain parameter is included in the command line, you must provide a WebGate password. Otherwise, the command can fail. The app_agent_password parameter accepts as the password whatever follows the equal sign, =. For instance, if you enter app_agent_password= and then enter a space character and web_domain=*value*, the app_agent_password is presumed to be a space character followed by web_domain.
n/a	SSL-enabled communication with the directory server is not supported by OAMCfgTool.

10.2.10 Troubleshooting Tips for Provider Deployment

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 10-95

10.2.10.1 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.

10.2.10.2 Apache Bridge Failure: Timed Out

If you experience a failure of the Apache bridge, you might see a message stating that there is no backend 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>
```

10.2.10.3 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.

10.2.10.4 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 reposted, 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.

10.2.10.5 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.

10.2.10.6 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:** "About Using OAMCfgTool" on page 10-20

If you have an Oracle Fusion Middleware Application installed you already have the OAMCfgTool. In this case, skip Step 1.

Solution:

1. **Applications Deployed on a Stand Alone WebLogic Server**: Obtain OAMCfgTool, as follows.

   a. Log in to Oracle Technology Network at:

      http://www.oracle.com/technology/software/products/middleware/ht docs/111110_fmw.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 System 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 System Administration Guide.

9. In the load balancer host identifiers, add host name variations for both WebGates: WebGateA and WebGateB.

10.2.10.7 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 10-59.

10.2.10.8 Error 403: Unable to Access the Application

An error message like the following:

```
403 Forbiden
```

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 10-40.

10.2.10.9 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.

10.2.10.10 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

10.2.10.11 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.

10.2.10.12 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 10-45.

10.2.10.13 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`.

10.2.10.14 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 Oracle Access Manager Identity Assertion for Single Sign-On" on page 10-16.

10.2.10.15 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>
    ```

10.2.10.16 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.

10.2.10.17 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.

10.2.10.18 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 with Oracle Access Manager, you can create a policy for matching the URL with JSESSIONID. For this example, suppose the protected URL is:

```
/myapp/login
```

You can create a policy within the appropriate policy domain to match the URL with JSESSIONID.

To avoid issues matching a re-written URL

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.

2. Click Policy Manager., click My Policy Domains, and click the link to the desired policy domain.

3. Click the Policies tab, and then click Add.

   Fill in and save **General** details:

   > Name: *JSESSIONID Matching Policy*
   >
   > Resource operation(s): Select all applicable HTTP operations
   >
   > Resource: Select the context root (create it if needed). For example: `/myapp`
   >
   > URL Pattern: `login`
   >
   > Click Save.

10.3 Deploying the OracleAS 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 10-73.

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

10.3.1 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

10.3.1.1 Oracle WebLogic Security Framework

Figure 10–13 illustrates the location of components in the Oracle WebLogic Security Framework, including the OSSO Identity Asserter. Additional details follow.

Figure 10–13   Location of OSSO Components in the Oracle WebLogic Security Framework
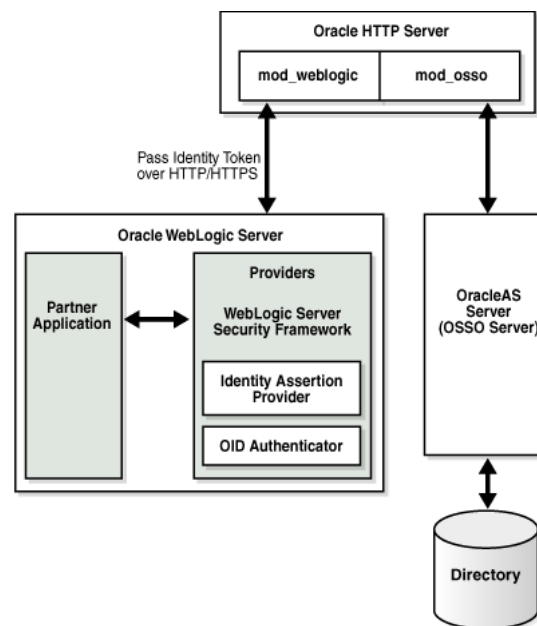
At the top of the figure, Oracle HTTP Server is installed. This installation includes mod_weblogic and mod_osso, which are required to pass the identity token to the Providers and Oracle WebLogic Server. The Oracle WebLogic Server includes the partner application and the Identity Asserter (also known 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)

10.3.1.2 OSSO Identity Asserter Processing

Figure 10–14 illustrates the processing that occurs when you have OSSO implemented with the Identity Asserter. Additional details follow the figure.

Figure 10–14   OSSO Identity Asserter Processing
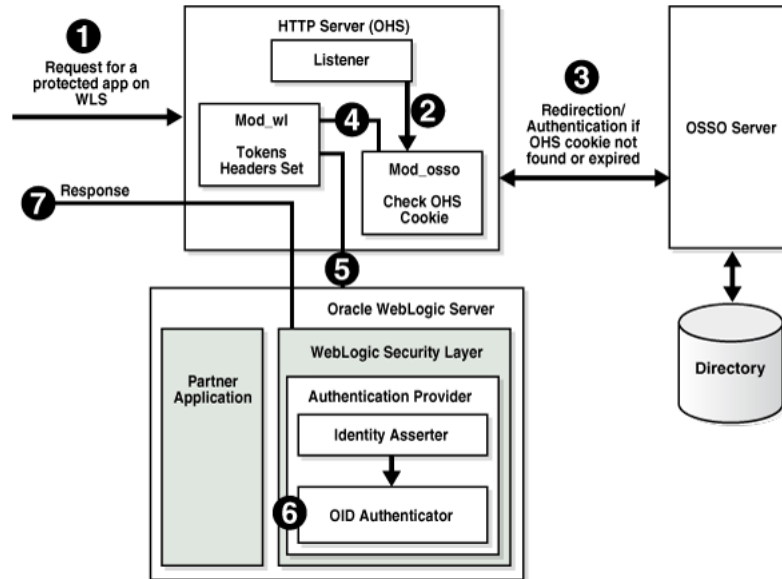
The first time a request for a protected resource arrives at the mid-tier Web server, the request is redirected to the 10g OracleAS Single Sign-On server, which requires user credentials For a certificate-based authentication, no login page is displayed. After the user has been successfully authenticated, 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.

10.3.1.3 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 10–10 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 10–10   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.

10.3.2 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/oim10 14.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 10-75.

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 10-76.

4. Configure mod_osso, as described in "Configuring mod_osso to Protect Web Resources" on page 10-77.

5. Add the OSSO Identity Asserter to the appropriate domain, as explained in section "Adding Providers to a WebLogic Domain for OSSO" on page 10-81.

6. Configure a connection filter, as explained in section "Establishing Trust Between Oracle WebLogic Server and Other Entities" on page 10-83.

7. Configure the use of the solution by the application, as explained in section "Configuring the Application for the OSSO Identity Asserter" on page 10-84.

8. Identify and resolve issues with your OSSO Identity Asserter implementation, see "Troubleshooting for an OSSO Identity Asserter Deployment" on page 10-85.

10.3.2.1 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_INSTANCE}/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>
```

10.3.2.2 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 10–11 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 10–11   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:
	http://*oracle_http_host.domain:port*
-update_mode	Optional. CREATE, the default, generates a new record.
-remote_midtier	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 Administer 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/oim10
> 14.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_midtier -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".

10.3.2.3 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)

10.3.2.3.1 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".

10.3.2.3.2 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 10–1, "SSO Authentication with Dynamic Directives"

- Example 10–2, "SSO Logout with Dynamic Directives"

> **Note:** After adding dynamic directives, be sure to restart the Oracle HTTP Server, and the proceed to "Adding Providers to a WebLogic Domain for OSSO".

Example 10–1   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/i
> as/htdocs/101401.html

Example 10–2   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 10-93
>
> ■ *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/product
>   s/ias/htdocs/101401.html

> **Note:**   After adding dynamic directives, be sure to restart the Oracle
> HTTP Server, and the proceed to "Adding Providers to a WebLogic
> Domain for OSSO".

10.3.2.4 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 10-31

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 10-31
>
> - *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".

10.3.2.5 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 10–12 provides a description of each parameter in a connection filter.

Table 10–12   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".

10.3.2.6 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.

10.3.3 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 10-95
>
> - *Oracle Application Server Single Sign-On Administrator's Guide* for 10g, Troubleshooting, on the Oracle Technology Network at: http://www.oracle.com/technology/documentation/oim1014.html

10.3.3.1 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 Standalone 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)

```
OssoConfigFile  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>
OssoIdleTimeout off
OssoIpCheck on
OssoConfigFile  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 Standalone WebLogic Server

The *Oracle Fusion Middleware Security Guide* 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 are provided here only.

Differences between the two environments include:

- **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:**   Stand-Alone Oracle WebLogic Server with OSSO requires Oracle HTTP Server 11g Web server.

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" in the chapter "Configuring Single Sign-On in Oracle Fusion Middleware" of the *Oracle Fusion Middleware Security Guide*.

Also included in the following note are additional, optional details 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 also be acquired from the Oracle Web Tier, as described here.

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 10-88.

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 10-75.

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 10-73.

a. Perform steps described in "Registering Oracle HTTP Server mod_osso with OSSO Server 10.1.4" on page 10-76.

b. Perform steps described in "Configuring mod_osso to Protect Web Resources" on page 10-77.

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 10-81.

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 10-83.

> **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 10-84.

> **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 10-93 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 nd 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 is 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 Security Guide*.

10.3.3.2 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 10-84.

10.3.3.3 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>
```

10.3.3.4 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 10-77
>
> ■ *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 When Redirecting to OSSO for Logout

10.3.3.4.1 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>
OssoConfigFile  conf/osso.conf
OssoHTTPOnly On
---
---
---
<Location /osso>
AuthType Osso
---
---
</Location>
```

```
</VirtualHost>
```

10.3.3.4.2 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"
```

10.3.3.4.3 Mod_osso Does Not Encode the Return URL When Redirecting to OSSO for Logout
Mod_osso does not URL 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)

10.3.3.5 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.

10.4 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.

- **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 10-80. For details about Oracle Access Manager logout, see "Configuring Global Logout for Oracle Access Manager" on page 10-57.

- **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 10–13.

Table 10–13   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	■ 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 -Dsso.filter.ssotoken=HEADERNAME, and restart the server.

When you contact Oracle Support you might be requested to set up debugging. The following procedure describes how to do this.

10.5 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 the following to expand: **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>
```

11

Introduction to Oracle Fusion Middleware Audit Framework

In Oracle Fusion Middleware 11*g* 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

11.1 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

11.1.1 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.

11.1.2 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.

11.1.3 Oracle Fusion Middleware Audit Framework in 11g

Oracle Fusion Middleware Audit Framework is a new service in 11*g* Release 1 (11.1.1), 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.

- JavaEE applications - The objective is to provide a framework for JavaEE applications, starting with Oracle's own components. JavaEE applications will be able to create application-specific audit events.

  In 11*g* Release 1 (11.1.1), the audit framework is only available for Oracle's own applications.

- 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*.

11.2 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 13, "Using Audit Analysis and Reporting" for details.

- Audit record storage

  Data store (database) and files (bus-stop) are available. Maintaining a common location for all audit records simplifies maintenance.

  Using a 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 11*g* infrastructure

 - is usable across Oracle Fusion Middleware 11*g* 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

 - integrates with Oracle Platform Security Services to provide multiple benefits

11.3 Oracle Fusion Middleware Audit Framework Concepts

This section introduces basic concepts of the Oracle Fusion Middleware Audit Framework:

- Audit Architecture

- Key Technical Concepts

- Audit Record Storage

- Analytics

11.3.1 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.

- 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 store. In the event that no audit 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 store is in place, the bus-stop acts as an intermediary between the component and the audit store. The local files are periodically uploaded to the data store based on a configurable time interval.

A key advantage of the audit 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 store, if one is configured. For Java component auditing, the audit loader is 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 Store

The audit 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 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.

■ 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 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 11–1   Audit Event Flow
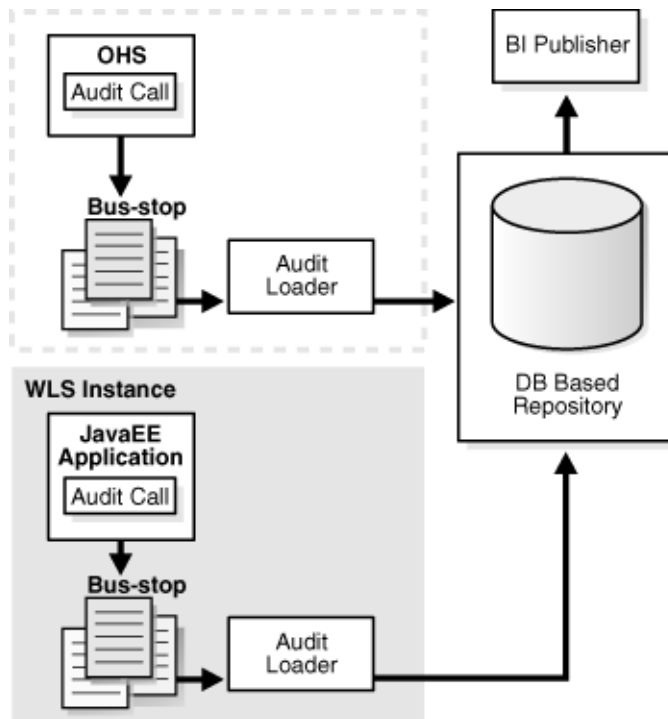
Audit Flow

The process can be illustrated by looking at the actions taken in the framework when an event (say, login) occurs at a component like Oracle HTTP Server or Oracle Virtual Directory within an application server instance:

> **Note:**   The architecture shown in Figure 11–1 contains a data store; if your site did not configure a data store for auditing, the audit records reside in the bus-stop files.

1. Oracle Fusion Middleware Audit Framework is activated for a component when the component starts up.

2. The component calls an audit function to audit the event.

3. The framework checks if events of this type, status, and with certain attributes need to be audited.

4. 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.

5. The event is stored on a local file in an intermediate location known as the bus-stop; each component has its own bus-stop.

6. The next component in the flow is the Audit Loader, a which is 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.

   If a database is configured for an audit store, the audit loader pulls the events from the bus-stops and moves the data to the audit store.

7. Reports can also be generated from the audit data using Oracle BI Publisher. A set of pre-defined reports are available. (See Chapter 13, "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.

11.3.2 Key Technical Concepts

This section introduces key concepts in the Oracle Fusion Middleware Audit Framework.

> **See Also:** *Oracle Fusion Middleware Administrator's Guide* for details about these concepts.

11.3.2.1 Building Blocks of the Framework

This section introduces key aspects of 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 be integrate d with the Oracle Fusion Middleware Audit Framework through configuration with the `jps-config.xml file`.

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".

Oracle Platform Security Services

Oracle Platform Security Services, a key component of the Oracle Fusion Middleware 11*g*, is the Oracle Fusion Middleware security implementation for Java features such as Java Authentication and Authorization Service (JAAS) and JavaEE security.

For more information about OPSS, see Section 2.1, "What is Oracle Platform Security Services?".

11.3.3 Audit Record Storage

As shown in Figure 11–1, 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 JavaEE 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.

    > **See Also:** "Audit Flow" in Section 11.3.1, "Audit Architecture".

- permanent storage in a database; this is known as the audit store.

    If using a database, audit records generated by all components in all Oracle Fusion Middleware 11*g* instances in the domain are stored in the same store. You must use an audit store to utilize Oracle Business Intelligence Publisher reports.

You can move from file-based storage to an audit store. This requires a specific configuration procedure. See Section 12.2.3, "Configure a Database Audit 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 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.

11.3.4 Analytics

With Oracle Fusion Middleware 11*g*, you can utilize Oracle BI Publisher 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 BI Publisher

- You can select records based on criteria like username, date-time range, and so on.

  Note that Oracle BI Publisher works with the database audit store only, and is not usable with bus-stop files.

The pre-defined audit report types available with Oracle BI Publisher include:

- errors and exceptions
- operational
- user activity
- authentication and authorization history
- transaction history
- and more

  See Also: Section C.2, "Pre-built Audit Reports"

You can also use the audit schema details to create custom audit reports as needed.

12

Configuring and Managing Auditing

This chapter explains how to perform day-to-day audit administration tasks.

> **See Also:** Chapter 11, "Introduction to Oracle Fusion Middleware Audit Framework" for background information about auditing in Oracle Fusion Middleware.

- Audit Administration Tasks
- Managing the Audit Store
- Managing Audit Policies
- Audit Logs
- Advanced Management of Database Store

12.1 Audit Administration Tasks

The audit administrator should plan the site's audit setup carefully by following the below steps.

- Implementation Planning

  This includes planning the type of store to use for audit records, data store configuration details, and so on.

  See Section 12.2, "Managing the Audit 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 12.3, "Managing Audit Policies" for details.

- Reports Management

  This includes planning for and configuring audit reports and queries.

  See Chapter 13, "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 12.5, "Advanced Management of Database Store" for details about audit store administration.

12.2 Managing the Audit 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 store to provide scalability and high-availability for the audit framework.

In addition, an audit 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 11*g* Release 1 (11.1.1) CD pack.

This section explains these audit store management tasks in detail:

- Create the Audit Schema using RCU
- Set Up Audit Data Sources
- Configure a Database Audit Store for Java Components
- Configure a Database Audit Store for System Components
- Tuning the Bus-stop Files

12.2.1 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 store for audit records (see Section 12.2.3.2, "Configure the Audit 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 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.

12.2.2 Set Up Audit Data Sources

As explained in Section 12.2.1, "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 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**.

12.2.2.1 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*

12.2.3 Configure a Database Audit Store for Java Components

After the schema is created, configuring a database-based audit store involves:

- creating a data source that points to the audit schema you created and

- configuring the audit store to point to the data source

This section describes the following tasks related to audit store configuration:

- View Audit Store Configuration

- Configure the Audit Store

Note:

These steps configure the audit store for Java components only. Separate steps are needed to configure the audit store for system components. See Section 12.2.4, "Configure a Database Audit 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.

12.2.3.1 View Audit Store Configuration

Note: This task is performed with Oracle Enterprise Manager Fusion Middleware Control.

To view the current audit store configuration, navigate to *Domain*, then **Security**, then **Audit Store**.
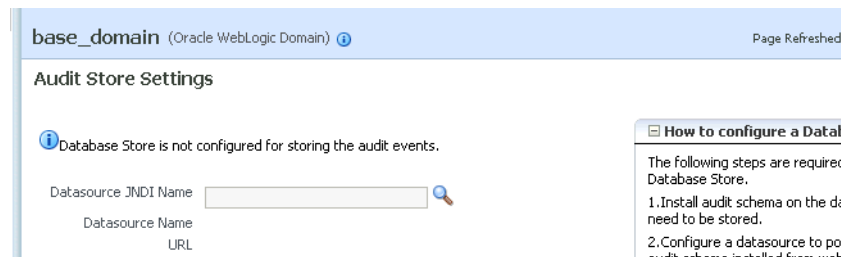
- whether or not a database is configured as the audit store. By default a database is not configured, and audit records are stored in bus-stop files.

- 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 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 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 store is file-based.

See Section 12.2.2, "Set Up Audit Data Sources" for datasource examples.

12.2.3.2 Configure the Audit Store

You can change from storing audit records in a file to using a database audit store.

Take these steps to configure the audit 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 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 12.3.1, "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 in $DOMAIN_HOME/servers/$SERVER_NAME/logs/.

12.2.3.3 Deconfigure the Audit Store

Since a database is the recommended store for audit records, switching from database to file mode is discouraged. However, Section 12.3.4, "Manage Audit Policies Manually" discusses a property called the audit.repositoryType which allows you to switch to a file by setting its value to 'File'.

> **Note:** You cannot use Fusion Middleware Control or WLST to switch from database to file mode; this requires manual configuration as explained in Section 12.3.4, "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.

12.2.4 Configure a Database Audit 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 store is handled by OPMN.

> **Note:** If your system component runs in a clustered deployment, you must configure the audit store at each instance of the component so that all instances push out records to the store.

> **Note:**
>
> These steps configure the audit store for system components only. Separate steps are needed to configure the audit store for Java components. See Section 12.2.3, "Configure a Database Audit 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.

You must execute the following steps in every instance of the component to configure an audit store:

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
                $ORACLE_HOME/modules/oracle.osdt_11.1.1/osdt_cert.jar:
                $ORACLE_HOME/modules/oracle.osdt_11.1.1/osdt_core.jar:
                $ORACLE_HOME/jdbc/lib/ojdbc5.jar:
                $ORACLE_HOME/modules/oracle.iau_11.1.1/fmw_audit.jar:
               $ORACLE_HOME/modules/oracle.pki_11.1.1/oraclepki.jar
          -Doracle.home=$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 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 you have `ORACLE_HOME` and `ORACLE_INSTANCE` defined. For example:

```
ORACLE_HOME = /u01/oracle/as11_oh,
ORACLE_INSTANCE = /u01/oracle/instances/instance
```

6. Populate the audit 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
     $ORACLE_HOME/modules/oracle.osdt_11.1.1/osdt_cert.jar:
     $ORACLE_HOME/modules/oracle.osdt_11.1.1/osdt_core.jar:
     $ORACLE_HOME/jdbc/lib/ojdbc5.jar:
     $ORACLE_HOME/modules/oracle.iau_11.1.1/fmw_audit.jar:
     $ORACLE_HOME/modules/oracle.pki_11.1.1/oraclepki.jar
     -Doracle.home=$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
```

12.2.4.1 Deconfigure the Audit 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 store through the `opmn.xml` file to update the RMD definition to deconfigure the audit 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 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.

12.2.5 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 12.3.4, "Manage Audit Policies Manually".

System Components

The size of a file for the file storage mode can be set in the auditconfig.xml file. SeeSection 12.3.4.4, "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 12.3.4, "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 12.3.4.4, "Manually Configuring Audit for System Components".

12.3 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 Require Server or Instance Restart

When creating or updating audit policies, note that:

- for Java components, policy changes at the domain level require that all managed servers (on which the affected components and applications are running) be restarted.

- for system components, policy changes require the affected component instance to be restarted.

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

> **See Also:**
>
> - Section 11.3.2, "Key Technical Concepts" for additional background.
>
> - Appendix D, Oracle Fusion Middleware Audit Framework Reference for a list of Java components and system components.

12.3.1 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 12.3.2, "Manage Audit Policies for System Components with Fusion Middleware Control"
>
> - See the note at the beginning of Section 12.3, "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.
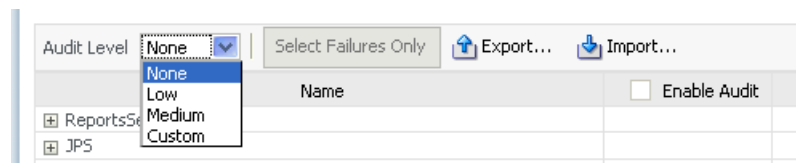
> **Note:** The table of events under the drop-down box cannot be edited for the pre-defined levels. It can only be edited in custom level.

- 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.

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 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.

Click on the icon to bring up the Edit Filter dialog.

> **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", 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.

> **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.

12.3.2 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.

> **Note:** ■ Audit policy for Java components is managed in the domain context. See Section 12.3.1, "Manage Audit Policies for Java Components with Fusion Middleware Control"
>
> ■ See the note at the beginning of Section 12.3, "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	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	Enable Audit	Filter	Edit Filter
⊟ Oracle Internet Directory	☐		
⊟ UserSession	☐		
⊟ UserLogin	☐		
Success	☐		✎
Failure	☐		✎
⊟ UserLogout	☐		
Success	☐		✎
Failure	☐		✎

Click on the icon to bring up the Edit Filter dialog.
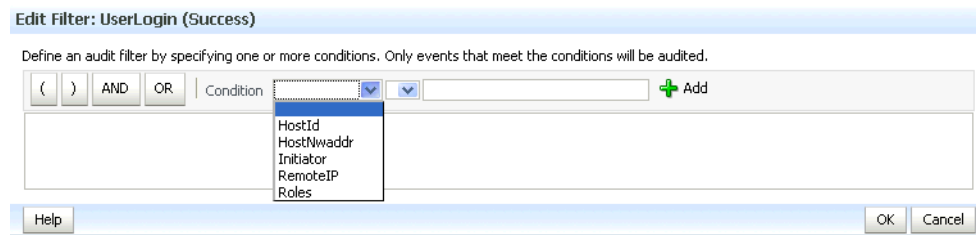
> **Note:** Each filter attribute has a formal name and a display name. You may see either name in the filter edit dialog. Display names are shown in the drop-down, while names are shown in the edit dialog. For example, if you select 'Client Address IP' in the drop-down box, it is renamed to 'RemoteIP' after you add it to the filter expression.

7. Click "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.

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", 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.

> **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.

Click **Revert** to discard any policy changes and revert to the existing policy.

12.3.3 Manage Audit Policies with WLST

This section explains how to view and update audit policies using WLST:

■ 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.

12.3.3.1 View Audit Policies with WLST

Take these steps to view 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')
```

- 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 `getNonJavaEEAuditMBeanName` command. See Section C.4.1, "getNonJavaEEAuditMBeanName" 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=CSAuditMBean,name=CSAuditProxyMBean")
```

12.3.3.2 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.

12.3.3.3 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"

12.3.3.4 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.

12.3.3.5 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.

12.3.4 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](#)

12.3.4.1 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
```

12.3.4.2 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 12–1. Taken together, the set of properties and their values are known as the audit policy:

Table 12–1 Audit Properties in jps-config.xml

Property	Description	Example
audit.filterPreset	the level of auditing - None, Low, Medium, and Custom	
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.	JPS:CheckAuthorization, CreateCredential; OIF:UserLogin
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".
audit.loader.repositoryType	store type for the audit events. If type is Database (Db), audit.loader.jndi property must also be defined.	File or DB
audit.loader.jndi	For DB, the jndi datasource where audit events will be uploaded	jdbc/AuditDB
audit.loader.interval	Controls the frequency of audit loader's upload to database. Integer is in Seconds.	15
audit.maxDirSize	Controls the size of the directory where the audit files will be written. Integer is in Bytes.	102400000
audit.maxFileSize	Controls the size of a bus stop file where audit events are written. Integer is in Bytes.	10240000

Example jps-config.xml file

Here is a sample file illustrating an audit policy:

```xml
<?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>
```

12.3.4.3 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 12–1.

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 database when you switch to database store again.

12.3.4.4 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>
```

12.4 Audit Logs

This section contains the following topics:

- Location of Audit Logs
- Audit Log Timestamps

12.4.1 Location of 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.

For a listing of all audit log locations, how to configure the loggers, and how to use the logs to diagnose issues, see Section I.1.1.3, "Audit Diagnostic Log Files".

12.4.2 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.

12.5 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*.

12.5.1 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 12–1 shows the data in the base table and how it relates to the component-specific tables.

Figure 12–1   Audit Schema
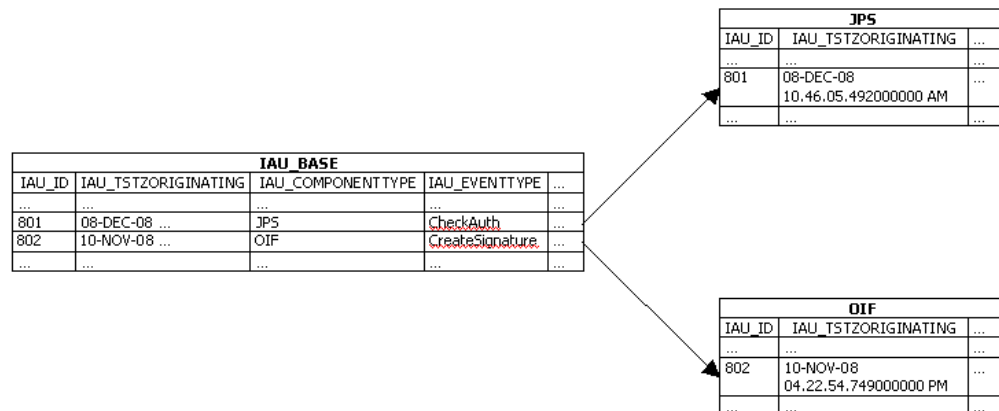
The average record size in the base table `IAU_BASE` is approximately 0.3 KB. When you plan for tablespace sizing:

- 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.

12.5.2 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 12–2 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 12–2   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`)
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"

12.5.3 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).

12.5.4 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.

12.5.5 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 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/htd ocs/data_pump_overview.html

12.5.6 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.

12.5.6.1 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 store to minimize the application down time.

> **Note:** Two sample SQL scripts are shipped with the product:
>
> - `$RCU_ HOME/rcu/integration/iau/scripts/convertPartition edTables.sql` (linux) or `%RCU_ HOME\rcu\integration\iau\scripts\convertPartition edTables.sql` (Windows) converts the base and component tables in audit schema into partitioned tables
>
> - `$RCU_ HOME/rcu/integration/iau/scripts/createPartitions ByQuarter.sql` (linux) or `%RCU_ HOME\rcu\integration\iau\scripts\createPartitions ByQuarter.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 12.2.4.1, "Deconfigure the Audit 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.

12.5.6.2 Backup and Recovery of Partitioned Tables

Backup and recovery were discussed in Section 12.5.4, "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.

12.5.6.3 Import, Export, and Data Purge

Import and export were discussed in Section 12.5.5, "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.

12.5.6.4 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

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

13.1 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

13.1.1 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 11*g* 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 I.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

13.1.2 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 13.1.3, "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

13.1.3 Set Up Oracle Reports in Oracle Business Intelligence Publisher

In this section you configure Oracle Business Intelligence Publisher to work with the audit datasource.

> **Note:** 11*g* Release 1 (11.1.1) reports can work with the 11*g* Release 1 (11.1.1.2.0) schema. However, 11*g* Release 1 (11.1.1.2.0) reports can work only with an 11*g* Release 1 (11.1.1.2.0) schema; they cannot work with an 11*g* Release 1 (11.1.1) schema.

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 AuditReportTemplate.jar into your Reports folder. You should see a new folder called Oracle_Fusion_Middleware_Audit. You can find AuditReportTemplate.jar at $MW_ORA_HOME/modules/oracle.iau_
   11.1.1/reports/AuditReportTemplates.jar

3. Set up the DataSource 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.

13.1.4 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.*

13.1.5 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 the following section.

13.1.6 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 13.1.5, "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

13.2 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

  These reports are component-specific. 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.

13.3 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 13–1 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.

13.4 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 9.5, "Managing the Domain Credential Store" for details about credential management.

3. Log in to Oracle XML 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	Event Details
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.

9. 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
Timestamp	2009-01-11 05:58:46 PM GMT
Initiator	weblogic

Request Details

Component Type	JPS
Thread Id	14

Oracle Platform Security

Credential Map	mapSimpleServlet
Credential Key	keySimpleServlet

10. 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.

User ID	Timestamp	Component/ Application Name	Administrator ID	Message	Event	Event Details
cn=fwu	2/27/2009 12.36.57 PM	oid1	cn=orcladmin	Operation name: add	Create Account	Detai d...
cn=linda balls,ou=treasury,ou=fin-accounting ou=americas o=inc,c=us	2/27/2009 10:00:40 AM	oid1	cn=orcladmin	Operation name: add	Create Account	Detai s...
cn=lincoln ball,ou=treasury,ou=fin accounting ou=europe,o=inc,c=us	2/27/2009 10:00:40 AM	oid1	cn=orcladmin	Operation name: add	Create Account	Detai s...
cn=lincoln balls,ou=treasury,ou=fin-accounting ou=americas,o=inc,c=us	2/27/2009 10:00:40 AM	oid1	cn=orcladmin	Operation name: add	Create Account	Detai s...

11. 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	stakb08
Process ID	196

Event Details

Event Category	Account Management
Event Type	Create Account
Event Status	SUCCESS
Timestamp	2009-02-27 12:36:37 PM SST
Initiator	cn=orcladmin
Target	cn=fwu

Request Details

ECID	View Related Audit Events ...
Component Type	OID
Component Name	oid1
Client IP Address	170.87.7.191
Session Id	27

Oracle Internet Directory

Event Operation	add

12. Finally, return to the Common Reports folder and select Errors and Exceptions. Select the **All Errors and Exceptions** report.

13. 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:

14. Click Details to obtain more information:

13.5 Audit Report Details

This section provides detailed reference information about the standard (pre-built) audit report.

The standard audit reports are grouped as follows:

1. The All Events report

This report contains all audit records generated in a pre-defined interval.

2. Common Reports

These are reports that contain audit records across multiple components.

3. 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

Component-Specific Reports

For a list of reports, see Section C.2.2, "Component-Specific Audit Reports".

13.5.1 List of Audit Reports in Oracle Business Intelligence Publisher

Table 13–1 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 13–1   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.
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.

Table 13–1 (Cont.) List of Audit Reports

Report	Description	Located in Folder
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.
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

Table 13–1   (Cont.)   List of Audit Reports

Report	Description	Located in Folder
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

13.5.2 Attributes of Audit Reports in Oracle Business Intelligence Publisher

Table 13–2 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 13–1 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 13–2   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.

Table 13–2   (Cont.)   Attributes of Audit Reports

Attribute	Description
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.

13.6 Customizing Audit Reports

This section discusses advanced report generation and creation options:

- Using Advanced Filters on Pre-built Reports

- Creating Custom Reports

13.6.1 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.

13.6.2 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.
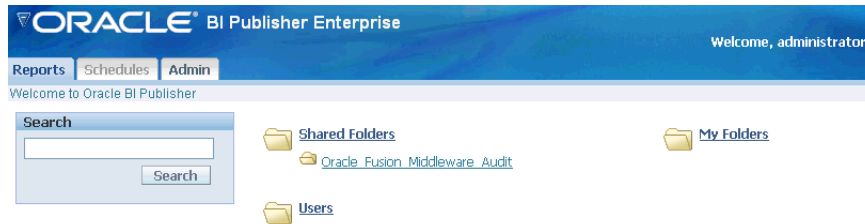
2. Navigate to the Oracle Fusion Middleware Audit folder.
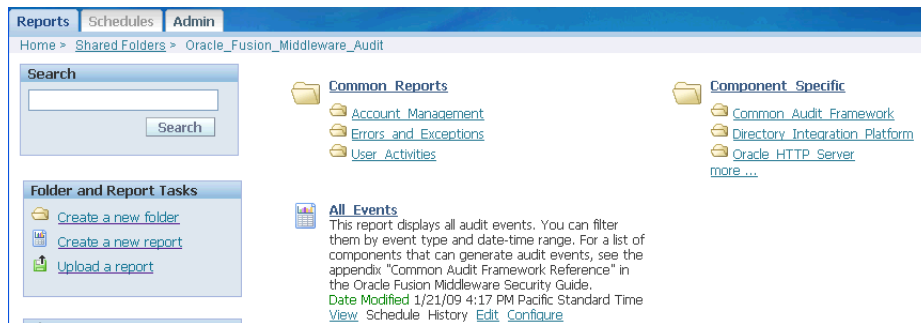
3. Create a folder to maintain your custom reports. Under **Folder and Event Tasks**, click **New Folder**.

    Enter a folder name.
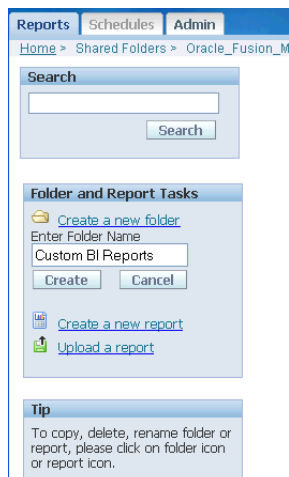
4. The new folder, Custom BI Reports, appears on the main audit reports folder.

5. 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**.

6. 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**.
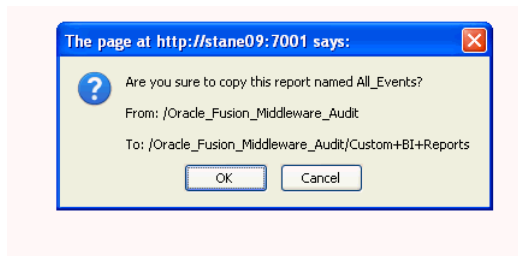
   The report is now moved from the clipboard to the custom folder:

   ■ Provide a descriptive name for the new report by selecting the icon to the left of the report, and clicking **Rename this report**.
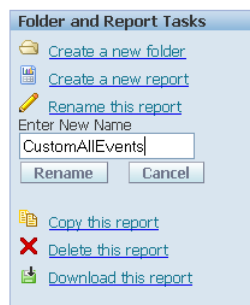
7. Now you are ready to customize the report. Click **Edit** from the menu choices under the report title.

8. The Edit page appears.
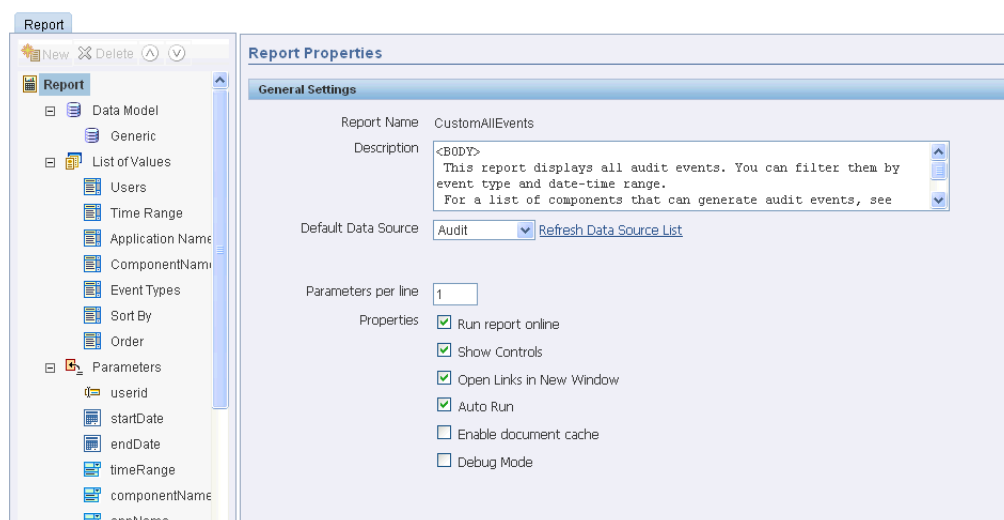
Two panels are displayed; on the main panel titled General Settings, you can control basic features like the report title and runtime controls. To the left of the main panel, a second panel displays two sets of information that you can use to create relevant content for your report:

- 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**.
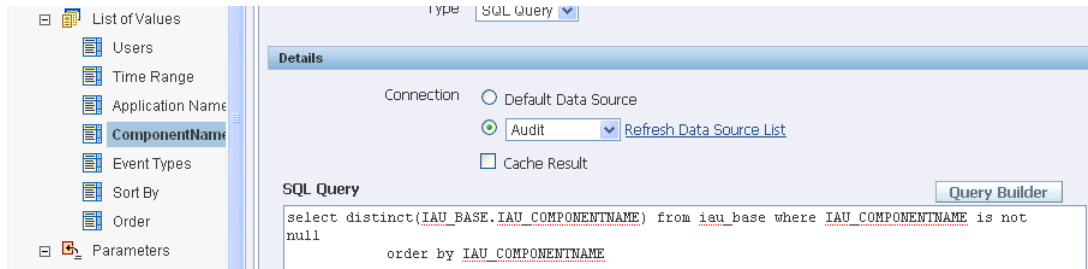
- A table appears listing the available components. Select the component, say `JPS`. A second table appears showing the component event fields:
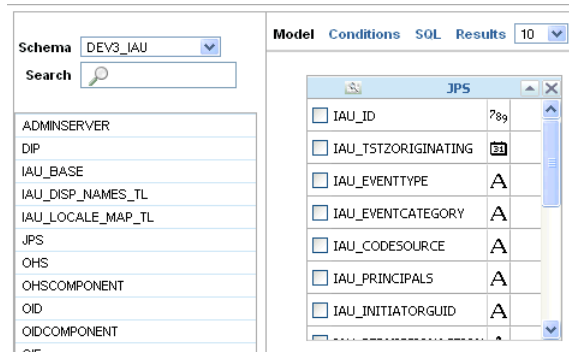
- In the JPS table select `IAU_EVENTTYPE`.
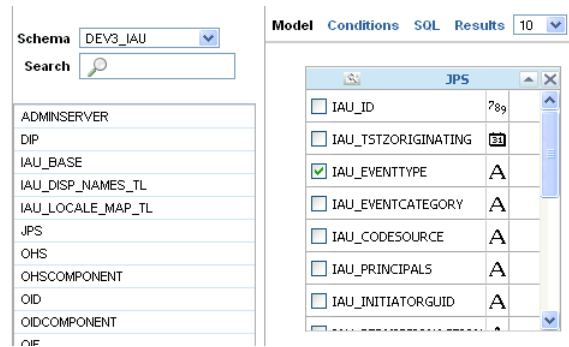
- Click **Conditions**, enter the condition `login` and click **Save**.
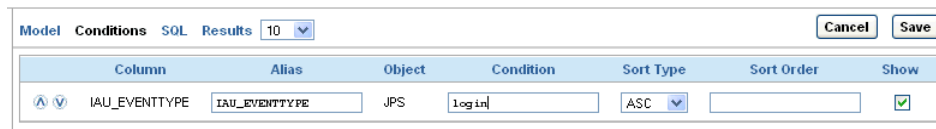
10. The condition is now included in the report. Be sure to click **Save** again on the upper left corner to commit your changes to the report definition.
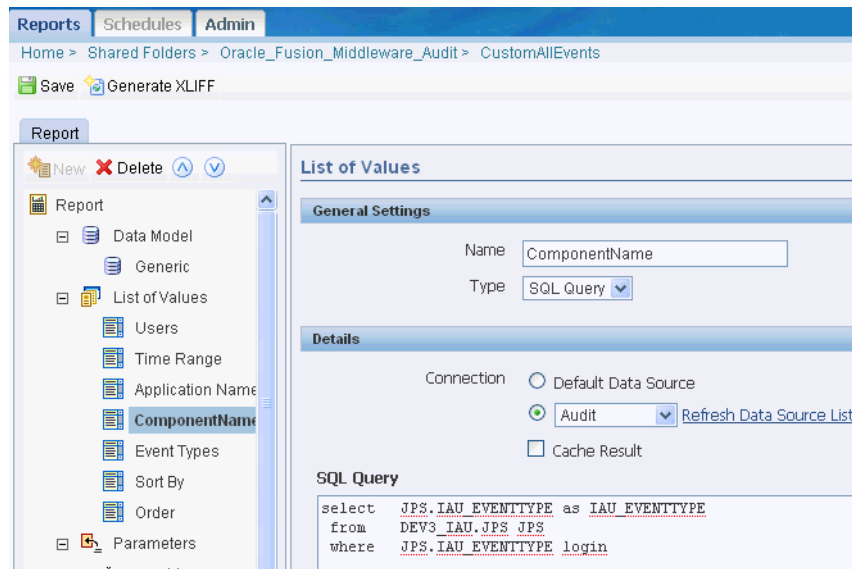
11. You can now return to the report in the Custom BI Reports folder and view the data.

Part IV

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 14, "Overview of Developing Secure Applications with Oracle Platform Security Services"

- Chapter 15, "Manually Configuring JavaEE Applications to Use OPSS"

- Chapter 16, "Developing Authentication"

- Chapter 17, "Developing with the Credential Store Framework"

- Chapter 18, "Developing Authorization"

- Chapter 19, "Developing with the User and Role API"

- Chapter 20, "Developing with Oracle HTTPClient Security"

14

Overview of Developing Secure Applications with Oracle Platform Security Services

This chapter explains the features and benefits of using Oracle Platform Security Services to develop and deploy secure applications to work with Oracle Fusion Middleware.

This chapter includes the following sections:

- About Oracle Platform Security Services for Developers
- The Oracle Platform Security Services APIs
- Common Uses for Oracle Platform Security Services
- Using OPSS with Oracle Application Development Framework
- Using the Oracle Security Developer Tools
- Using OPSS Outside Oracle JDeveloper/Oracle ADF

14.1 About Oracle Platform Security Services for Developers

This section explains the benefits of securing applications with Oracle Platform Security Services and introduces the major components of the tool-set. It contains these topics:

- The Development Cycle
- Challenges of Securing Java Applications
- Meeting the Challenges with Oracle Platform Security Services
- OPSS Architecture

14.1.1 The Development Cycle

JavaEE software development is based on a develop-deploy-manage cycle. The Oracle Platform Security Services security implementation plays an important part in all phases of the cycle.

The following list summarizes the JavaEE development cycle, emphasizes the tasks specific to developing secure applications, and highlights the security enhancements that OPSS provides.

1. The developer creates Web components, enterprise beans, servlets, and application clients based on business requirements.

While the developer has access to a declarative approach, additional value is obtained when using Oracle ADF, which makes use of OPSS APIs.

2. The developer defines JavaEE logical roles and assigns them privileges through security constraints, all through configuration in standard JavaEE deployment descriptors.

3. The components are assembled and combined into an Enterprise Archive (EAR) file.

   As part of this process, the assembler specifies options appropriate to the environment.

4. The assembler defines application-level security constraints and resolves potential conflicts between module-level configurations.

5. The EAR file is deployed to Oracle WebLogic Server.

   As part of the deployment process, the deployer may map JavaEE roles to deployment users and roles.

6. The system administrator maintains and manages the deployed application.

   This task includes creating and managing roles and users in the deployment environment as required by the application customers.

For finer-grained code-based or subject-based access control using Java 2 or JAAS features, the traditional steps include:

1. The developer identifies any resources that may be accessed and must be protected as appropriate.

2. The developer defines permissions to protect these resources.

3. The developer implements code for runtime authorization checks.

4. The system administrator maintains any necessary policy configuration to enforce the desired permissions. Policy provisioning should be completed prior to runtime.

Oracle ADF and OPSS provide these enhancements:

■ At Design Time - modeling of application roles, defining resources as permissions, and assigning permissions to roles. Application credential management is supported, for example, ADF connections can store credentials in the Credential Store Framework during design time.

■ At Deployment Time - policy and credential migration options are available

■ Post-deployment, the administrator performs essential tasks such as mapping application roles to enterprise users or groups which are reflected at run-time

14.1.2 Challenges of Securing Java Applications

Java developers face some challenges in developing secure applications:

■ The JavaEE standard does not define any API for fine-grained authorization, credential mapping, role mapping, auditing, or integration with single-sign.

■ Developers need to acquire in-depth security knowledge at the expense of focusing on application business logic.

■ There is no consistent security experience across platforms. For example, custom security solutions often develop their own security framework, which is often not portable across platforms.

- Custom solutions for securing JavaEE applications often lack support for large enterprise security deployments.

  Such key aspects as manageability, availability, scalability, and reliability are often missing from custom solutions.

14.1.3 Meeting the Challenges with Oracle Platform Security Services

Oracle Platform Security Services (OPSS) is a portable security services abstraction layer that provides a robust security framework and saves development time and effort. OPSS enhances traditional JavaEE development in many respects:

- provides basic security services such as authentication, authorization, auditing, role management, and credential management.

- allows developers to focus on the application logic

- provides you the same services that Oracle Fusion Middleware products get:
  - OPSS is the security platform for Fusion applications and Oracle Fusion Middleware components, including Oracle WebLogic Server, Oracle Entitlement Server, Oracle SOA Suite, and Oracle WebCenter among others.

    > **Note:** This is just a sampling of the products that rely on OPSS.

- is standards-based and enterprise-ready:
  - stress-tested to support enterprise deployments.
  - interoperable across different LDAP servers and single sign-on (SSO) systems.
  - certified on Oracle WebLogic Server

- provides the same set of APIs for all types of applications (in-house, third-party, Oracle Fusion)

- Optimizes development time with its use of abstraction layers

- Application maintenance is simplified since security rules can be changed without affecting application code

- enables legacy and third-party security provider integration

OPSS support for Identity Management (IdM) includes:

- a lightweight infrastructure that allows customers to build and deploy small to mid-size applications

- a plug-in interface to IdM systems:
  - Applications build against OPSS can be plugged to a centrally deployed Identity Management system
  - Customers can scale their applications to switch to a centrally deployed Identity Management system
  - No code changes are required in the application when switching between IdM systems.

14.1.4 OPSS Architecture

Figure 14–1 shows the basic components of the OPSS architecture. There are specific APIs for most of the features discussed earlier in this manual that are available for use

by application developers. Underlying SPIs (service provider interfaces), mentioned briefly in Section 2.2, "OPSS Architecture Overview," are mostly invisible to application developers and administrators.

Figure 14–1 OPSS Architecture
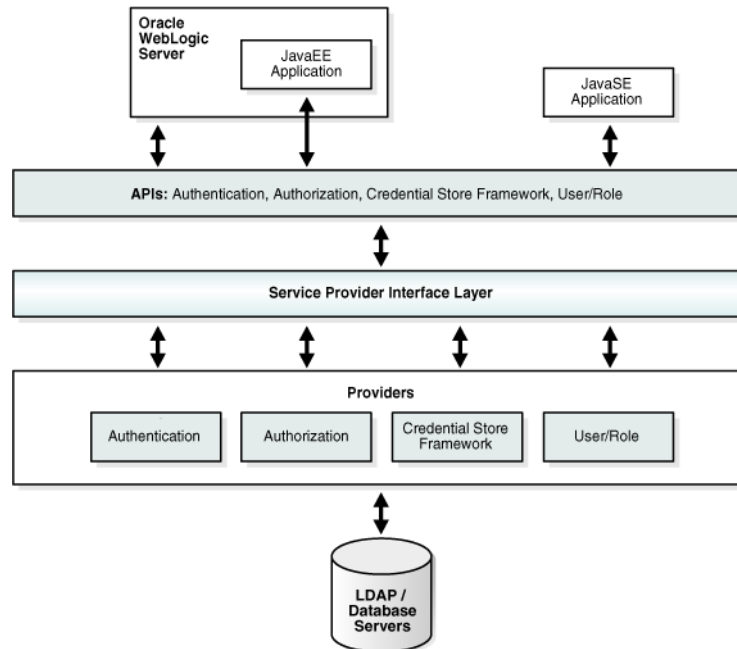
The Oracle Platform Security architecture has the following characteristics:

- A layered architecture that decouples the application layer from the underlying implementation.

- An extensible architecture that provides explicit extensibility points (through the SPI layer) where custom implementations (such as custom login modules) can be plugged into the framework to provide special functionality.

14.2 The Oracle Platform Security Services APIs

This section describes the APIs available to developers working with Oracle Platform Security Services:

- The LoginService API

- The User and Role API

- JAAS Authorization and the JpsAuth.checkPermission API

- The Credential Store Framework API

14.2.1 The LoginService API

OPSS provides the LoginService authentication API to enable JavaSE applications to obtain information from an identity store and to manage the contents of the store.

Support for authentication is through the login module, a component that authenticates users and populates a subject with principals. This process occurs in two distinct phases:

- In the first phase, the login module attempts to authenticate a user by means of credentials supplied by the user.

- In the second phase, the login module assigns relevant principals to a subject, which is eventually used to perform a privileged action.

For details, see Chapter 16, "Developing Authentication".

14.2.2 The User and Role API

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 underlying identity store could be an LDAP directory server such as Oracle Internet Directory, 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.

The API supports:

- LDAP directory servers such as Oracle Internet Directory

- flat files

- other custom repositories such as databases, by implementing a custom provider for the repository

With the User and Role API, you can:

- access repositories programmatically in a portable way,

- eliminate the need to account for the intricacies of particular identity sources

- enable your application to work seamlessly against different repositories

- switch between various identity repositories without any code changes to your application

For details, see Chapter 19, "Developing with the User and Role API".

14.2.3 JAAS Authorization and the JpsAuth.checkPermission API

The JavaEE authorization model uses role membership to control access to EJBs and web resources that are referenced by URLs; the Java 2 authorization model uses permissions (instead of role memberships) to control access decisions.

You can specify authorization policies in application code. Sensitive lines of code are preceded with calls to check 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 a security exception.

Java 2 authorization is based on permissions, rather than roles, and access control decisions are evaluated by calls to the `SecurityManager` or the

`AccessController`. When used with JAAS, this model allows for a programmatic authorization capability, thus providing fine-grained control to resources.

Oracle Fusion Middleware supports authorization using JavaEE DD/annotation based authorization and JAAS/Java2 permission based authorization. Both declarative and programmatic approaches for enforcing authorization policies are supported; the latter is implemented through the `JpsAuth.checkPermission` API, and AccessController.checkPermission can be used as well.

Using OPSS APIs provides the following benefits beyond the traditional authorization models:

- OPSS extends JAAS with the ability to use application roles that are assigned permissions.

- OPSS provides policy management support, which is lacking in the standard JAAS model. See Section 18.3.4.2, "Managing Policies" for an example.

- Using the JpsAuth.checkPermission OPSS API as opposed to the standard checkPermission API provides additional benefits such as more robust debugging and integrated audit support.

For details about authorization features of OPSS, see Chapter 18, "Developing Authorization".

14.2.4 The Credential Store Framework API

A credential store a secure, central store for credentials and collections of credentials. Multiple applications can use the same credential stor.e

The Credential Store Framework (CSF) API provides the mechanism by which applications access the credential store.

The CSF API supports file-based (Oracle wallet) and LDAP-based credential stores.

Critical functions provided by the CSF API include returning credentials for a given map name, assigning credentials to and deleting credentials from a given map name, and other operations related to credential maps and keys.

Operations on CredentialStore are secured by CredentialAccessPermission, which implements the fine-grained access control model utilized by CSF.

For details about the API, see Chapter 17, "Developing with the Credential Store Framework".

14.3 Common Uses for Oracle Platform Security Services

The same set of OPSS APIs can be used by both JavaEE and JavaSE developers. Topics in this section illustrate common applications for the APIs, and demonstrate differences between JavaEE and JavaSE implementations.

14.3.1 A JavaEE Application using OPSS APIs

This example shows a traditional JavaEE application enhanced with OPSS security APIs.

Figure 14–2   JavaEE Application using Multiple OPSS APIs
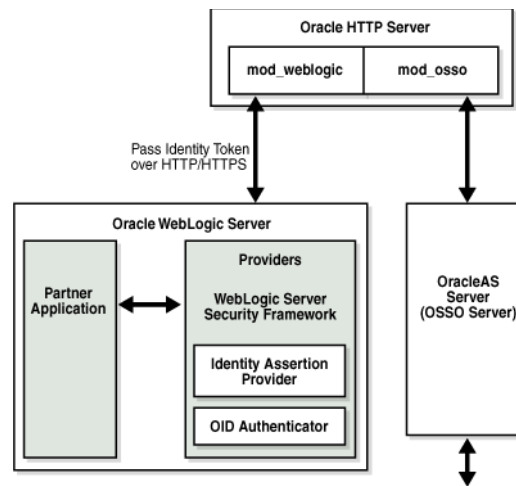
Key features include:

- Integration with Oracle WebLogic Server

- Credential Store Framework API to secure credentials in the LDAP directory or file-based credential store. Different types of credentials will be stored here - external database credentials, external Web Service credentials, and so on.

- User and Role API to query attributes stored in the identity store

- JpsAuth.checkPermission API for authorization

14.3.2 Authentication with OPSS APIs

Developers have these choices for implementing authentication:

- declarative authentication, where you configure authentication in web.xml (this is standard JavaEE security)

- programmatic security. Oracle Fusion Middleware provides several APIs, including:

  – Oracle WebLogic Server's authentication API, `weblogic.security.auth.Authenticate`

  – OPSS' `oracle.security.jps.service.login.LoginService` API for JavaSE applications. This API supports user/password authentication and username assertion. The assertion functionality is protected by `JpsPermission` with the name `IdentityAssertion`.

This example shows a JavaEE application that must assert an identity through a token or through user credentials.

Figure 14–3   Programmatic Authentication
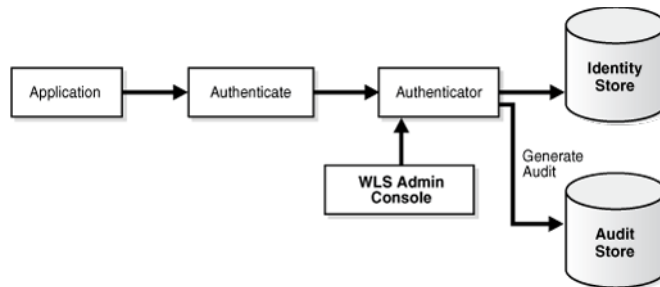
Key features include:

- username and password supplied by the application for programmatic authentication with the `Authenticate` API

- uses a WebLogic authenticator

- identity assertion through a token (authentication without a password)

- assertion protected by a code source permission. Only applications that have been granted the code source permission (codebase permission grant `oracle.security.jps.JpsPermission` with name `IdentityAssertion"` nd action `execute`) can use this API for identity assertion.

> **See Also:**
>
> - *Oracle Fusion Middleware Understanding Security for Oracle WebLogic Server*
>
> - Chapter 16, "Developing Authentication".

14.3.3 Programmatic Authorization

This example shows a JavaEE application that uses portable, fine-grained authorization.

Figure 14–4   Fine-grained Authorization
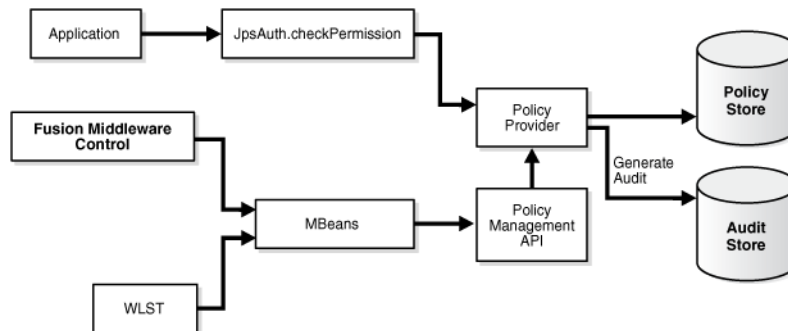
Key features include:

- authorization through `JpsAuth.checkPermission` API calls

- auditing of authorization decisions

14.3.4 Credential Store Framework

This example shows an application that needs to access and store credentials for an external system such as a database.

Figure 14–5   Storing External Passwords in Credential Store Framework
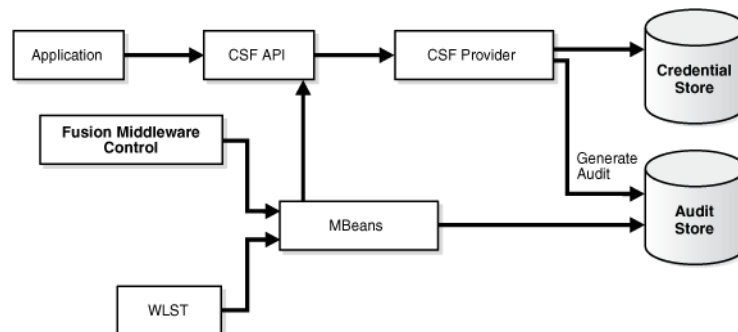
Key features include:

- credentials (username and password, symmetric keys, and so on) stored securely in credential store

- support for LDAP-based credential stores in addition to Oracle Fusion Middleware's out-of-the-box, file-based credential store called Oracle wallet.

- credentials can be managed with either Oracle Enterprise Manager Fusion Middleware Control or `WLST` command-line tool

- credential store operations can be audited

14.3.5 User and Role

This example shows an application that needs to search for users in an identity store. Various queries are needed; for example, searching all users in "APAC" or locating all emails for users in a given role.

Figure 14–6   Searching the Identity Store with User and Role API
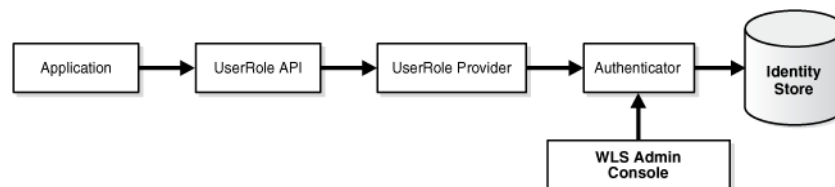
Key features include:

- calling the User and Role API to access user attributes

- the same APIs work on user attributes in the default authenticator or an external LDAP store.

  The User and Role API is automatically configured based on the configuration in the authentication provider, either default or any other LDAP based authentication.

- same API regardless of where the attributes are stored

14.3.6 Oracle ADF Authorization

For an example of authorization using Oracle ADF, see Section 14.4.2, "How Oracle ADF Uses OPSS".

14.3.7 JavaSE Application

This example shows a JavaSE Swing application using different OPSS APIs.

Figure 14–7   JavaSE Application using OPSS APIs
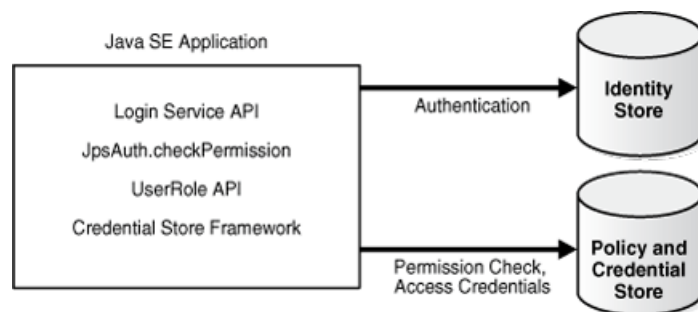
Note: In an LDAP-based store, like that shown in the figure, both policies and credentials are maintained in the same store, while file-based stores maintain separate files for each.

Key features include:

- LoginService API for authentication

- JpsAuth.CheckPermission for authorization

- User and Role API to query attributes stored in LDAP or other back-end

- use of credential store to secure credentials

14.4 Using OPSS with Oracle Application Development Framework

When you use Oracle ADF to develop and deploy applications, you are able to directly leverage the security features of OPSS, since Oracle ADF is integrated with OPSS.

This section introduces Oracle ADF and provides an example of OPSS security in an Oracle ADF application.

14.4.1 About Oracle ADF

The Oracle Application Development Framework (Oracle ADF) is an end-to-end application framework that builds on Java Platform, Enterprise Edition (Java EE) standards and open-source technologies to simplify and accelerate implementing service-oriented applications. For enterprise solutions that search, display, create, modify, and validate data using web, wireless, desktop, or web services interfaces, Oracle ADF can simplify the development effort.

Used in tandem, Oracle JDeveloper 11g and Oracle ADF give you an environment that covers the full development life cycle from design to deployment, with drag-and-drop data binding, visual UI design, and team development features built in.

14.4.2 How Oracle ADF Uses OPSS

The Oracle ADF Security framework is the preferred technology to provide authentication and authorization services to the Fusion web application. Among the advantages:

- Oracle ADF Security is built on top of the Oracle Platform Security Services (OPSS) architecture, which provides a critical security framework and is itself well-integrated with Oracle WebLogic Server.

- Oracle JDeveloper and Oracle ADF use the OPSS application life cycle listener framework to migrate credential and policy data when the application is deployed.

> **See Also:** Chapter 7, "Deploying Secure Applications"

Oracle ADF's built-in support for security features including OPSS features helps reduce some of the effort that would be required to implement those features outside Oracle ADF; indeed, certain features are not available using only container-managed security.

This example shows an Oracle ADF application that needs to use both fine-grained authorization and JavaEE container-based authentication.

Figure 14–8   Oracle ADF using JpsAuth.checkPermission
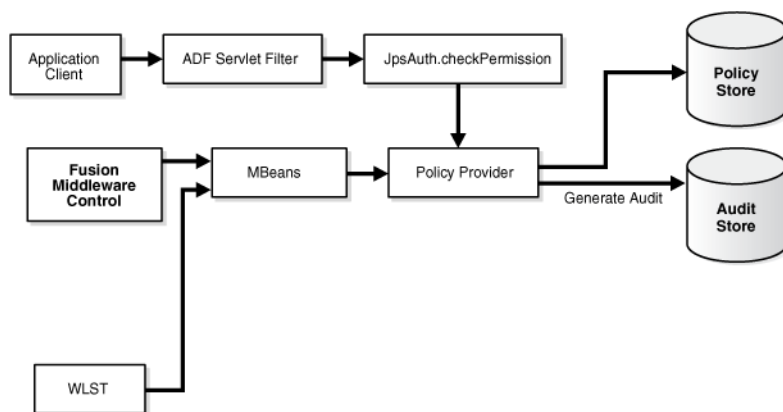
Key features include:

- use of JDeveloper's security wizard to create required security configuration
- calls by Oracle ADF filter to `JpsAuth.checkPermission`
- Task flows and regions protected using custom Oracle ADF permissions

For more information, see:

- ADF Security in the *Oracle Fusion Middleware Security Overview*
- *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*

14.4.3 The Oracle ADF Development Life Cycle

This example show how an application is first deployed to integrated Oracle WebLogic Server (Oracle WebLogic Server embedded in Oracle JDeveloper). A

developer then produces an EAR file that is deployed, through Oracle Enterprise Manager Fusion Middleware Control, to another Oracle WebLogic Server domain.

This Oracle WebLogic Server domain is likely to be located in a test or staging area.

Figure 14–9   Oracle ADF Application Deployed to Oracle WebLogic Server
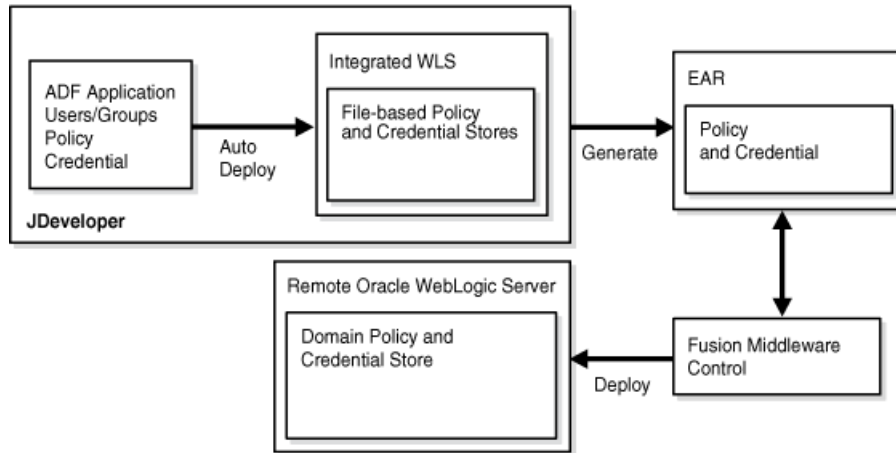
Key features include:

- Oracle ADF application developed with Oracle JDeveloper
- uses Oracle ADF security wizard and Oracle ADF authorization policy editor
- Oracle JDeveloper provides an integrated user experience, migrating artifacts to the run-time environment:
  - Users and groups defined at design-time are available in the default authenticator
  - authorization policy and credential data is migrated using the OPSS listener framework.
- application developer creates EAR file containing policy and credentials
- administrator deploys the EAR to a remote Oracle WebLogic Server using Fusion Middleware Control or `WLST`

> **Note:**   For more information about deployment tools and options, see Chapter 7, "Deploying Secure Applications".

14.5 Using the Oracle Security Developer Tools

Oracle Security Developer Tools provide you with the cryptographic building blocks necessary for developing robust security applications, ranging from basic tasks like secure messaging to more complex projects such as securely implementing a service-oriented architecture. The tools build upon the core foundations of cryptography, public key infrastructure, web services security, and federated identity management, and are widely used in building Oracle's own security offerings.

For more information about the tools, see:

- Oracle Security Developer Toolkit in the *Oracle Fusion Middleware Security Overview*
- *Oracle Fusion Middleware Reference for Oracle Security Developer Tools*

14.6 Using OPSS Outside Oracle JDeveloper/Oracle ADF

You can make use of OPSS APIs in your applications if you are using a development IDE other than Oracle JDeveloper and Oracle ADF.

However, in that case, you will need to perform manual configuration in OPSS configuration files and `web.xml`, so you do not get the benefits of automatic configuration and security migration that are available when using Oracle JDeveloper.

For more information about this topic, see Chapter 15, "Manually Configuring JavaEE Applications to Use OPSS".

15

Manually Configuring JavaEE Applications to Use OPSS

This chapter describes the manual configuration and packaging recommended for J2EE 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 J2EE 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 JavaEE Application Manually
- Configuring a JavaEE Application 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)

Before using the information in this chapter, it is recommended that you be familiar with the concepts introduced in Chapter 14, "Overview of Developing Secure Applications with Oracle Platform Security Services."

15.1 Configuring the Servlet Filter and the EJB Interceptor

> **Note:** Oracle JDeveloper automatically inserts the required servlet filter (`JpsFilter`) and EJB interceptor (`JpsInterceptor`) configurations for Oracle ADF applications.
>
> You enter the configurations explained in this section manually *only* if you are packaging or configuring a J2EE application that uses OPSS outside that environment.

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.

Both allow the configuration of 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 domain 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; 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 wants to use.

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 false.

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 true. 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 18.3.1, "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>
```

15.1.1 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.

15.1.2 Summary of Filter and Interceptor Parameters

The following table summarizes the description of the parameters used by the JpsFilter and the JpsInterceptor:

Table 15–1 Summary of JpsFilter and JpsInterceptor Parameters

Parameter Name	Values	Default	Function	Notes
application.name	Any valid string	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 domain 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	FALSE	To enable or disable the anonymous Subject.	Set to TRUE to create to allow the creation of a Subject with the anonymous user and the anonymous role.
remove.anonymous.role	TRUE or FALSE	TRUE	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.
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.	

15.2 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 I.6, "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>
```

15.3 Packaging a JavaEE Application Manually

This section explains the packaging requirements for a servlet or an EJB (using custom policies and credentials) in a WAR file that is to be deployed with the Oracle WebLogic Administration Console.

Application policies are defined in the file `jazn-data.xml`. The *only* supported way to package this file with an application is 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 constrains 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 J2EE 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 domain 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.

15.3.1 Packaging Policies with Application

Application policies are defined in the file `jazn-data.xml`. The *only* supported way to package this file with an application is 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 the 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.

15.3.2 Packaging Credentials with Application

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 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 the `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.

15.4 Configuring a JavaEE Application to Use OPSS

This section describes several configurations that a developer would perform manually for a J2EE 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 the Command migrateSecurityStore
- Example of Configuration File jps-config.xml

15.4.1 Parameters Controlling Policy Migration

The migration of policies is controlled by several properties that configure two listeners that control the migration behavior.

All parameters are configured in the file `META-INF/weblogic-application.xml`.

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 7.2.1, "Deploying JavaEE 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 WLST
> command `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.

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.

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.

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>
```

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 must always 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 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.

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>
```

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 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.

15.4.2 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.

15.4.2.1 To Skip Migrating All Policies

The following matrix shows the settings that prevent the migration from taking place:

Table 15–2   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.

15.4.2.2 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 15–3   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.

15.4.2.3 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 15–4   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.

15.4.2.4 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 15–5   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 15–6   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 code-based 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.

15.4.2.5 To Migrate Policies in a Static Deployment
Table 15–7 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 15–7   Settings to Migrate Policies with Static Deployments

JpsApplicationLifecycleListener	Set
jps.policystore.migration	MERGE or OVERWRITE

Table 15–8 shows the setting that skip the migration of application policies when the application is statically deployed.

Table 15–8   Settings Not to Migrate Policies with Static Deployments

JpsApplicationLifecycleListener	Set
jps.policystore.migration	OFF

15.4.2.6 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 domain 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 WLST command 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 domain policy file `$DOMAIN_HOME/config/fmwconfig/system-jazn-data.xml`.

15.4.3 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*.

15.4.4 Parameters Controlling Credential Migration

The migration of credentials is defined by a parameter that configures the listener that controls the migration behavior. This parameter is configured in the file `META-INF/weblogic-application.xml`.

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.

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>
```

If set, this parameter's value must be one of the following: MERGE, OVERWRITE, or OFF. The OVERWRITE value is available *only* 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 specified as illustrated in the following fragment:

```
<wls:listener>
  <wls:listener-class>
     oracle.security.jps.wls.listeners.JpsApplicationLifecycleListener
  </wls:listener-class>
</wls:listener>
```

15.4.5 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 it life as being packaged with an application, but when the application is undeployed credentials are *not* removed.

15.4.5.1 To Skip Migrating Credentials

The following matrix shows the setting that prevents the migration from taking place:

Table 15–9   Settings to Skip Credential Migration

	Valid at deploy or redeploy
jps.credstore.migration	OFF

15.4.5.2 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 15–10   Settings to Migrate Credentials with Merging

	Valid at deploy or redeploy
JpsApplicationLifecycleListener	Set
jps.credstore.migration	MERGE

15.4.5.3 To Migrate Credentials with Overwriting

This operation is valid only when the WebLogic server is running in development mode. The following matrix shows the setting that migrates all credentials overwriting matching target credentials:

Table 15–11   Settings to Migrate Credentials with Overwriting

	Valid at deploy or redeploy
JpsApplicationLifecycleListener	Set
jps.credstore.migration	OVERWRITE
jps.app.credential.overwrite.allowed	This system property must be set to TRUE

15.4.6 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:/path/foo.jar</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 classpath, so the policy provider can load them when a service instance is initialized.

15.4.6.1 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
```

15.4.6.2 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
```

15.4.6.3 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
```

15.4.7 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 bootstrap credentials with an WLST command, see Section 9.5.2.5, "modifyBootStrapCredential."

15.4.8 Migrating Identities with the Command migrateSecurityStore

Identity data can be migrated manually from a source repository to a target LDAP repository using the WLST command `migrateSecurityStore`. The command 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 XML-based identity store.

For example, this command 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 command 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.

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

> **Important:** Before invoking a security-related WLST command in a shell, you must run the script `wlst.sh`, as illustrated in the following sample:
>
> ```
> > sh $ORACLE_HOME/common/bin/wlst.sh
> ```
>
> This ensures that the required JARs are added to the classpath. Failure to run the above script in a new shell renders the WLST commands unusable.
>
> Before running an online command, connect to the server as follows:
>
> ```
> >java weblogic.WLST
> >connect('servername', 'password', 'localhost:portnum')
> ```

Script and Interactive Modes Syntaxes

To migrate identities, 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 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 command 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 can be XML-based, LDAP-based backed by an Oracle Internet Directory server, or the WebLogic embedded LDAP identity store. No other destination is supported.

- `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 command 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.

15.4.9 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 J2EE and J2SE 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>
```

16

Developing Authentication

This chapter applies only to JavaSE applications. For authentication for JavaEE applications, it provides links to other Oracle WebLogic documentation.

This chapter is divided in the following sections:

- Links to Authentication Topics for JavaEE Applications
- Developing Authentication for JavaSE Applications

Prior to using this information, it is strongly recommended to be familiar with the context in which these APIs are used. For details, see Section 14.3.2, "Authentication with OPSS APIs."

16.1 Links to Authentication Topics for JavaEE Applications

The following documents are a good source of information for developing authentication in JavaEE 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 JavaEE 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 JavaEE 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."

16.2 Developing Authentication for JavaSE Applications

This section is divided into the following sections:

■ The Identity Store

■ Configuring an LDAP Identity Store in JavaSE Applications

■ Supported Login Modules for JavaSE Applications

■ Using the OPSS API LoginService in JavaSE Applications

16.2.1 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 *identity store* is the abstract concept that refers to the storage where identity data is kept, and authentication providers are ways to access an identity store.

The way an application obtains information from an OPSS security store (identity, policy, or credential store) and manages its contents is using the OPSS APIs, as illustrated in the following graphic:
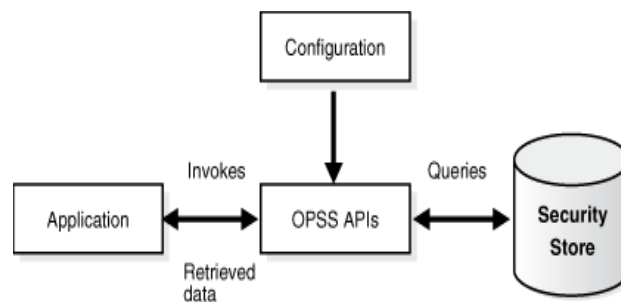
16.2.2 Configuring an LDAP Identity Store in JavaSE Applications

A JavaSE 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>
</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."

16.2.3 Supported Login Modules for JavaSE 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.

16.2.3.1 The Identity Store Login Module

A JavaSE 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 JavaSE 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 user 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 JavaSE 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)
```

16.2.3.2 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:

```java
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);
 }
         }
      }
   }
}
```

16.2.3.3 Using the Identity Login Module for Assertion

To use the Identity Store login module for assertion, a developer must:

- Provide an IdentityAssertion permission with execute action, so the permission `oracle.security.jps.JpsPermission` can execute the login module for assertion.

- Implement a callback handler that extends the class
  `oracle.security.jps.callback.IdentityCallback`.

The above two requirements are illustrated in the following code and configuration samples.

Provisioning the JpsPermission

The following configuration sample illustrates a grant allowing the code `MyApp` the required `JpsPermission` to execute the assertion login module:

```
<grant>
  <grantee>
    <codesource>
      <url>file:${domain.home}/servers/MyApp/-</url>
    </codesource>
  </grantee>
  <permissions>
    <permission>
      <class>oracle.security.jps.JpsPermission</class>
      <name>IdentityAssertion</name>
      <actions>execute</actions>
    </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>
      <actions>execute</actions>
    </permission>
  </permissions>
</grant>
```

Implementing the CallbackHandler

The following code fragment illustrates the implementation of a handler and the method handle, and how to instantiate it:

```
import oracle.security.jps.callback.IdentityCallback;
import javax.security.auth.Subject;
import javax.security.auth.login.LoginContext;

class MyAssertionCallbackHandler extends CallbackHandler {
        private String name = null;

        public MyAssertionCallbackHandler(String name) {
          this.name = name;
        }
```

```
        public void handle(Callback[] callbacks) throws IOException,
UnsupportedCallbackException {
            for (Callback callback : callbacks) {
                if (callback instanceof IdentityCallback) {
                    IdentityCallback idcb = (IdentityCallback) callback;
                    idcb.setIdentity(name);
                } else {
                    //throw exception
                }
            }
        }
}

CallbackHandler cbh = new  MyAssertionCallbackHandler("name");
Subject sub = new Subject();

// The first parameter must be the name of the jpsContext in which the
// IdStore Login Module is referrenced
LoginContext context = new LoginContext("myContext", sub,  cbh);

context.login();
Subject s = context.getSubject();
...
```

16.2.4 Using the OPSS API LoginService in JavaSE Applications

To invoke a login module programmatically in JavaSE 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>
```

17

Developing with the Credential Store Framework

This chapter describes how to work with the APIs for the Credential Store Framework (CSF).

In an earlier chapter, we explained how applications can use CSF to securely store the credentials for external systems (Web sites, databases, and so on) in a credential store; and the additional benefits of CSF, such as the ability to audit credential store operations and use common Oracle Fusion Middleware user interfaces.

Prior to using the information in this chapter to work with the CSF APIs, it is strongly recommended that you familiarize yourself with the context in which these APIs are used. For details, see:

- Chapter 14, "Overview of Developing Secure Applications with Oracle Platform Security Services"

- Section 14.2.4, "The Credential Store Framework API", which shows a common usage scenario

This chapter contains 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

17.1 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 9, "Configuring the Credential Store"

17.2 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 domain 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 8.4, "Managing the Domain 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 9.4, "Migrating Credentials to the Domain Credential 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.

17.3 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.

17.3.1 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.**

17.3.2 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>AppName.jar</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.

17.3.3 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>AppName.jar</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>
</jazn-policy>
```

17.4 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 -> PasswordCredential4
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.

17.5 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 JavaSE

For details, see Chapter 9, "Configuring the Credential Store".

17.6 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

17.6.1 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 2.5.3, "Scenario 3: Securing a JavaSE 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 17.3, "Setting the Java Security Policy Permissions".

Examples

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

17.6.2 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 17.3, "Setting the Java Security Policy Permissions".

3. Start Oracle WebLogic Server.

4. Deploy and test the application.

17.7 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 JavaSE or JavaEE 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: JavaSE Application with Wallet Store

- Example 2: JavaEE Application with Wallet Store

- Example 3: JavaEE Application with LDAP Store

17.7.1 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";
                        }
                });
            }
        }
```

17.7.2 Example 1: JavaSE Application with Wallet Store

This example shows a sample JavaSE 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 JavaSE 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 domain 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 JavaSE 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 17.7.1, "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 JavaSE 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 instead of the default JDK provider
    static {
        java.security.Policy.setPolicy(new JavaPolicyProvider());
    }

    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 JavaEE environment for a JRF install for a supported application server, the OPSS policy provider will have been initialized.

17.7.3 Example 2: JavaEE Application with Wallet Store

This example shows a sample JavaEE 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:${domain.home}/servers/${weblogic.Name}
         /tmp/_WL_user/myTestApp/-</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 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 JavaSE 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 JavaSE Environment

In the JavaSE environment, the following calls are equivalent:

```
CredentialStore store =
JpsServiceLocator.getServiceLocator().lookup(CredentialStore.class);
```

and:

```
CredentialStore store =
```

```
JpsContextFactory.getContextFactory().getContext().getServiceInstance(CredentialSt
ore.class);
```

The latter call is shown in Section 17.7.2, "Example 1: JavaSE Application with Wallet Store".

17.7.4 Example 3: JavaEE Application with LDAP Store

This example uses the same JavaEE 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.

17.8 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.

18

Developing Authorization

This chapter explains how to develop and configure custom authorization in applications in the following sections:

- Authorization Overview
- Authorization for EJBs and Servlets (JavaEE Model)
- Authorization Using Permissions (JAAS/OPSS Model)
- The Class oracle.security.jps.ResourcePermission

Prior to using the information in this chapter, it is strongly recommended that you be familiar with the context in which it is used. For details, see Section 14.3.3, "Programmatic Authorization."

18.1 Authorization Overview

This section compares and contrasts the authorization available in the JavaEE and the JAAS models.

18.1.1 The JavaEE Authorization Model

The JavaEE authorization model uses role membership to control access to EJB methods and web resources that are referenced by URLs; the JAAS authorization model uses permissions (instead of role memberships) to control access decisions.

In the JavaEE 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 specified 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 18–1 shows the advantages and disadvantages of each approach.

Table 18–1 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).

Table 18–1   (Cont.) Comparing Authorization in the Java EE Model

Authorization Type	Advantages	Disadvantages
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.

18.1.2 The JAAS Authorization Model

Java 2 authorization is based on permissions, rather than roles, and access control decisions are evaluated by calls to the SecurityManager or the AccessController. When used in combination with the JAAS model, the model allows a programmatic authorization approaches, thus providing fine-grained control to resources.

In this model, an authorization policy specifies the following information:

- Application and enterprise groups.

- Members of a role, which can be other roles or users.

- 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 is granted the appropriate permissions to access the code. If the user has the appropriate permissions, the code is run. Otherwise, the code throws and exception.

For details about Java 2 standard permissions, see
http://java.sun.com/javase/6/docs/technotes/guides/security/permissions.html.

For details about a permission class available out-of-the-box, see The Class oracle.security.jps.ResourcePermission.

18.2 Authorization for EJBs and Servlets (JavaEE Model)

The JavaEE authorization model is based on roles and managed by the container; policies assign permissions to users and roles, and they are enforced by the container to protect resources.

A container can provide authorization to applications running in it in two ways: declaratively and programmatically. These topics and an example is found in the following sections:

- Declarative Authorization

- Programmatic Authorization

- JavaEE Code Example

18.2.1 Declarative Authorization

This authorization approach allows the 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 JavaEE 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*.

18.2.2 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).

18.2.3 JavaEE Code Example

This 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>");
        }
}
```

18.3 Authorization Using Permissions (JAAS/OPSS Model)

JAAS/OPSS authorization is based on controlling the operations that a class can perform when it is loaded into a running environment.

This section is divided into the following sections:

- Using the Method checkPermission
- Debugging and Auditing Support
- Using the Methods doAs and doAsPrivileged
- JAAS/OPSS Code Examples

18.3.1 Using the Method checkPermission

Oracle Fusion Middleware supports the use of the method `checkPermission` in the following standard classes:

- `java.lang.SecurityManager`
- `java.security.AccessController`

In addition, Oracle Fusion Middleware also supports the use of the method `checkPermission` in the following class:

- `oracle.security.jps.util.JpsAuth`

Oracle recommends the use of `checkPermission` in the class `JpsAuth` (instead of the previous two) 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.

Here is an example illustrating the use of this method:

```
//create permission
FilePermission perm = new FilePermission("/home/developer/foo.txt","read");
//check permission
AccessController.checkPermission(perm);
```

The method `checkPermission` behaves according to the value of the JAAS mode (see JAAS mode in Chapter 15.1, "Configuring the Servlet Filter and the EJB Interceptor"), as listed in the following table:

Table 18–2   Behavior of checkPermission According to JAAS Mode

JAAS Mode Setting	checkPermission
off or undefined	Enforces code-based security based on the security policy in effect, and there is no provision for subject-based security.
doAs	Enforces a combination of code-based and subject-based security according to the new access control context created through the `doAs` block.
doAsPrivileged	Enforces a combination of code-based and subject-based security, but is uses a null access control context. This setting implies subject-based security only.
subjectOnly	Takes into consideration grants involving principals *only* (and it disregards those involving codebase) when evaluating a permission.

18.3.2 Debugging and Auditing Support

The class `oracle.security.jps.util.JpsAuth` provides for auditing support, so applications can audit permission check failures and successes.

In addition, the following system property settings:

```
-Djps.auth.debug.enable=true
-Djps.auth.debug.enable.verbose =true
```

provide information on which principals or codesources failed a permission check, and details on those that passed is provided even when the container is not started with a security manager.

18.3.3 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.

18.3.4 JAAS/OPSS Code Examples

This section presents code examples that illustrate permission checking and policy management.

18.3.4.1 Checking Permissions

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 includes the following policies:

```
<policy-store>
 <applications>
  <application>
   <!-- application name or application context -->
    <name>PolicyServlet</name>
     <app-roles>
      <app-role>
       <name>developer</name>
       <display-name>application role developer</display-name>
       <class>oracle.security.jps.service.policystore.ApplicationRole</class>
       <members>
        <member>
        <class>weblogic.security.principal.WLSUserImpl</class>
        <name>weblogic</name>
       </member>
      </members>
     </app-role>
    </app-roles>
   <!-- application specific jaas policy -->
   <jazn-policy>
     <grant>
      <grantee>
       <principals>
        <principal>
         <class>oracle.security.jps.service.policystore.ApplicationRole</class>
         <name>developer</name>
        </principal>
       </principals>
      </grantee>
      <permissions>
       <permission>
        <class>java.lang.RuntimePermission</class>
        <name>getClassLoader</name>
       </permission>
      </permissions>
     </grant>
   </jazn-policy>
  </application>
 </applications>
</policy-store>
```

web.xml

The filter `JpsFilter` is configured as follows:

```
<web-app>
 <display-name>PolicyTest: PolicyServlet</display-name>
 <filter>
  <filter-name>JpsWlsFilter</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>JpsWlsFilter</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();
        }
    }
}
```

18.3.4.2 Managing Policies

The following code example illustrates the management of policies:

```
import oracle.security.jps.JpsContext;
import oracle.security.jps.JpsContextFactory;
import oracle.security.jps.service.policystore.ApplicationPolicy;
import oracle.security.jps.service.policystore.PolicyStore;
import oracle.security.jps.service.policystore.info.AppRoleEntry;

import javax.security.auth.AuthPermission;
```

```
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.AccessController;
import java.security.Permission;
import java.security.Principal;
import java.util.Date;
import java.util.List;

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>");
        try {
            JpsContextFactory ctxFact = JpsContextFactory.getContextFactory();
            JpsContext ctx = ctxFact.getContext();
            final PolicyStore policyStore =
ctx.getServiceInstance(PolicyStore.class);
            ApplicationPolicy policyServletPolicy
                    = policyStore.getApplicationPolicy("PolicyServlet");

 //get application role developer
            List<AppRoleEntry> developerAppRoleList =
policyServletPolicy.searchAppRoles("developer");
            AppRoleEntry approleEntry = developerAppRoleList.get(0);
            Principal pr = approleEntry.getPrincipal();

//grant permissions to this role
            Principal[] principals = new Principal[]{pr};
```

```
                AuthPermission doAsPerm = new AuthPermission("authPerm");
                final Permission[] perms = new Permission[]{doAsPerm};
                out.println("<br> Attempting to grant principal " + pr + " Permissions
in application Policy ");
                policyServletPolicy.grant(principals, null, perms);
                out.println("<br> Granted permissions");
                Subject subject = new Subject();
                subject.getPrincipals().add(pr);
                out.println("<br> principal has permissions ?" +
policyServletPolicy.hasPermission(subject, doAsPerm));
          } catch(Exception e) {
                e.printStackTrace();
                printException("Exception in Policy Management " , 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();
          }
    }
}
```

18.4 The Class oracle.security.jps.ResourcePermission

oracle.security.jps.ResourcePermission is a class provided by OPSS that
can be used as the class of a permission within any grant to protect application or
system resources. Application developers no longer need to write permission classes,
since ResourcePermission is already available out-of-the-box and can be used with
XML- or LDAP-based policy providers.

A permission that uses the class ResourcePermission defines the resource type, the
resource name, and a list of actions as illustrated in the following XML sample:

<permission>

```
  <class>oracle.security.jps.ResourcePermission</class>
  <name>resourceType=type,resourceName=name</name>
  <actions>comma-separated-list-of-actions</actions>
</permission>
```

Two such permissions with a given resource type and name need not have the same comma-separated list of actions:

```
<permission>
  <class>oracle.security.jps.ResourcePermission</class>
  <name>resourceType=myFileType,resourceName=myName</name>
  <actions>view,create</actions>
</permission>

<permission>
  <class>oracle.security.jps.ResourcePermission</class>
  <name>resourceType=myFileType,resourceName=myName</name>
  <actions>view,delete,modify</actions>
</permission>
```

19

Developing with the User and Role API

This chapter describes how to work with the APIs for user and role management. Prior to using this information, it is strongly recommended that you familiarize yourself with the context in which these APIs are used. For details, see:

- Chapter 14, "Overview of Developing Secure Applications with Oracle Platform Security Services"

- Section 14.2.2, "The User and Role API", which shows a common usage scenario

It contains the 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

19.1 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 Sun Java System Directory Server (from Sun Microsystems), 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 the context of an Oracle WebLogic Server container, the principal class should be cast to `weblogic.security.principal.WLSUserImpl`.

> **Note:** The following must be true when invoking the User and Role API:
>
> - The domain policy store must be LDAP-based
> - The domain administration server must be up and running

A Note about Using the User and Role APIs

As a general rule of thumb, authentication should only be performed by authentication providers, not through the User and Role APIs.

Additionally, it is recommended that authentication providers be configured with the connect DN of a user that does not have write privileges.

19.1.1 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.

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.

19.2 Summary of Roles and Classes

Table 19–1 lists the classes and interfaces of the User and Role API.

Table 19–1   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).

Table 19–1   (Cont.)   Classes and Interfaces in the User and Role API

Name	Type	Description
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 thus resulting in capability to form really complex nested search filter.
		See the Javadoc (Section 19.9, "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 `getIdentityStoreInstance(Hashtable)` on this object.
IdentityStoreFactoryBuilder	Class	This class builds the identity store factory.
IMException	Class	This exception is the super class 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 19.9, "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 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.
OperationNotSupportedException	Class	This exception is thrown by an service provider if it doesn't support an operation. For example this can be thrown by the service provider, in IdentityStore.getUserManager() call, if it doesn't provide support for UserManager.
PasswordPolicyException	Class	This class extends IMException (see above).
Property	Class	Property contains name-value information.

Table 19–1   (Cont.)   Classes and Interfaces in the User and Role API

Name	Type	Description
PropertySet	Class	A collection of property name and value pairs. Property class is used to represent the property name and value(s) pair. PropertySet 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 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 underlying identity repository while searching and matching results will be filtered out. See the Javadoc (Section 19.9, "The User and Role API Reference") for an example of a simple search filter.
StoreConfiguration	Interface	StoreConfiguration holds the configuration properties for a given IdentityStore instance. The behavior of this IdentityStore 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 Subject. 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.
UserProfile	Interface	This interface represents the detailed profile of a user. It allows for user properties to be accessed in a generic manner. You can read or modify any property of user with these APIs: ■ getProperty(java.lang.String) ■ getProperties(java.lang.String[]) ■ setProperty(oracle.security.idm.ModProperty) ■ setProperties(oracle.security.idm.ModProperty[])

19.3 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

19.3.1 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.

19.3.2 Setting Up the Environment

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.

19.3.3 Selecting the Provider

Oracle Platform Security Services support a range of user repositories, including the following LDAP directories:

- Microsoft Active Directory
- Novell EDirectory
- Sun Java System Directory Server Enterprise Edition
- Oracle Internet Directory
- Oracle Virtual Directory

- OpenLDAP

- Oracle WebLogic Server Embedded LDAP Directory

Your choice of identity store will dictate the provider class you utilize for the provider. The provider class must implement the interface specified by the User and Role API framework. Table 19–2 shows the available provider classes:

Table 19–2 LDAP Identity Provider Classes

Provider	Factory Name
Microsoft Active Directory	oracle.security.idm.providers.ad.ADIdentityStoreFactory
Novell EDirectory	oracle.security.idm.providers.edir.EDIdentityStoreFactory
Sun Java System Directory Server	oracle.security.idm.providers.iplanet.IPIdentityStoreFactory
Oracle Internet Directory	oracle.security.idm.providers.oid.OIDIdentityStoreFactory
Oracle Virtual Directory	oracle.security.idm.providers.ovd.OVDIdentityStoreFactory
OpenLDAP	oracle.security.idm.providers.openldap.OLdapIdentityStoreFactory
Oracle WebLogic Server Embedded LDAP Directory	oracle.security.idm.providers.wlsldap.WLSLDAPIdentityStoreFactory

19.3.4 Creating the Provider Instance

Once the provider's class name is identified, you 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 the store reference, which is the actual handle to the identity store, can be obtained:

```
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

19.3.5 Properties for Provider Configuration

Configuration is dependent on the user/role 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 repository runs

- the user and password to use in accessing the repository

For a list of LDAP-based providers, see Section 19.3.3, "Selecting the Provider".

19.3.5.1 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 19–3 lists the start-time configuration properties:

Table 19–3   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 user name 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).
ST_SECURITY_ CREDENTIALS	The credentials necessary to log in to the identity repository.
ST_LDAP_URL	The URL of the identity repository.

Table 19–3 (Cont.) Start-time Identity Provider Configuration Properties

Property Name	Description
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 that you set at runtime affect all subsequent operations executed by the provider and control the behavior of the IdentityStore instance of the provider.

You configure runtime properties by specifying the appropriate parameters and values on 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 19–3 lists the run-time configuration properties:

Table 19–4 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	
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
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

Table 19–4   (Cont.)   Runtime Identity Provider Configuration Properties

Property Name	Description
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

19.3.5.2 When to Pass Configuration Values

You can specify configuration data:

- when creating a factory instance

  See Also: Section 19.3.6, "Configuring the Provider when Creating a Factory Instance"

- when creating a store instance

  See Also: Section 19.3.7, "Configuring the Provider when Creating a Store Instance"

- at runtime, through a store configuration object

  See Also: Section 19.3.8, "Runtime Configuration"

19.3.6 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

19.3.6.1 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.

19.3.6.2 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`.

19.3.6.3 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
    (IPIdentityStoreFactory.ST_SECURITY_PRINCIPAL, "<User DN>");
factEnv.put
```

```
   (IPIdentityStoreFactory.ST_SECURITY_CREDENTIALS, "<User password>");
factEnv.put(IPIdentityStoreFactory.ST_LDAP_URL,
   "ldap://ldaphost:port/");

Map m = new Hashtable();
m.put(RoleProfile.OWNER, "myowner");

factEnv.put
   (IPIdentityStoreFactory.ST_PROPERTY_ATTRIBUTE_MAPPING, m);

ipFactory = builder.getIdentityStoreFactory(
   "oracle.security.idm.providers.iplanet.IPIdentityStoreFactory",
   factEnv);
```

19.3.6.4 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) >= 10
```

Here is an example:

```
factEnv.put
   (LDIdentityStoreFactory.ST_CONNECTION_POOL_MIN_CONNECTIONS,  "3");

factEnv.put
   (LDIdentityStoreFactory.ST_CONNECTION_POOL_MAX_CONNECTIONS, "16");
```

19.3.6.5 Configuring a Custom Connection Pool Class

This section explains how to provide 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.stdldap.JNDIPool");
```

19.3.7 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 19.3.6, "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.

19.3.8 Runtime Configuration

Earlier, in Section 19.3.6, "Configuring the Provider when Creating a Factory Instance" and Section 19.3.7, "Configuring the Provider when Creating a Store Instance", we demonstrated how to perform configuration during instance creation. 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");
```

19.3.9 Programming Considerations

This section contains tips for working with providers and provider artifacts.

19.3.9.1 Provider Portability Considerations

To ensure that your application is portable when switching providers (say, from fusiondb provider to Oracle Internet Directory provider or the converse), follow these guidelines when working with the User and Role APIs:

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.getWildCardCharacter()` API. This will fetch the correct wild character based upon the underlying provider. For example, the API will return `%` for the FusionDB provider and return `*` for the Oracle Internet Directory (OID) provider.

```
SmpleSearchFilter 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 = .................

SmpleSearchFilter 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.

19.3.9.2 Considerations when Using IdentityStore Objects

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 DAS, 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 User and Role APIs. 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.

19.3.10 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 would 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.

19.4 Searching the Repository

The User and Role API provides two types of query functions:

- APIs that return a single identity

- APIs that return a list of identities

This section describes searches and related tasks you can accomplish with the APIs, 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

19.4.1 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 APIs 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. These API's are minimal in that:

- they accept only string values

- they do not support regular expressions

These functions raise an exception if multiple entities having the same value exist in the store.

19.4.2 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.

19.4.3 Specifying Search Parameters

The SearchParams Object

The SearchParams object contains the following information:

- Search Filter - this is discussed below

- 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 that is 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

19.4.4 Using Search Filters

The User and Role APIs include different types of search filters to facilitate a variety of search operations. This section t explains key facts about the use of search filters:

- Operators in Search Filters

- Handling Special Characters when Using Search Filters

- Examples of Using Search Filters

19.4.4.1 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 make usage simpler, 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 the converse).

The NOT operator is accessible using the following SearchFilter API:

- void negate();

- boolean isNegated();

19.4.4.2 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.

19.4.4.3 Examples of Using Search Filters

Several usage examples are presented.

Example 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 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 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 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 5: Complete Search with Output

This code 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());
}
```

19.4.5 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 sf1 = oidStore.getSimpleSearchFilter(
                 UserProfile.GUID,
                 SimpleSearchFilter.TYPE_EQUAL, guid);
              SearchParameters params = new SearchParameters();
              params.setFilter(sf1);
              SearchResponse resp = oidStore.search(params);
              while (resp.hasNext())
                 System.out.println("user for guid : " + guid + ","+ resp.next());
```

19.5 User Authentication

For verification purposes, you can use the User and Role APIs for password-based authentication of users.

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");
```

19.6 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

19.6.1 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.

19.6.2 Creating an Identity

Two APIs 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`.

19.6.3 Modifying an Identity

In order 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.setProperties(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.

19.6.4 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);
```

19.7 Examples of User and Role API Usage

This section contains two examples illustrating practical applications of the User and Role APIs:

- 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

19.7.1 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.

19.7.2 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 usrname = "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();
        }
      }
    }
```

19.7.3 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 usrname = "amyd";
            // delete user if already exists
            try
            {
              User usr = adStore.searchUser(usrname);
              usrmanager.dropUser(usr);
            }catch(IMException ime){}

            System.out.println("creating user "+usrname);
            char[] password = {'w', 'e', 'l', 'c', 'o', 'm','e','3'};
            User usr = usrmanager.createUser(usrname, 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();
    }
  }
}
```

19.8 SSL Configuration for LDAP-based User and Role API Providers

This section describes SSL support for the User and RoleAPIs. It contains these topics:

- Out-of-the-box Support for SSL
- Customizing SSL Support for the User and Role API

19.8.1 Out-of-the-box Support for SSL

LDAP-based providers for User and Role APIs 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 providers are:

- Microsoft Active Directory
- Novell eDirectory
- Sun Java System Directory Server
- Oracle Internet Directory

- OpenLDAP

- Oracle WebLogic Server Embedded LDAP Directory

19.8.1.1 System Properties

To support SSL you need to 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.

19.8.1.2 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");
   ```

19.8.2 Customizing SSL Support for the User and Role API

You can customize SSL support by providing a customized `SSLSocketFactory` to the User and Role API provider.

19.8.2.1 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");
```

19.9 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

20

Developing with Oracle HTTPClient Security

Oracle HTTPClient security supports the Secure Sockets Layer (SSL) on client HTTP connections with the package `HTTPClient`.

This chapter describes the basic features of `HTTPClient`, how to configure it with system properties, and how to use it with the standard Java Secure Socket Extension (JSSE) or with NTLM authentication.

This chapter is divided in the following sections:

- Overview of Oracle HTTPClient Security
- Oracle HTTPClient Security Features
- JSSE System Properties
- Using HTTPClient with JSSE
- SSL Host Name Verification
- Using NTLM Authentication with Oracle HTTPClient

For general information about JSSE, see
`http://java.sun.com/javase/6/docs/technotes/guides/security/jsse/JSSERefGuide.html`.

20.1 Overview of Oracle HTTPClient Security

Oracle HTTPClient security implements the package `HTTPClient.HTTPConnection` and provides a comprehensive set of functions that allows the creation of HTTP connections (secured or otherwise) between a source and a target. The Oracle implementation of the class `HTTPClient` diverges from the original open source version upon which it is based, and even though many similarities between them remain, the two implementations are not necessarily compatible with each other.

For further details about JSSE and `java.net`, see information in the following links:

- `http://java.sun.com/javase/technologies/security`
- `http://java.sun.com/javase/6/docs/api/`

20.2 Oracle HTTPClient Security Features

Oracle HTTPClient provides the following major features for security-aware applications:

- Selection of cipher suite and of multiple cryptographic algorithms

- Support for JKS and PKCS12 (Oracle Wallet) keystores. Starting with this release, however, the JKS keystore is the preferred keystore for use with HTTPClient. Any Oracle Wallet usage with HTTPClient should be switched over to JKS usage.

- HTTPS tunneling through proxies

- HTTP proxy authentication

- (Limited) Support of the `java.net.URL` framework

Oracle HTTPClient supports the use of the protocols HTTP 1.0 and HTTP 1.1 for communication between client and server, and the JSSE SSL implementation. Note that without adding OraclePKIProvider to the JVM, Oracle Wallets cannot be used. See Prerequisites for using JSSE.

The following four sections describe in detail the major features supported by Oracle HTTPClient security.

20.2.1 Keystore Formats

Starting with this release, an application should use only a JKS keystore with Oracle HTTPClient. The JKS keystore is the default keystore in Java and no JVM settings are required to use this keystore. Applications that want to migrate the content of their existing Oracle Wallets to JKS can use `orapki` command.

The choice of data format in a keystore depends on the choice of the security provider configured in the JVM. If you choose the default Sun Microsystems security provider, then you must use a JKS-formatted keystore. If you choose the `oracle.security.pki.OraclePKIProvider`, which should be registered with the JVM, then you must use the keystore formats PKCS12 or SSO (auto-login) Oracle wallet.

PKCS12 and JKS are standard keystore formats; SSO Oracle wallet is an Oracle proprietary format.

Wallets formatted with PKCS12 or SSO contain all their credentials encrypted; the main difference between these two keystores is that, in case of an SSO wallet, a password to open the wallet to access information in it is not required.

For details about using the `orapki` command to migrate an Oracle Wallet to JKS, see section H.1.6 Converting Between Oracle Wallet and JSK Store in *Oracle Fusion Middleware Administrator's Guide*.

For further details about Oracle Wallets, see chapter Oracle Wallet Manager and orapki in *Oracle Fusion Middleware Administrator's Guide*.

20.2.2 SSL Connection Information

Applications can access information about an established SSL connection using the method `getSSLSession` in the class `HTTPConnection` in the Oracle package `HTTPClient`. After a connection has been established, applications can retrieve connection information such as the cipher suite used for the connection and the peer certificate chain.

20.2.3 Support for java.net.URL

A user of the `java.net.URL` framework may configure JSSE according to the system properties listed in the following document (JSSE Reference Guide): http://java.sun.com/javase/6/docs/technotes/guides/security/jsse/JSSERefGuide.html

20.2.4 Cipher Suites

A *cipher* is an algorithm that performs encryption and decryption, and a *cipher suite* is a set of such algorithms. Before data can flow through an SSL connection, both sides of the connection must negotiate and agree on the various connection parameters, including the common cipher to use for data transmission. Selecting a particular cipher and related SSL connection parameters allows the communicating participants to establish the appropriate security level on the transmission channel.

The following choice of parameters of a secure transmission are recommended:

- Key exchange - choose the public-key cryptographic algorithm RSA rather than the cryptographic protocol Diffie-Hellman.

- Symmetric cipher - choose Triple DES or RC4 128 rather than any other encryption method, because they use strong keys.

- Hash function - choose SHA1 digest rather than MD5, because SHA1 produces a stronger digest.

When null encryption is chosen, SSL performs only authentication and data integrity checks. The system `property https.cipherSuites` specifies the subset of available cipher suites (comma-delimited list of cipher suite names).

20.3 JSSE System Properties

This section describes some standard JSSE system keystore properties used to set security credential information.

javax.net.ssl.keyStore

This property indicates the location and name of the keystore or wallet file to use as the keystore.

javax.net.ssl.keyStorePassword

This property indicates the password necessary to open the keystore (keystore or wallet file). Specifying the keystore password in Java system property may pose a security risk in some environments.

To avoid this risk, if a password is necessary, do not store it in a clear text. Alternatively, use the method `System.setProperty` to set it dynamically (before starting the `HTTPConnection`) and then unset it after the handshake is completed. Note, however, that this still may present the security risk since another application running in the server may read the password by polling this property.

javax.net.ssl.keyStoreType

This property indicates the type of file used for the keystore. When using the OraclePKIProvider, the value can be PKCS12 or SSO; when using the default Sun Microsystems JSSE implementation, the value must be JKS.

javax.net.ssl.trustStore

This property, similar to `javax.net.ssl.keyStore`, indicates the location and name of the keystore or wallet file to use as the truststore.

javax.net.ssl.trustStorePassword

This property, similar to `javax.net.ssl.keyStorePassword`, indicates the password necessary to open the truststore file (keystore or wallet).

javax.net.ssl.trustStoreType

This property, identical in function to the property `javax.net.ssl.keyStoreType`, indicates the type of file used for the truststore.

20.4 Using HTTPClient with JSSE

This section describes the requirements for using and configuring Oracle HTTPClient with JSSE. The information applies only to scenarios where Oracle HTTPClient is used with Oracle Wallet.

By default, HTTPClient uses the JSSE implementations available in the JVM.

20.4.1 Prerequisites for using JSSE

To use `HTTPClient` with JSSE:

- Use Sun Microsystems JDK version 1.2 or higher (JSSE is included as part of JDK 1.4 or higher).

- If Oracle wallet support is required, include the file `oraclepki.jar` in your classpath. Note that the files `jssl-1_1.jar` and `jssl-1_2.jar` (used by Oracle Java SSL) are no longer required.

  > **Note:** Starting with this release, an application should use only a JKS keystore with Oracle HTTPClient. The JKS keystore is the default keystore in Java and no JVM settings are required to use this keystore. Application wanting to migrate the content of their existing oracle Wallets to JKS can use `orapki` command.

- This step is necessary only if you are using Oracle Wallet. Register the OraclePKIProvider with the JVM, by using either of the following steps:

  - Edit the file `jre/lib/security/java.security`, by adding an entry similar to `security.provider.n=oracle.security.pki.OraclePKIProvider`, where `n` is replaced with the appropriate index number.

  - Call the method `Security.addProvider(new OraclePKIProvider())`

20.4.2 Configuring HTTPClient

> **Note:** By default, `HTTPClient` uses the JVM JSSE `SSLSocketFactory`, which recognizes the SSL-related system properties. An application can, however, construct its own JSSE `SSLSocketFactory` for HTTPClient to use. This custom `SSLSocketFactory` could be designed to support multiple keystores or truststores and to ignore SSL-related system properties.
>
> Also, by default, `HTTPClient` uses the JVM JSSE implementation. If, however, the Sun SSL provider is configured, the cacerts file is used as a truststore based on the rules defined in http://java.sun.com/javase/6/docs/technotes/guides/security/jsse/JSSERefGuide.html.

To configure `HTTPClient` to use JSSE as the underlying SSL provider, proceed as follows:

1. This step is necessary only if not using the JDK infrastructure.

   Create a truststore using the Sun Microsystems utility `keytool`. For details about this tool usage, see `http://java.sun.com/javase/6/docs/technotes/tools/index.html#security`.

   Assign the truststore to the underlying SSL provider. For the Sun JSSE provider, set the system properties `javax.net.ssl.trustStore`, `javax.net.ssl.trustStoreType`, and `javax.net.ssl.trustStorePassword`. Oracle discourages setting passwords in system properties, and some truststore types may not require password for read access.

2. Create an instance of `HTTPClient.HTTPConnection`, as illustrated in the following code sample:

   ```
   HTTPConnection conn = new HTTPConnection( "https", "my.bank.com", -1 );
   ```

3. Optionally, call `connect` to establish the SSL connection and get SSL session info, as illustrated in the following code sample:

   ```
   conn.connect();
   SSLSession sessionInfo = conn.getSSLSession();
   ```

20.5 SSL Host Name Verification

Although SSL verifies that the certificate chain presented by the server is valid and contains at least one certificate trusted by the client, that does not prevent impersonation by malicious third parties. An HTTPS standard that addresses this risk requires that HTTPS servers have certificates issued to their host name. Then, it is the responsibility of the client to perform the validation after the SSL connection is established.

Oracle HTTP Client Security uses a host name verifier (an implementation of the interface `javax.net.ssl.HostnameVerifier`) to attest that the host name in an SSL certificate matches the host name in the URI used to access the protected server. `HTTPClient` invokes the `HostnameVerifier` instance immediately after establishing an SSL session and throws the exception `javax.net.ssl.SSLPeerUnverifiedException` if it detects a host name mismatch.

> **Important:** You can use your own implementation of the host name verifier or the one provided by Oracle (as described below). Any such implementation *must* contain a constructor with no arguments.

For further details about the class `HostnameVerifier`, see `http://java.sun.com/j2se/1.4.2/docs/api/javax/net/ssl/HostnameVerifier.html`

Host name verification is performed *only if* it is enabled by a system property or programmatically, as described in the following two sections.

20.5.1 Enabling Host Name Verification with a System Property

To enable host name verification without having to alter your code, set the system property `HTTPClient.defaultHostnameVerifier` to the full name of the class implementing the interface `javax.net.ssl.HostnameVerifier`.

20.5.2 Enabling Host Name Verification Programmatically

To enable host name verification programmatically, use:

- ```
  static HostnameVerifier setDefaultHostnameVerifier
  (HostnameVerifier myHNVerifier)
  ```

  This method sets the host name verifier for all connections in the entire Java VM to the passed instance. It returns the host name verifier previously set as default (if there was one) or null (if the host name verification was disabled for the Java VM).

- ```
  HostnameVerifier setHostnameVerifier
  (HostnameVerifier myHNVerifier)
  ```

  This method sets the verifier for the connection to the passed instance and overrides any default setting; if the passed argument is null, it disables the host name verification for the connection. It returns the host name verifier previously set for the connection (if there was one) or null (if the host name verification was disabled for the connection).

20.5.3 Oracle Standard Host Name Verifier

Oracle HTTP Client Security includes the class `StandardHostnameVerifier`, an implementation of the host name verifier that provides standard host name matching rules for site identity checking.

`StandardHostnameVerifier` compares (case-insensitive) the SSL session host name and the common name (CN) of the distinguished name (DN) from the first certificate in the SSL certificate chain; moreover, it allows the use of wildcard characters while attempting to match both names during the comparison. For example, if wildcard is enabled, the strings "*.oracle.com" and "www.oracle.com" match.

`StandardHostnameVerifier` exposes the following methods:

- ```
  boolean setRecognizeWildcardCNs(boolean recognizeWildcardCNs)
  ```

  This method sets the comparison to recognize or not to recognize wildcard characters; it returns the value previously set.

- ```
  boolean isRecognizeWildcardCNs()
  ```

  This method returns true is the comparison uses wildcards, or false otherwise.

- ```
  boolean verify(java.lang.String hostname,
  javax.net.ssl.SSLSession sslSession)
  ```

  This method returns true, if the passed hostname matches the authentication scheme of the server in the passed session, or false otherwise.

The following line illustrates how to set the Oracle host name verifier as the default host name verifier with a system property:

```
HTTPClient.defaultHostnameVerifier=HTTPClient.StandardHostnameVerifier;
```

20.5.4 Additional Verification

To perform further validation (beyond host name verification) by using the data in the object returned by the method `getSSLSession`, proceed as follows.

First, establish a connection to the server without transferring any data:

```
httpsConnection.connect();
```

Next, obtain a session info:

```
SSLSession sessionInfo = httpsConnection.getSSLSession();
```

The object `SSLSession` exposes many attributes of the SSL connection which may be used to verify the connection. For details, see
http://java.sun.com/j2se/1.5.0/docs/api/javax/net/ssl/SSLSession.html

20.6 Using NTLM Authentication with Oracle HTTPClient

NT LAN Manager (NTLM) is a Microsoft authentication protocol supported by Oracle HTTPClient that clients, proxies, and servers can use.

NTLM is a connection-oriented protocol because after a connection is authenticated no further credentials are required for transmission while the connection remains open. Thus, a client using NTLM proves its identity to establish the connection and need not send its password thereafter.

Proxy servers may also use NTLM for client authentication, but, unlike some request-oriented authentication schemes (such as Basic and Digest authentication), an NTLM client authenticates its connection with only the proxy server, not the resource server.

HTTPClient also supports Basic and Digest Authentication. This support equals that provided in the open source version of HTTPClient.

20.6.1 NTLM Domain Name and Realm

An NTLM account identifier has the format `NTDname\userName`, where the optional string `NTDname` identifies the NT Domain name; if not present, the domain name is assumed to be the default NT Domain name, which is set with the system property `HTTPClient.ntlm.defaultDomainName`. Furthermore, if the account identifier contains no NT Domain name and there is no default set, then the NTLM-protected server uses its own NT Domain name.

A realm, as specified in some authentication schemes (such as Basic authentication), does not apply to NTLM since the challenge does not have a realm directive; thus, all NTLM credentials are assumed to be part of the same empty realm within an HTTPClient connection.

20.6.2 Connecting to NTLM-Protected Servers

To connect to an NTLM-protected resource or proxy server, credentials must be added to the HTTPClient AuthorizationInfo credential store. This is accomplished in one of two ways according to the type of server to which the client is connecting. In any case, `HTTPClient` automatically queries the credential store when challenged by an NTLM server.

To connect to an NTLM-protected resource server, add NTLM credentials using the HTTPConnection instance as illustrated in the following code snippet:

```
HTTPConnection conn = new HTTPConnection( myHost, myPort );
conn.addNtlmAuthentication( myUsername, myPassword );
HTTPResponse response = conn.Get( "/index.htm" );
int status = response.getStatusCode();
assertEquals( 200, status );
```

Note that the example above adds NTLM credentials for the resource server represented in myHost.

Alternatively, add NTLM credentials using the object AuthorizationInfo directly as illustrated in the following code snippet:

```
HTTPConnection conn = new HTTPConnection( myHost, myPort );
conn.setCurrentProxy( myProxyHost, myProxyPort );
AuthorizationInfo.addNtlmAuthentication(myProxyHost, myProxyPort, myUsername,
myPassword, conn.getContext() );
HTTPResponse response = conn.Get( "/index.htm" );
int status = response.getStatusCode();
assertEquals( 200, status );
```

Note that the example above adds NTLM credentials for the proxy server represented in myProxyHost.

To connect to an NTLM-protected proxy server, add NTLM credentials using `HTTPConnection.addNtlmAuthentication` directly, as illustrated above, since the method `addNtlmAuthentication` adds credentials for the associated resource server, not the proxy server.

Part V

Appendices

This part contains the following appendices:

- Appendix A, "OPSS Configuration File Reference"
- Appendix B, "File-Based Identity and Policy Store Reference"
- Appendix F, "OPSS System and Configuration Properties"
- Appendix C, "Oracle Fusion Middleware Audit Framework Reference"
- Appendix D, "User and Role API Reference"
- Appendix H, "References"
- Appendix E, "Administration with WLST Scripting and MBean Programming"
- Appendix G, "Upgrading Security Data"
- Appendix I, "Troubleshooting Security in Oracle Fusion Middleware"

A

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 JavaEE applications) or `jps-config-jse.xml` (for JavaSE applications) and is located in the directory `$DOMAIN_HOME/config/fmwconfig`.

For JavaSE 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 keystore, login modules, and the audit service. For a complete example of a configuration file see Section 15.4.9, "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
`<jpsConfig>` {1}	Defines the top-level element in the configuration file.
`<property>` {0 or more}	Defines names and values of properties. It can also appear elsewhere in the hierarchy, such as under the elements `<propertySet>`, `<serviceProvider>`, and `<serviceInstance>`.

`<propertySets>` {0 or 1}
    `<propertySet>` {1 or more}
        `<property>` {1 or more}
	Groups one or more `<propertySet>` elements so that they can referenced as a group.

`<extendedProperty>` {0 or more}
    `<name>` {1}
    `<values>` {1}
        `<value>` {1 or more}
	Defines a property that has multiple values. It can also appear elsewhere in the hierarchy, such as under the elements extendedProperty and serviceInstance.

`<extendedPropertySets>` {0 or 1}
    `<extendedPropertySet>` {1 or more}
        `<extendedProperty>` {1 or more}
            `<name>` {1}
            `<values>` {1}
                `<value>` {1 or more}
	Groups one or more `<extendedPropertySet>` elements so that they can referenced a group.

`<serviceProviders>` {0 or 1}
    `<serviceProvider>` {1 or more}
        `<description>` {0 or 1}
        `<property>` {0 or more}
	Groups one or more `<serviceProvider>` 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.

`<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}
	Groups one or more `<serviceInstance>` elements, each of which configures and specifies property values for a service provider defined in a `<serviceProvider>` element.

`<jpsContexts>` {1}
    `<jpsContext>` {1 or more}
        `<serviceInstanceRef>` {1 or more}
	Groups one or more `<jpsContext>` 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 DNs 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 DNs 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}
```

<extendedProperty>

```
<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>

<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>

<extendedPropertySets>

This is the parent element for the <extendedPropertySet> element.

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>

<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 `<jspContext>` 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 is the parent element for the <jpsContext> element.

Attributes

Name	Description
default	Specifies the context that is used by an application if none is specified. The default value of the <jpsContexts> element must match the name of a <jpsContext> 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>

<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}
```

<property>

```
<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>

<propertySets>

This is the parent element for the <propertySet> element.

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>

<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 jps-config.xml file.
	Values: string
	Default: n/a (required)
provider	Indicates which service provider this is an instance of.
	The provider 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}
```

<serviceInstance>

```
                         <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 an XML-file-based identity store service. For a file-based identity store, the subscriber name is the default realm. The example sets the lo cation 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>

<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>

<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
type	Specifies the type of service provider being declared; it must be either of the following:
	CREDENTIAL_STORE
	IDENTITY_STORE
	POLICY_STORE
	AUDIT
	LOGIN
	ANONYMOUS
	KEY_STORE
	IDM (for pluggable identity management)
	CUSTOM
	The implementation class more specifically defines the type of provider, such as by implementing an XML-file-based identity store or LDAP-based policy store, for example.
	Values: string (a value above)
	Default: n/a (required)
name	Designates a name for this service provider. This name is referenced in the `provider` attribute of `<serviceInstance>` elements to create instances of this provider. No two `<serviceProvider>` elements may have the same `name` attribute setting within a configuration file.
	Values: string
	Default: n/a (required)

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 is the parent element of all <serviceProvider> elements.

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.

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 JavaSE 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
`<jazn-data>`	This is the top-level element in the `system-jazn-data.xml` file.

Table B–1   (Cont.) Hierarchy of Elements in system-jazn-data.xml

Hierarchy	Description
```<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}```	The `<jazn-realm>` section specifies security realms, and the users and enterprise groups (as opposed to application-level roles) included in each realm.

Table B–1   (Cont.) Hierarchy of Elements in system-jazn-data.xml

Hierarchy	Description
| ```
<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}
            <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}
``` | The `<policy-store>` section configures application-level policies. You can define roles at the application level, and members in the roles. Members can be users or roles.

When `<jazn-policy>` is specified under the `<application>` element, it specifies policies at the application level.

`<jazn-policy>` can also appear under the `<jazn-data>` element, in which case it specifies policies at the system level. |

Table B–1   (Cont.) Hierarchy of Elements in system-jazn-data.xml

Hierarchy	Description

```<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}```
	When the ```<jazn-policy>``` element is located under the ```<jazn-data>``` element, it specifies policies at the system-level.
	
	```<jazn-policy>``` can also appear under the ```<application>``` element, in which case it specifies policies at the application level.

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>
- <app-role>
- <app-roles>
- <application>
- <applications>
- <attribute>
- <class>
- <codesource>
- <credentials>
- <description>
- <display-name>
- <extended-attributes>
- <grant>
- <grantee>
- <guid>

- \<jazn-data\>
- \<jazn-policy\>
- \<jazn-realm\>
- \<member\>
- \<members\>
- \<name\>
- \<owner\>
- \<owners\>
- \<permission\>
- \<permissions\>
- \<policy-store\>
- \<principal\>
- \<principals\>
- \<realm\>
- \<role\>
- \<roles\>
- \<type\>
- \<uniquename\>
- \<url\>
- \<user\>
- \<users\>
- \<value\>
- \<values\>

<actions>

This element specifies actions that are permitted with respect to the associated permission class and name. Values are case-sensitive and are specific to each Permission implementation. Examples of actions include "invoke" and "read,write".

Parent Element

<permission>

Child Element

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.

<app-role>

<app-role>

This element configures roles for an application.

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.

- <uniquename> specifies a unique name to reference the application role. This is for internal use only.

Parent Element

<app-roles>

Child Element

<class>,<description>,<display-name>,<guid>,<members>,<name>,
<uniquename>

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}
                    <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}
```

<app-role>

```
<class> {1}
<uniquename> {0 or 1}
<guid> {0 or 1}
```

Examples

See `<policy-store>` for examples.

<app-roles>

<app-roles>

This is the parent element for `<app-role>` elements, which configure roles for an application.

Parent Element

`<application>`

Child Element

`<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 configures 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 Element

<app-roles>,<description>,,<jazn-policy>,<name>

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>

<applications>

This is the parent element for `<application>` elements, which configure roles and policies for an application.

Parent Element

`<policy-store>`

Child Element

`<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 additional attribute to associate with the application role.

Parent Element

<extended-attributes>

Child Element

<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}
                    <uniquename> {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 takes a string value and has several uses, depending on its location in the 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 Element

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}
                            <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}
                    ...
        <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>

<codesource>

This element specifies a codesource URL to which permissions are being granted as part of a policy configuration.

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.

Parent Element

<grantee>

Child Element

<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 contains the authentication password for a user. The credentials are in obfuscated form by default.

Parent Element

<user>

Child Element

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>

<description>

This element contains a text string to provide more information about an item. An item can be an application role, application policy, permission grant, security role, or user, depending on the parent element.

Parent Element

<app-role>, <application>, <grant>, <role>, or <user>

Child Element

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>

<display-name>

This element specifies an item's name that can be used for display by a GUI tool. An item can be an application role, user, or enterprise group, depending on the parent element.

Parent Element

<app-role>, <role>, or <user>

Child Element

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>

<extended-attributes>

This element specifies additional attributes associated with the application role.

Parent Element

<app-role>

Child Element

<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}
                    <members> {0 or 1}
                        <member> {1 or more}
                            <name> {1}
                            <class> {1}
                            <uniquename> {0 or 1}
                            <guid> {0 or 1}
```

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>

<grant>

This element contains elements that assign a set of permissions to a grantee (a codesource, or a set of principals, or both) as part of a policy configuration.

Parent Element

<jazn-policy>

Child Element

<description>, <grantee>, <permissions>

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 is used in conjunction with a parallel <permissions> element to specify who or what the permissions are granted to: a set of principals, a codesource, or both.

This is used for policy configuration.

Parent Element

<grant>

Child Element

<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. 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 identity 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 Element

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}
                    <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}

        <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}
              ...
```

Example

See `<jazn-realm>` for examples.

<jazn-data>

<jazn-data>

This is the top-level element in the `system-jazn-data.xml` file-based identity and policy store.

Attributes

Name	Description
schema-major-version	Specifies the major version number of the `system-jazn-data.xml` XSD. The value of this attribute is fixed at `11` for use with Oracle Fusion Middleware 11*g*.
schema-minor-version	Specifies the minor version number of the `system-jazn-data.xml` 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 Element

<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>

<jazn-policy>

This element configures 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 Element

<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>
```

<jazn-policy>

```
                                    </principal>
                                </principals>
                                <codesource>
                                    <url>http://www.oracle.com/policyOnly</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>

<jazn-realm>

This element specifies security realms, and the users and enterprise groups (as opposed to application-level roles) they include.

This 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 Element

<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>
```

<jazn-realm>

```
                                            <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>
```

```
                              <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>
```

<member>

This element can be found under a <role> or <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 <uniquename> and <guid>, which specify a unique name and unique global identifier; these optional subelements are for internal use only.

Parent Element

<members>

Child Element

- 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>, <uniquename>, <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}
        <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}
```

Example

See `<jazn-realm>` and `<policy-store>` for examples.

<members>

<members>

This is the parent element for <member> elements.

Parent Element

<role>,<app-role>

Child Element

<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}
                    <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}
```

Example

See `<jazn-realm>` and `<policy-store>` for examples.

<name>

<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>
  ```

Parent Element

<app-role>, <application>, <attribute>, <member>, <owner>,
<permission>, <principal>, <realm>, <role>, or <user>

Child Element

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}
                    <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}

            <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>

<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 Element

`<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}
```

Example

<owners>

<owners>

This is the parent element for <owner> elements.

Parent Element

<role>

Child Element

<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}
```

Example

<permission>

<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 Element

<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>

<permissions>

This is the parent element for <permission> elements.

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 Element

<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.

<policy-store>

<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 Element

<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>
```

<policy-store>

```
                                <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>

<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 3.7, "Principal Name Comparison Logic."

Parent Element

<principals>

Child Element

<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>

<principals>

This is the parent element for `<principal>` elements.

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 Element

`<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.

<realm>

<realm>

This element specifies a security realm, and the users and roles that belong to the realm.

Parent Element

<jazn-realm>

Child Element

<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.

<role>

<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 Element

<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.

<roles>

<roles>

This is the parent element for <role> elements. It specifies the set of enterprise security roles that belong to a security realm.

Parent Element

<realm>

Child Element

<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 Element

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.

<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 identity 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 Element

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}
            <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}
```

<url>

This element specifies the URL of the codesource being granted the permissions in question. This element is used within a <codesource> element to specify policy grant configuration.

Note the following usage rules:

- You cannot restrict permissions to a single class.

- Code source url value trailing with ".jar" matches the JAR files in the specified directory.

- Code source url value trailing with "/" matches all class files (not JAR files) in the specified directory.

- Code source url value trailing with "/*" matches all files (both class and JAR files) in the specified directory.

- Code source url value trailing with "/-" matches all files (both class and JAR files) in the specified directory, and recursively matches all subdirectories of that directory.

Parent Element

<codesource>

Child Element

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}
                <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

```
<grant>
  <grantee>
    <codesource>
       <url>http://www.oracle.com</url>
```

```
                </codesource>
            </grantee>
```

See `<jazn-policy>` for examples.

<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 true 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 jazn.com realm, for example.
	Values: true or false
	Default: false

Parent Element

<users>

Child Element

<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>

<users>

This is the parent element for the `<user>` elements. It specifies the set of users who belong to a realm.

Parent Element

`<realm>`

Child Element

`<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>

<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 Element

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}
                    <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}
```

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>
```

<value>

<value>

<values>

This is the parent element for the <value> elements, each of which specify a value for an attribute. An attribute can have more than one value.

Parent Element

<attribute>

Child Element

<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}
                    <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}
```

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>
```

C

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 11*g* 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, ProvEven

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	DeleteProvProfile	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
AssertionTemplateAuthoring	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
	ModifyAssertionTemplate	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 J2EE 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, Aritifact,...)
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
OutgoingMessageStringCLOB	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 13.5, "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 describes 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
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
	IAU_DISP_NAME_TRANS	VARCHAR2(4000 Bytes)	Yes	5

Table C–14   (Cont.)   The Audit Schema

Table Name	Column Name	Data Type	Nullable	Column ID
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
	IAU_MODIFIEDATTRIBUTEVALUE	VARCHAR2(2048 Bytes)	Yes	21

Table C–14   (Cont.)   The Audit Schema

Table Name	Column Name	Data Type	Nullable	Column ID
	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_CUSTEVENTOP	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
	IAU_NAMEIDVALUE	VARCHAR2(255 Bytes)	Yes	8

Table C–14 (Cont.) The Audit Schema

Table Name	Column Name	Data Type	Nullable	Column ID
	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_MESSAGETYPE	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
	IAU_PROPERTYTYPE	VARCHAR2(255 Bytes)	Yes	31

Table C–14   (Cont.)   The Audit Schema

Table Name	Column Name	Data Type	Nullable	Column ID
	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
	IAU_ASSERTIONNAME	VARCHAR2(255 Bytes)	Yes	6

Table C–14   (Cont.)   The Audit Schema

Table Name	Column Name	Data Type	Nullable	Column ID
	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
	IAU_ APPLICATIONDOMAINNAME	VARCHAR2(40)	Yes	5

Table C–14   (Cont.)   The Audit Schema

Table Name	Column Name	Data Type	Nullable	Column ID
	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
	IAU_CASECHALLENGEQUESTION	VARCHAR2(4000)	Yes	8
	IAU_CASECHALLENGERESULT	NUMBER(38)	Yes	9

Table C–14 (Cont.) The Audit Schema

Table Name	Column Name	Data Type	Nullable	Column ID
	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
	IAU_POLICYOVERRIDEID	NUMBER	Yes	47
	IAU_POLICYOVERRIDEIDS	VARCHAR2(4000)	Yes	48

Table C–14   (Cont.)   The Audit Schema

Table Name	Column Name	Data Type	Nullable	Column ID
	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

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–15 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.

Table C–15   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
setAuditPolicy	Update audit policy settings.	Online
getAuditRepository	Display audit store settings.	Online
setAuditRepository	Update audit store settings.	Online

Table C–15   (Cont.) WLST Audit Commands

Use this command...	To...	Use with WLST...
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 getNonJavaEEAuditMBeanName

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

```
getNonJavaEEAuditMBeanName('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> getNonJavaEEAuditMBeanName(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'])
```

Argument	Definition
mbeanName	Specifies the name of the component audit MBean for system components.

C.4.2.3 Example

The following command displays the audit settings for all JavaEE 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'])
```

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.

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 `true`, 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.

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 7.5.3, "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')
```

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.

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=CSAuditMBean,name=CSAuditPro
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 7.5.3, "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')
```

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.

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 JavaEE 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 12.3.1, "Manage Audit Policies for Java Components with Fusion Middleware Control"
>
> - Section 12.3.2, "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 "stadl17" -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 JavaEE and JavaSE), 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

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, mussing attributes are indicated by a dash.

D

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 19, "Developing with the User and Role API"

D.1 Mapping User Attributes to LDAP Directories

Table Table D–1 shows 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 WLS Embedded LDAP	Microsoft Active Directory	Sun Java System Directory Server	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
DEPARTMENT	departmentnumber	departmentnumber	departmentnumber	departmentnumber	departmentnumber	departmentnumber
DATE_OF_BIRTH	orcldateofbirth	-	-	-	-	-
BUSINESS_FAX	facsimiletelephonenumber	facsimiletelephonenumber	facsimiletelephonenumber	facsimiletelephonenumber	facsimiletelephonenumber	facsimiletelephonenumber

Table D–1 (Cont.) User Attributes in UserProfile.Property

Attribute	Oracle Internet Directory	Oracle WLS Embedded LDAP	Microsoft Active Directory	Sun Java System Directory Server	Novell eDirectory	OpenLDAP
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
PREFERRE D_ LANGUA GE	Preferredlan guage	preferredlan guage	preferredlangu age	preferredlan guage	preferredla nguage	preferredlan guage
BUSINESS_ POSTAL_ ADDR	postaladdres s	postaladdres s	postaladdress	postaladdres s	postaladdr ess	postaladdres s
MIDDLE_ NAME	orclmiddlena me	-	-	-	-	-
ORGANIZ ATIONAL_ UNIT	ou	ou	ou	ou	ou	ou
WIRELESS _ACCT_ NUMBER	orclwirelessa ccountnumb er	-	-	-	-	-
BUSINESS_ PO_BOX	postofficebox	postofficebox	postofficebox	postofficebox	postofficeb ox	postofficebox
BUSINESS_ STATE	St	st	st	st	st	st
HOME_ ADDRESS	Homepostala ddress	homepostala ddress	homepostalad dress	homepostala ddress	homepostal address	homepostala ddress
NAME_ SUFFIX	Generationq ualifier	generationqu alifier	generationqual ifier	generationqu alifier	generation qualifier	generationqu alifier
BUSINESS_ STREET	street	street	street	street	street	street
INITIALS	initials	initials	initials	initials	initials	initials
USER_ NAME	username (see Note below)	uid	samaccountna me	uid	uid	uid
BUSINESS_ POSTAL_ CODE	postalcode	postalcode	postalcode	postalcode	postalcode	postalcode
BUSINESS_ PAGER	pager	pager	pager	pager	pager	pager
LAST_ NAME	sn	sn	sn	sn	sn	sn
BUSINESS_ PHONE	telephonenu mber	telephonenu mber	telephonenum ber	telephonenu mber	telephonen umber	telephonenu mber
FIRST_ NAME	givenname	givenname	givenname	givenname	givenname	givenname
TIME_ ZONE	orcltimezone	-	-	-	-	-

Table D–1   (Cont.) User Attributes in UserProfile.Property

Attribute	Oracle Internet Directory	Oracle WLS Embedded LDAP	Microsoft Active Directory	Sun Java System Directory Server	Novell eDirectory	OpenLDAP
MAIDEN_ NAME	orclmaidenn ame	-	-	-	-	-
PASSSWOR D	userpasssswo rd	userpassswo rd	userpasssword	userpassssw ord	userpassssw oord	userpasssswo rd
DEFAULT_ GROUP	orcldefaultpr ofilegroup	-	-	-	-	-
ORGANIZ ATION	o	o	o	o	o	o
HOME_ PHONE	homephone	homephone	homephone	homephone	homephon e	homephone
BUSINESS_ MOBILE	mobile	mobile	mobile	mobile	mobile	mobile
UI_ ACCESS_ MODE	orcluiaccessi bilitymode	-	-	-	-	-
JPEG_ PHOTO	jpegphoto	jpegphoto	jpegphoto	jpegphoto	jpegphoto	jpegphoto
MANAGE R	manager	manager	manager	manager	manager	manager
TITLE	title	title	title	title	title	title
EMPLOYE E_ NUMBER	employeenu mber	employeenu mber	employeenum ber	employeenu mber	employeen umber	employeenu mber
LDUser.PA SSWORD	userpasswor d	userpasswor d	userpassword	userpasswor d	userpasswo rd	userpasswor d

> **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 Table D–2 shows 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 WLS Embedded LDAP	Microsoft Active Directory	Sun Java System Directory Server	Novell eDirectory	OpenLDAP
DISPLAY_ NAME	displayname	-	displayname	displayname	displayname	displayname
MANAGER	-	-	-	-	-	-

Table D–2 (Cont.) Role Attribute Values in LDAP Directories

Role Attribute	Oracle Internet Directory	Oracle WLS Embedded LDAP	Microsoft Active Directory	Sun Java System Directory Server	Novell eDirectory	OpenLDAP
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 shows parameters for which the APIs can use default configuration values, and the source of the value in different directory servers.

Table shows the source for Oracle Internet Directory and AD

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
ST_USER_NAME_ATTR	#config	cn
ST_USER_LOGIN_ATTR	#config	samaccountname
ST_GROUP_NAME_ATTR	#config	cn
ST_MAX_SEARCHFILTER_LENGTH	500	500

Table D–3   (Cont.)   Default Values - Oracle Internet Directory and Microsoft Active

Parameter	Oracle Internet Directory	Active Directory
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", "javaserializeddata", "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 shows the source for Sun Java System Directory Server and Novell eDirectory.

Table D–4   Default Values - Sun Java System Directory Server and Novell eDirectory

Parameter	Sun Java System Directory Server	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" }
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"

Table D–4   (Cont.)   Default Values - Sun Java System Directory Server and Novell

Parameter	Sun Java System Directory Server	Novell eDirectory
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 shows the source 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>

Table D–5   (Cont.)   Default Values - OpenLDAP and Oracle Virtual Directory

Parameter	OpenLDAP	Oracle Virtual Directory
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", "javaserializeddata", "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 shows the source for Oracle WebLogic Server LDAP.

Table D–6   Default Values - Oracle WebLogic Server LDAP

Parameter	Oracle WLS Embedded LDAP
RT_USER_OBJECT_CLASSES	{"inetorgperson", "person", "organizationalperson", "wlsUser"}

Table D–6   (Cont.)   Default Values - Oracle WebLogic Server LDAP

Parameter	Oracle WLS Embedded LDAP
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
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.

E

Administration with WLST Scripting and MBean Programming

This appendix describes advanced administrative configuration 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 JavaEE Beans that are used by Oracle Enterprise Manager Fusion Middleware Control and WLST security commands to manage, configure, and monitor Oracle Platform Security Services.

The use of MBeans is recommended in JavaEE 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 JavaSE 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 `jps-config.xml`. This MBean provides the only way to modify configuration data. Update or write operations require server restart to effect changes.	`com.oracle.jps:type=JpsConfig`
Credential Store	Manages domain 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.	`com.oracle.jps:type=JpsCredentialStore`
Global Policy Store	Manages global policies in the domain policy store configured in the default context. Update or write operations do not require server restart to effect changes. All changes are effected immediately.	`com.oracle.jps:type=JpsGlobalPolicyStore`
Application Policy Store	Manages application policies in the domain policy store configured in the default context. Update or write operations do not require server restart to effect changes. All changes are effected immediately.	`com.oracle.jps:type=JpsApplicationPolicyStore`
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.	`com.oracle.jps:type=JpsAdminPolicyStore`

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 user 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 classpath:

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

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

  - `$ORACLE_HOME/modules/oracle.jps_11.1.1/jmxframework.jar`

  - `$ORACLE_HOME/modules/oracle.jps_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();
        invokeAdminPolicyMBeanMethhods();
    }

    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  invokeAdminPolicyMBeanMethhods() {
        //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("Werblogic has Admin role: " + bool);
            bool = (Boolean) wlsMBeanConn.invoke(adminPolName,"checkRole",new
Object[] {"Configurator"}, new String[]{STR_NAME});
            System.out.println("Werblogic 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("Werblogic 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 IV, "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")
  @AttrributeGetterRequiredGlobalSecurityRole(GlobalSecurityRole.Operator)
    public boolean isActive();
  @AttrributeSetterRequiredGlobalSecurityRole(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:

- `@AtrributeGetterRequiredGlobalSecurityRole` specifies that a user must belong to the role Operator to access the get method `isActive`.

- `@AtrributeSetterRequiredGlobalSecurityRole` 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
Operator	Read access to configuration MBeans; read and write access to all run time MBeans	Operator
Monitor	Read access to all MBeans	Monitor

Table E–2   (Cont.) Mapping of Logical Roles to WebLogic Groups

Logical Role	Default Privileges	WebLogic Group
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
ACTION, ACTION_ INFO, UNKNOWN, or attribute setter	System runtime MBean	Admin, Operator

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	Application runtime MBean	Admin, Operator, ApplicationAdmin, ApplicationOperator

F

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 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.

Table F–1 lists the Java system properties available with OPSS.

Table F–1   Java System Properties Used by OPSS

Name	Description
jps.policystore.hybrid.mode	Enables or disables the hybrid mode.
	The hybrid mode is used to facilitate the transition from the Sun java.security.Policy to the OPSS Java PolicyProvider. When the hybrid mode is enabled, the OPSS Java Policy Provider reads from both files, java.policy and system-jazn-data.xml.
	Default value: TRUE.
jps.authz	Enables or disables the delegation of calls to JDK API AccessController.checkPermission, which reduces runtime and debugging overhead.
	Default value: FALSE.
jps.combiner.optimize	Enables or disables the caching of a subject's protection domain.
	Default value: FALSE.

Table F–1   (Cont.) Java System Properties Used by OPSS

Name	Description
`jps.combiner.optimize.lazyeval`	Enables or disables the evaluation of a subject's protection domain when a check permission is triggered. Default value: FALSE.
`jps.auth.debug`	Controls server logging output. Default value: TRUE. For details, see Section I.1.2.1, "jps.auth.debug."
`jps.auth.debug.verbose`	Controls server logging output. Default value: FALSE. For details, see Section I.1.2.2, "jps.auth.debug.verbose."
`oracle.security.jps.config`	Specifies the path to the domain configuration files `jps-config.xml` or `jps-config-jse.xml`. Paths specifications in those files can be absolute or relative to the location of the configuration file.

F.2 OPSS Configuration Properties

This section describes the properties of various instances in the following sections:

- LDAP Policy Store Properties
- LDAP Credential Store Properties
- LDAP Identity Store Properties
- Generic LDAP Properties
- Anonymous and Authenticated Roles Properties
- Policy Provider Framework Properties
- Keystore Properties

F.2.1 LDAP Policy Store Properties

Table F–2 lists the properties of LDAP-based policy store instances.

Table F–2   LDAP-Based Policy Store Properties

Name	Description
`bootstrap.security.principal.key`	The key for the password credentials to access the LDAP policy store, stored in the CSF store. Valid only in J2EE applications. No default value. When the policy store is migrated from a file-based to an LDAP-base store, the out-of-the-box value in the jps-config.xml file is `bootstrap`.
`bootstrap.security.principal.map`	The map for the password credentials to access the LDAP policy store, stored in the CSF store. Valid only in J2EE applications. Default value: BOOTSTRAP_JPS.
`oracle.security.jps.farm.name`	The RDN format of the domain node in the LDAP policy store. Valid in J2EE and J2SE applications. No default value.
`oracle.security.jps.ldap.root.name`	The RDN format of the root node in the LDAP policy store. Valid in J2EE and J2SE applications. No default value.
`ldap.url`	The URL of the LDAP policy store, with the format `ldap://host:port`. Valid in J2EE and J2SE applications. No default value.

Table F–2 (Cont.) LDAP-Based Policy Store Properties

Name	Description
`policystore.type`	The type of the LDAP policy store.
	Valid in J2EE and J2SE applications. No default value.
`oracle.security.jps.policystore.rolemember.cache.type`	The type of the role member cache.
	Valid only in J2EE applications. Valid values:
	■ 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.
	Default value: STATIC.
`oracle.security.jps.policystore.rolemember.cache.strategy`	The type of strategy used in the role member cache.
	Valid only in J2EE applications. Valid values:
	■ 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.
	Default value: FIFO.
`oracle.security.jps.policystore.rolemember.cache.size`	The size of the role member cache.
	Valid only in J2EE applications. Default value: 1000.
`oracle.security.jps.policystore.policy.lazy.load.enable`	Enables or disables the policy lazy load.
	Valid only in J2EE applications. Default value: TRUE.
`oracle.security.jps.policystore.policy.cache.strategy`	The type of strategy used in the permission cache.
	Valid only in J2EE applications.
	■ 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.
	Default value: PERMISSION_FIFO.
`oracle.security.jps.policystore.policy.cache.size`	The size of the permission cache.
	Valid only in J2EE applications. Default value: 1000.
`oracle.security.jps.policystore.refresh.enable`	Enables or disables the refresh of the policy store.
	Valid only in J2EE applications. Default value: TRUE.
`oracle.security.jps.policystore.purge.timeout`	The time, in milliseconds, after which the policy store is refreshed.
	Valid only in J2EE applications. Default value: 43200000 (12 hours).
`oracle.security.jps.policystore.refresh.interval`	The interval, in milliseconds, at which the policy store is polled for changes.
	Valid only in J2EE applications. Default value: 600000.

Table F–2   (Cont.) LDAP-Based Policy Store Properties

Name	Description
`oracle.security.jps.policystore.r efresh.permissions.invalidate.thr eshold`	The number of user's permissions after which the permission cache is invalidated.
	Valid only in J2EE applications. Default value: 50.
`oracle.security.jps.policystore.r olemember.cache.warmup.enable`	This property 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).
	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.
	Valid only in J2EE applications. Default value: FALSE.

The following fragment illustrates the configuration of the above properties for a J2EE application:

```
<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://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>
```

The following fragment illustrates the configuration of the above properties for a J2SE application:

```
<serviceInstance name="policystore.oid" provider="policy.oid">
   <property value="OID" name="policystore.type"/>
   <property name="security.principal" value="cn=orcladmin"/>
   <property name="security.credential" value="welcome1"/>
   <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>
```

Table F–3 lists the reload/refresh properties of LDAP-based policy store instances.

Table F–3   LDAP-Based Policy Store Reload/Refresh Properties

Name	Description
`oracle.security.jps.ldap.cache.enable`	Enables or disables the refresh of the cache. Default value: TRUE.
`oracle.security.jps.ldap.cache.purge.timeout`	The time, in milliseconds, after which the existing cache will be purged and the whole cache will be reloaded. Default value: 43200000 (12 hours).
`oracle.security.jps.ldap.cache.refresh.interval`	The interval, in milliseconds, at which the policies will be scanned for any change in LDAP store such as additions, deletions, or updates. Default value: 43200000 (12 hours).
`oracle.security.jps.ldap.cache.load.factor`	The load factor for the cache object. Default value: 0.7.
`oracle.security.jps.ldap.cache.initial.capacity`	The initial default capacity, in MB, of the cache. Default value: 20.

F.2.2 LDAP Credential Store Properties

Table F–4 lists the properties of LDAP-based credential store instances.

Table F–4   LDAP-Based Credential Store Properties

Name	Description
`bootstrap.security.principal.key`	The key for the password credentials to access the LDAP credential store, stored in the CSF store. Valid only in J2EE applications. No default value. When the policy store is migrated from a file-based to an LDAP-base store, the out-of-the-box value in the jps-config.xml file is `bootstrap`.
`bootstrap.security.principal.map`	The map for the password credentials to access the LDAP credential store, stored in the CSF store. Valid only in J2EE applications. Default value: BOOTSTRAP_JPS.
`oracle.security.jps.farm.name`	The RDN format of the domain node in the LDAP credential store. Valid in J2SE and J2EE applications. No default value.
`oracle.security.jps.ldap.root.name`	The RDN format of the root node in the LDAP policy store. Valid in J2SE and J2EE applications. No default value.
`ldap.url`	The URL of the LDAP credential store, with the format `ldap://host:port`. Valid in J2SE and J2EE applications. No default value.

The following fragment illustrates the configuration of the above properties for a J2EE 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"/>
```

```
        </serviceInstance>
```

The following fragment illustrates the configuration of the above properties for a J2SE application:

```
<serviceInstance name="credstore_ldap_instance" provider="credstore_ldap_
provider">
    <property name="security.principal" value="cn=orcladmin"/>
    <property name="security.credential" value="welcome1"/>
    <property name="oracle.security.jps.farm.name" value="cn=wls-jrfServer"/>
    <property name="oracle.security.jps.ldap.root.name" value="cn=jpsTestNode"/>
    <property name="ldap.url" value="ldap://stttt:3060"/>
</serviceInstance>
```

F.2.3 LDAP Identity Store Properties

Table F–5 lists the properties of LDAP-based identity store instances.

Table F–5   LDAP-Based Identity Store Properties

Name	Description
idstore.type	The type of the LDAP identity store.
	Valid only in J2SE applications. No default value.
security.principal.alias	The CSF map name.
	Valid only in J2SE applications. No default value.
security.principal.key	The CSF key name.
	Valid only in J2SE applications. No default value.
ldap.url	The LDAP URL value.
	Valid only in J2SE applications. No default value.
user.search.bases	The user search base for the LDAP server in DN format.
	Valid only in J2SE applications. No default value.
group.search.bases	The group search base for the LDAP server in DN format.
	Valid only in J2SE applications. No default value.
idstore.config.provider	The out-of-the-box idstore provider.
	Valid only in J2EE applications. The only supported value is:
	oracle.security.jps.wls.internal.idstore.WlsLdapIdStoreConfig Provider

The following fragment illustrates the configuration of the above properties:

```
<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 Generic LDAP Properties

Table F–6 lists generic properties of LDAP-based stores that can be specified in service instances. 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`	Specifies the type of LDAP connection that the JNDI connection pool uses.
	Values: none, simple, and DIGEST-MD5.
	Default: simple.
`connection.pool.max.size`	Specifies the maximum number of connections in the LDAP connection pool.
	Values: integer
	Example: 30
`connection.pool.min.size`	Specifies the minimum number of connections in the LDAP connection pool.
	Values: integer
	Example: 5
`connection.pool.protocol`	Specifies the protocol to use for the LDAP connection.
	Values: plain, ssl.
	Default: plain.
`connection.pool.provider.type`	Specifies the connection pool to use.
	Values: JNDI, IDM.
	Default: JNDI.
`connection.pool.timeout`	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.
	Values: an integer in string form.
	Defaults to 300000 (5 minutes)
`oracle.security.jps.ldap.max.retry`	Specifies the maximum number of retry attempts if there are problems with the LDAP connection.
	Values: integer
	Example: 5

The following fragment illustrates the configuration of the above properties:

```
<jpsConfig ... >
    ...
```

```
<!-- These are various JPS common properties used for LDAP operations -->
<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 of anonymous users, anonymous roles, and authenticated roles. Some of them may also be used to configure the anonymous service or an identity store login module.

Table F–7   Anonymous and Authenticated Roles Properties

Name	Description
`anonymous.role.description`	Provides a description of the anonymous role.
	Example: `This is the anonymous role used by the anonymous service instance.`
`anonymous.role.name`	Specifies the principal name of the anonymous role.
	Default value: `anonymous-role`
`anonymous.role.uniquename`	Specifies the name of the anonymous role.
	Default value: `anonymous-role`
`anonymous.user.name`	Specifies the principal name of the anonymous user.
	Default value: `anonymous`
`authenticated.role.description`	Provides a description of the authenticated role.
	Example: `This is the role used for authenticated users by the identity store service instance.`
`authenticated.role.name`	Specifies the principal name of the role used for authenticated users.
	Default value: `authenticated-role`
`authenticated.role.uniquename`	Specifies the name of the authenticated role.
	Default value: `authenticated-role`
`remove.anonymous.role`	Specifies whether the anonymous role should be removed from the subject after a user is authenticated.
	Default value: FALSE.

F.2.6 Policy Provider Framework Properties

Table F–8 lists the properties of the policy provider framework.

Table F–8   Policy Provider Framework Properties

Name	Description
`policystore.delegation.permission`	Specifies the fully qualified class name of the permission that extends PolicyDelegationPermission. This is used in runtime for custom provider delegation by the policy framework. By default, this property is not specified in `jps-config.xml`.

Table F–8   (Cont.) Policy Provider Framework Properties

Name	Description
`policystore.role.memberattr`	Specifies the attribute of a static LDAP role object that specifies the distinguished names (DNs) of the members of the role.
	Example: `uniquemember`
`policystore.role.nameattr`	Specifies the name of the LDAP attribute that uniquely identifies the name of the role.
	Example: `cn`
`policystore.role.objectclass`	Specifies LDAP schema object classes that represent a role. If specifying multiple classes, separate the classes with a space.
	The default for Sun Java System Directory Server is `groupOfUniqueNames`. For Active Directory, the default is `group`.
	Example: `orclrole`
`policystore.role.searchbase`	Specifies a list of space-delimited distinguished names (DN) in the LDAP directory that contains roles.
	Example: `cn=groups,dc=us,dc=abc,dc=com`
`policystore.role.searchscope`	Specifies how deep in the LDAP directory tree to search for roles.
	Default values: `subtree` or `onelevel`

The following example illustrates the configuration of a policy store service provider, using an Oracle Internet Directory, and its use in a jpscontext.

```
<jpsConfig ... >
   ...
   <serviceProviders>
      <serviceProvider type="POLICY_STORE" name="policystore.ldap.provider"
                 class= "oracle.security.jps.internal.policystore.ldap.LdapPolicyStoreProvider">
         <description>LDAP-based PolicyStore</description>
         <property name="policystore.type" value="OID"/>
         <property name="connection.pool.max.size" value="30"/>
         <property name="connection.pool.provider.type" value="IDM"/>
      </serviceProvider>
   </serviceProviders>
   ...
   <serviceInstances>
      <serviceInstance name="policystore.oid" provider="policystore.ldap.provider">
         <property name="max.search.filter.length" value="4096"/>
         <property name="security.principal" value="cn=orcladmin"/>
         <property name="security.credential" value="password"/>
         <property name="ldap.url" value="ldap://xyz.us.oracle.com:389"/>
         <property name="policystore.jpsbase" value="cn=jps,cn=oraclecontext"/>
         <property name="policystore.role.objectclass" value="orclrole"/>
         <property name="policystore.role.searchbase" value="cn=roles"/>
         <property name="policystore.role.searchscope" value="subtree"/>
         <property name="policystore.role.nameattr" value="cn"/>
         <property name="policystore.role.memberattr" value="uniquemember"/>
         <property name="policystore.role.roleheirarchyattr" value="assignedRoles"/>
      </serviceInstance>
   </serviceInstances>
   ...
   <jpsContexts default="default">
      <jpsContext name="default">
```

```
            <serviceInstanceRef ref="policystore.oid"/>
        </jpsContext>
    </jpsContexts>
</jpsConfig>
```

F.2.7 Keystore Properties

Table F–9 lists the properties that configure keystore services. When using encryption or signing, you specify a password to retrieve the private key and provide the map to access the keystore credential.

Table F–9   Keystore Properties

Name	Description
keystore.path	The path to the keystore.
keystore.type	The type of keystore.
keystore.csf.map	The map of the keystore credentials in the credential store.
keystore.pass.csf.key	The key to the private password in the credential store.
keystore.sig.csf.key	The private key in the credential store when using signing.
keystore.enc.csf.key	The private key in the credential store when using encryption.

The following example illustrates the configuration of the above properties:

```
<serviceInstance provider="keystore.provider" name="keystore">
    <description>Default JKS Keystore Service</description>
    <property value="${oracle.instance}/config/default-keystore.jks" name="keystore.path"/>
    <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="sign-csf-key" name="keystore.sig.csf.key"/>
    <property value="enc-csf-key" name="keystore.enc.csf.key"/>
</serviceInstance>
```

G

Upgrading Security Data

This appendix describes the procedure to update JAZN security data used in release 10.1.3.x to security data used by OPSS in release 11*g* Release 1 (11.1.1) using the offline WLST command `upgradeSecurityStore`, which allows the separate upgrading of identity, policy, or credential data.

> **Upgrading from 11gR1 to 11gR1 PS1:** Upgrading details are documented in section Special Instructions for Oracle Fusion Middleware 11g Release 1 (11.1.1.1.0) in *Oracle Fusion Middleware Installation Planning Guide*.

G.1 Upgrading Security Data

OPSS replaces JAZN (which existed in 10.1.3.x. JAZN to store identities and policies) and old data in JAZN can be upgraded to OPSS as described in this section.

If the target of the upgrading is an LDAP-based repository, then some setting up before the command is used is required, as described in Section 8.1.2, "Prerequisites to Using an LDAP-Based Policy Store."

The commands listed below are **offline** (that is, they do not require a connection to a running server to operate) and can be run in interactive mode or in script mode. In interactive mode, you enter the command at a command-line prompt and view the response immediately after. In script mode, you write commands in a text file (with a py file name extension) and run it without requiring input, much like the directives in a shell script.

> **Important:** Before invoking a security-related WLST command in a shell, you must run the script `wlst.sh`, as illustrated in the following sample:
>
> `> sh $ORACLE_HOME/common/bin/wlst.sh`
>
> This ensures that the required JARs are added to the classpath. Failure to run the above script in a new shell renders the WLST commands unusable.

> **Note:** To prevent security vulnerabilities during the upgrade, enable an SSL transmission between Oracle Containers for Java EE and Oracle HTTP Server.

Script and Interactive Modes Syntaxes

The command 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 11*g* 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 11*g* 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 11*g* 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"])
```

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

- `type` specifies the kind of security data being upgraded. The only valid values are xmlIdStore, xmlPolicyStore, oidPolicyStore, or xmlCredStore.

- `jpsConfigFile` specifies the location of a configuration file `jps-config.xml` relative to the directory where the command is run. The target store of the upgrading is read from the context specified with the argument `dst`.

- `srcJaznDataFile` specifies the location of a 10.1.3.x jazn data file relative to the directory where the command is run. This argument is required if the specified `type` is xmlIdStore, xmlPolicyStore, or xmlCredStore.

- `srcJaznConfigFile` specifies the location of a 10.1.3.x jazn configuration file relative to the directory where the command 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.

G.1.1 Examples of Use

In the following examples, arguments are written in separate lines for clarity of exposition.

Example 1 - Upgrading Identities

The following invocation illustrates the migration of 10.1.3 file-based identities to an 11*g* 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 command assumes that: (a) the files `jps-config-idstore.xml` and `jazn-data.xml` are located in the directory where the command 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 migrates every user in the element `<users>`, to the XML identity store `R11idStore`.

Example 2 - Upgrading to File-Based Policies

The following invocation illustrates the migration of a 10.1.3 file-based policy store to an 11*g* Release 1 (11.1.1) policy store:

```
upgradeSecurityStore -type xmlPolicyStore
                     -jpsConfigFile jps-config.xml
                     -srcJaznDataFile jazn-data.xml
                     -dst destContext
```

This use of the command assumes that: (a) the files `jps-config.xml` and `jazn-data.xml` are located in the directory where the command is run; and (b) 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`.

Example 3 - Upgrading to Oracle Internet Directory LDAP-based Policies

The following invocation illustrates the migration of a 10.1.4 Oracle Internet Directory LDAP-based policy store to an 11*g* 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>
```

H

References

This appendix contains references documentation useful to developes.

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

I

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:

I.1 Diagnosing Security Errors

This section describes how to detect and solve security errors and it contains the following topics:

- Log Files
- 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.

I.1.1 Log Files

This section describes the various log files supported by Oracle WebLogic Server and how to configure, set log levels, and locate and view log files with Fusion Middleware Control, in the following sections:

- [Diagnostic Log Files](#)
- [Generic Log Files](#)
- [Audit Diagnostic Log Files](#)
- [Using Fusion Middleware Control Logging Support](#)

I.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.log`xxxxx, where xxxxx denotes an integer between 1 and 99999.

Here are some examples of diagnostic file full names: *DomainName*/`servers/AdminServer/logs/AdminServer-diagnostic.log00 001` (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.

I.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*.`log`xxxxx and `domain.log`xxxxx, 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.

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 10, Managing Log Files and Diagnostic Data, in *Oracle Fusion Middleware Administrator's Guide*.

I.1.1.3 Audit Diagnostic Log Files

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 I–1 lists the various diagnostic log files.

Table I–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)
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)

Table I–1   (Cont.)   Log Files for Audit Diagnostics

Component	Log Location	Configuring Loggers
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

I.1.1.3.1   Configuring the 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

I.1.1.3.2   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.

I.1.1.4 Using Fusion Middleware Control Logging Support

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 I.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.

I.1.2 System Properties

To increase the logging output with Java options at server start, add the following Java options to the script that starts your Oracle WebLogic Server and restart the server:

- jps.auth.debug

- jps.auth.debug.verbose

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.

I.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/kebqfo/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.

I.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/jspservlet/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.vebose` is set to
false.

I.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

I.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: 0000Hum5kxw7MAn54nU4Ui19PD8S000005,0]

  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 JavaSE 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.

I.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.

I.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.

I.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/wsil-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
#
```

I.2 Reassociation Failure

Policy and credential reassociation from an XML-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=JPSContext,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.

I.2.1 Missing Policies in Reassociated Policy Store

Symptom

When an XML-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. Granted code-based 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_Ll00002s,1:5001] [arg: 11.1.1.0.0]  Policy schema
version set to 11.1.1.0.0
```

If re-associating to a Release 11*g* 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
```

I.3 Server Fails to Start - 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

I.4 Server Fails to Start - 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.

I.5 Server Fails to Start - 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.

I.6 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 obtained 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);
```

```
// Similarily, 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());
> ```

I.7 Failure to Connect to an LDAP Server

This section explains the likely reasons why a connection to an Oracle Virtual Directory or 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 I.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.

I.8 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 J2SE 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."

I.9 Failure to Access Data in the Domain 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 permisson ( 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 8.4.1, "Managing Policies with Fusion Middleware Control."

To include a permission using a WLST command, see Section 8.4.2, "Managing Policies with WLST Commands."

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>
```

I.10 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 or an Oracle Virtual 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 8.1.2, "Prerequisites to Using an LDAP-Based Policy 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 reachable. Typically, the setting of this port involves an LDAP server administrator.

I.11 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 you application is using.

On the one hand, the application stripe (or subset of policies in the domain 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 authorized 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.

I.12 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 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 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.

I.13 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>
</jazn-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.

I.14 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.

I.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.

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: appplication 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.

I.16 Characters in Policies

This section explains several issues related to characters used in policies.

I.16.1 Use of Special Characters in Oracle Internet Directory 10.1.4.3

Symptom

When the domain 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).

Solution

Apply the patch for bug number 7711351 to Oracle Internet Directory 10.1.4.3.

I.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 XML-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*")
```

I.16.3 Missing Newline Characters in XML Policy Store

In an XML-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>
```

I.17 Granting Permissions in J2SE Applications

This section describes the correct way to code a grant in J2SE applications. Even though the problem described is not an issue in J2EE applications, for maximum portability, it is recommended that this solution be used in J2EE 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[]`

I.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

I.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 13.1.3, "Set Up Oracle Reports in Oracle Business Intelligence Publisher".

I.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.

I.19 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.

Index

W

X