Oracle® Fusion Middleware

Administrator's Guide for Oracle Entitlements Server

11*g* Release 2 (11.1.2)

E27153-03

November 2012

ORACLE®

Oracle Fusion Middleware Administrator's Guide for Oracle Entitlements Server, 11*g* Release 2 (11.1.2)

E27153-03

Primary Author: Michael Teger

Contributing Author:

Contributor:

Contents

12 Customizing the Administration Console

13 Management Tasks

A Installation and Configuration Parameters

B Configuring Attribute Retrievers Manually

C Managing Advanced Policies with WLST

Index

List of Tables

List of Figures

List of Examples

Preface

The *Oracle Fusion Middleware Administrator's Guide for Oracle Entitlements Server* provides information on system configuration administration and management of policy objects using the Oracle Entitlements Server Administration Console. Manual configuraton using back end files is also covered.

Audience

The intended audience of this guide is security administrators.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible to all users, including users that are disabled. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=doc.

Accessibility of Code Examples in Documentation

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

Accessibility of Links to External Web Sites in Documentation

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

Access to Oracle Support

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

Related Documents

For more information, see the following documents:

- *Oracle Fusion Middleware Developer's Guide for Oracle Entitlements Server*
- *Oracle Fusion Middleware Installation Guide for Oracle Identity and Access Management*
- *Oracle Fusion Middleware Quick Installation Guide for Oracle Identity and Access Management*
- *Oracle Fusion Middleware Application Security Guide*
- *Oracle Fusion Middleware High Availability Guide*
- *Oracle Fusion Middleware Integration Guide for Oracle Access Manager*
- *Oracle Fusion Middleware Securing Oracle WebLogic Server*
- *Oracle Fusion Middleware Administrator's Guide*

Conventions

The following text conventions are used in this document:

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

1

Introducing Oracle Entitlements Server

This chapter contains an overview of the features and architecture of Oracle Entitlements Server. It contains the following sections:

1.1 About Access Control

Access control refers to a system used to grant or deny access to an enterprise's information, systems or resources. For the purposes of this documentation, the entity that is being protected is referred to as the protected *resource* while the entity to which access is granted or denied is referred to as the *subject* or *principal* (most often a real person). An *authorization policy* matches a subject with a set of *operations* that determine what the subject is allowed to see and do within the protected resource. The operations are dependent on the type of resource. For example, the operations attached to a text file (read, modify, delete) are different from those that can be attached to a banking application (view account, transfer money, modify profile).

Access control assures that only authorized subjects can access protected resources thus preventing the resources from unauthorized or inadvertent modification. Authorization of a subject typically occurs after authentication. In general, an access control system may comprise two types of authorization:

- Coarse grained authorization is perimeter authorization that uses technology principally focused on "keeping the bad guys out." Generally, it is performed by interceptors outside the application making the authorization call. It takes into account a URL and the policies regarding the subject requester.
- Fine grained authorization is more detailed, and primarily controlled by the application making the authorization call. It takes into account the URL of the protected resource and its configured policies as well as information that may include resource-specific or user-specific attributes and the context of the request. For example, granting an employee access to a portal during normal business hours and denying this access during the weekend hours would call for fine grained authorization.

Thus, an access control system must support a policy model that is easy to administer yet allows for complex sets of conditions under which access can be granted (or denied). Oracle Entitlements Server is a product that provides centralized policy

management with centralized *or* distributed access control enforcement for all types of resources including software components and application business objects.

1.2 Overview of Oracle Entitlements Server

Oracle Entitlements Server is a fine-grained authorization product that allows an organization to protect its resources by defining and managing policies that control access to, and usage of, these resources. Access privileges are defined in a policy by specifying who can do what to which resource, when it can be done, and how. The policy can enforce controls on all types of resources including software components (URLs, Java Server Pages, Enterprise JavaBeans, methods, servlets and the like used to construct an application) and business objects (representations of user accounts, personal profiles and contracts such as bank accounts in a banking application, patient records in a health care application, or anything used to define a business relationship).

Oracle Entitlements Server supports the creation of Role Mapping Policies and Authorization Policies. A *Role Mapping Policy* defines constraints regarding which users are assigned roles and can map Application Roles in the Policy Store to external groups in the Identity Store. This mapping allows users in external groups to access resources as specified by the Application Roles. The mapping is allowed to be many-to-many. An *Authorization Policy* defines access to the protected software components and business objects. The following sections contain information on the previous releases of Oracle Entitlements Server and the features developed for this release of Oracle Entitlements Server.

- Section 1.2.1, "Understanding Oracle Entitlements Server Releases"
- Section 1.2.2, "Using the Authorization Policy Manager Console"
- Section 1.2.3, "Features of Oracle Entitlements Server 11gR2"

1.2.1 Understanding Oracle Entitlements Server Releases

Oracle Entitlements Server 11g represents a consolidation of Oracle Platform Security Services with Oracle Entitlements Server 10g (formerly BEA AquaLogic Enterprise Security). While Oracle Platform Security Services (OPSS) is a Java Authentication and Authorization Services (JAAS) security provider that offers coarse-grained authorization, Oracle Entitlements Server 11g is an end-to-end enterprise solution that includes multiple technologies including Java Standard Edition (SE) and Enterprise Edition (EE), service-oriented architecture (SOA) and .NET. Oracle Entitlements Server 11g offers fine-grained authorization in which a context for the authorization request may be provided and access is granted or denied based on this. The Oracle Entitlements Server 11g architecture is based on the interaction model described in the core eXtensible Access Control Markup Language (XACML) specifications. See Section 1.3, "Overview of the Oracle Entitlements Server Architecture" for details.

1.2.2 Using the Authorization Policy Manager Console

Authorization Policy Manager is the administration console for Oracle Entitlements Server. For purposes of the Oracle Entitlements Server documentation set, *Authorization Policy Manager* and variations of the *Oracle Entitlements Server* Administration Console (Administration Console, Console and the like) may be used interchangeably. See Chapter 3, "Getting Started" for more information on the Administration Console.

Note: Oracle Entitlements Server is not meant to contain the functionality of the product released as Oracle Authorization Policy Manager product. When referred to in this documentation set, Authorization Policy Manager is simply the Administration Console.

1.2.3 Features of Oracle Entitlements Server 11gR2

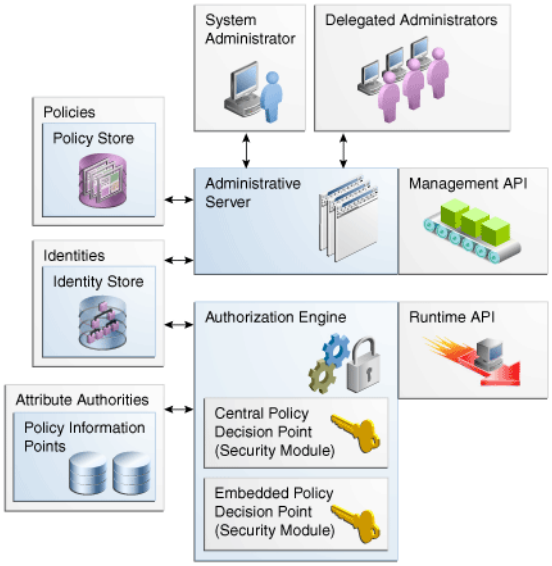
Oracle Entitlements Server 11gR2 offers fine-grained authorization and centralized entitlements management across heterogeneous application environments. Additionally, this release of Oracle Entitlements Server 11gR2 offers:

- Support for all data types and functions as defined in the XACML 3.0 specifications
- Support for viewing and modification of Oracle Platform Security Services created Application Grants (application policies)
- Logging enhancements to simplify policy evaluation analysis
- Policy Simulator
- Debugging enhancements
- Support for permission based advanced policies
- Enhanced PEP API query requests
- Administration Console support for plugging in multiple identity stores
- SMConfig UI, the Security Module configuration interface
- Support for .NET resources
- Security Modules for Java containers, Web Services (on WebLogic Server), Oracle Service Bus, Microsoft Sharepoint, JBoss, and Tomcat

1.3 Overview of the Oracle Entitlements Server Architecture

From a high-level, Oracle Entitlements Server (OES) comprises centralized policy management with centralized *or* distributed policy decision making. The architecture of Oracle Entitlements Server is based on the interaction model of entities discussed in the XACML specifications. This model defines entities that provide a flexible architecture that can adapt to many components and deployments. Figure 1–1 illustrates the components of the Oracle Entitlements Server. Each of these components corresponds to one of the XACML entities.

Figure 1–1 Components of Oracle Entitlements Server



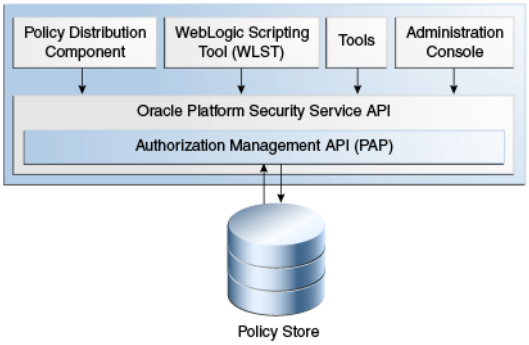
The following sections contain information on the Oracle Entitlements Server components and how they conform with the XACML entities.

- Section 1.3.1, "The Policy Administration Point"
- Section 1.3.2, "The Policy Decision Point and the Policy Enforcement Point"
- Section 1.3.3, "The Policy Information Point"

1.3.1 The Policy Administration Point

The Policy Administration Point (PAP) is where policies used to protect a specified protected resource are created and managed. The PAP makes these rules available to the Policy Decision Point in order for that entity to reach a *grant* or *deny* decision for a request to access the protected resource. The Oracle Entitlements Server PAP is comprised of the authorization management application programming interfaces (API). (In Figure 1–1, Administrative Server and Management API represent the PAP.) Access to (and use of) the PAP can be achieved with the Administration Console, the WebLogic Scripting Tool (WLST), the Policy Distribution Component or various other tools (including `migrateSecurityStore` and those for reassociation). Figure 1–2 illustrates the architecture.

Figure 1–2 Oracle Entitlements Server PAP Architecture



1.3.2 The Policy Decision Point and the Policy Enforcement Point

When Oracle Entitlements Server is deployed, a Policy Decision Point (PDP) receives a request for authorization, evaluates it based on applicable policies, reaches a decision and returns the decision to the Policy Enforcement Point (PEP), the entity which first made the authorization call. The PEP then enforces the decision.

> **Note:** The PDP can retrieve additional subject, resource, action and environment attributes from a Policy Information Point to add context to the request. See Section 1.3.3, "The Policy Information Point" for more information.

The PEP doesn't intercept authorization requests; it intercepts a request for access sent to an application and authorizes the request by forming an authorization request and sending it to the PDP. Upon receipt, it enforces the *grant* or *deny* decision returned by the PDP. The PEP is a software component that can be the protected application itself or a Security Module.

> **Note:** The PDP may also return information with the decision - referred to as an *obligation* - that allows the decision to be enforced within a particular context. The application is not forced to act upon these obligations. See Chapter 2, "Understanding the Policy Model" for more information.

Oracle Entitlements Server offers two types of Security Modules. One type was developed to act purely as a PDP by receiving requests and reaching decisions. The other type combines this PDP functionality with that of the PEP; thus, this Security Module type receives requests, reaches authorization decisions and enforces the decision. Chapter 7, "Deploying the Policy Decision Point" contains information on how the Security Module may be deployed. The following sections illustrate how the Security Modules work.

- Section 1.3.2.1, "Security Module as PDP"
- Section 1.3.2.2, "Security Module as Combination PDP / PEP"
- Section 1.3.2.3, "Understanding the Types of Security Modules"

> **Note:** A Security Module is referred to as the *OES Client* before and during its installation process.

1.3.2.1 Security Module as PDP

When a Security Module acts purely as a PDP, its only functionality is decision making. It receives an authorization request and returns its decision to the PEP that originally made the authorization call. With the Security Module acting solely as the PDP, an external entity must act as the PEP - make the authorization call (using the Oracle Entitlements Server authorization API) and enforce the returned decision.

Figure 1–3 illustrates the process when the PEP entity making the call is the resource (application) itself. It receives the request for access, initiates authorization (through communication with the Security Module) and enforces the returned decision.

Figure 1–3 Application Acting as PEP Requests Decision from PDP

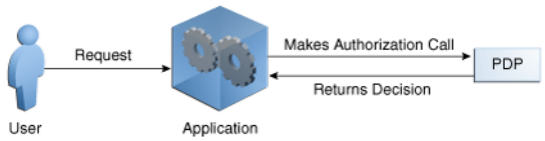
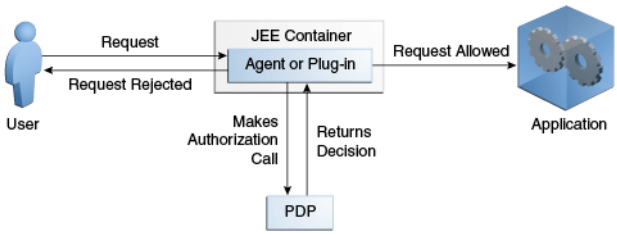


Figure 1–4 illustrates the process when the PEP entity is an agent or a plug-in (or similar software component) that intercepts the request before it reaches the application. The software component intercepts the request for access, initiates authorization (through communication with the Security Module) and forwards the decision returned from the Security Module to the application.

Figure 1–4 Agent Acting as PEP Intercepts Request and Makes Decision



Working together, these scenarios can offer a flexible authorization service. For example, the intermediary Web Services/XML gateway can request an authorization decision for a subject to access a portal. Assuming this primary decision is granted, the Web Services/XML gateway itself can then request secondary authorization decisions used to personalize the portal for the user that has been granted access.

1.3.2.2 Security Module as Combination PDP / PEP

When a Security Module acts in tandem as a PDP and a PEP, it intercepts authorization requests, makes a decision, and enforces the decision. With this release of Oracle Entitlements Server, the WebLogic Server, Oracle Service Bus and Microsoft SharePoint Security Modules work in this manner.

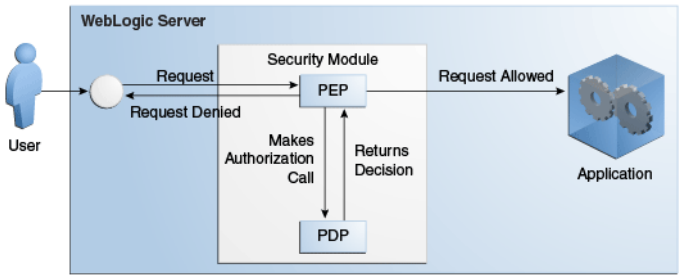
> **Note:** For more information on Oracle WebLogic Server, see the Oracle WebLogic Server Documentation Library at `http://download.oracle.com/docs/cd/E21764_01/wls.htm`.

For example, the WebLogic Server Security Module plugs directly into an Oracle WebLogic Server container that executes the protected application and will automatically request an authorization. In this scenario, a subject-initiated request to the application is intercepted by the WebLogic Server for authorization. The WebLogic Server, after successful authentication, attempts to authorize the request by making a call to a set of authorization providers configured during the Security Module's installation.

The Role Mapping and Authorization Proxy providers communicate with the Oracle Entitlements Server authorization engine (calling the PEP API which, in turn, calls the PDP). The PDP computes a decision and returns the decision to the PEP which returns

an appropriate response to the WebLogic Server. (Optionally, the PDP may return an obligation with the decision.) If access is denied, the WebLogic Server throws a security exception and prevents access. If access is permitted, the WebLogic Server allows access. Figure 1–5 illustrates this scenario.

Figure 1–5 Security Module as PDP and PEP



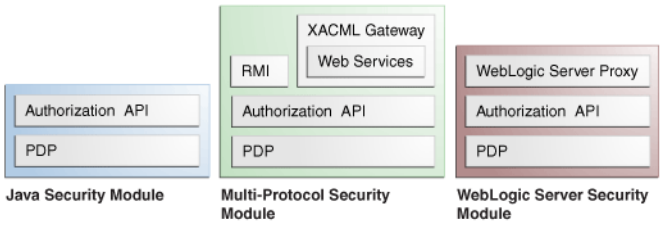
The benefit of using the providers is fine-grained component level authorization. For example, you can use the providers to protect access to a servlet URL while allowing the servlet itself to make additional PEP API calls to decide which elements should be rendered on the returned page (as illustrated by Figure 1–3, "Application Acting as PEP Requests Decision from PDP"). By default, the Role Mapping and Authorization Proxy providers are not enabled.

> **Note:** See Section 9.4, "Securing WebLogic Server Resources" for the procedure to enable the authorization providers using the WebLogic Server console. Post-configuration parameters are documented in Section A.2.5, "WebLogic Server Security Module."

1.3.2.3 Understanding the Types of Security Modules

Figure 1–6 illustrates how the various types of Security Modules have been developed.

Figure 1–6 Security Module Architecture



Based on this topology, the services of a Security Module can be invoked in several ways.

- The Java Security Module is a generic PDP that provides authorization decisions using Java or .NET API. This Security Module is supported on the following containers:
  - Java, Standard Edition (JSE)
  - WebSphere
  - JBoss

  - Tomcat
  - Microsoft Sharepoint Server

- The Multi-Protocol Security Module is an authorization service (based on service-oriented architecture principles) wrapped around a generic Java Security Module. It provides authorization decisions using RMI, Web Services and XACML (request and response). The Multi-Protocol Security Module can be co-located with the protected application or deployed on a central server.

  Note: Figure 1–4, "Agent Acting as PEP Intercepts Request and Makes Decision" works similarly to the Multi-Protocol Security Module with an XML gateway; it intercepts requests and forces authorization before sending them to on to the destination.

- The WebLogic, Oracle Service Bus (OSB) and Sharepoint Security Modules are Security Modules that include both a PDP and a PEP. They can receive requests directly from the container without the need for explicit authorization API calls.

  Caution: Security Modules deployed centrally are supported for RMI, Web Services or XACML calls but Oracle Entitlements Server and the Security Modules cannot be in the same WebLogic Server domain. For more information on the deployment models, see Chapter 7, "Deploying the Policy Decision Point."

See *Oracle Fusion Middleware Developer's Guide for Oracle Entitlements Server* for more information on requesting authorization decision and how Security Modules get updated policy information using the Policy Distribution Service. See Chapter 8, "Managing Security Module Configurations" and Chapter 9, "Securing Environment Specific Resources" for details on configuring and using Security Modules.

1.3.3 The Policy Information Point

The Policy Information Point (PIP) is a data repository - a source from which information can be retrieved for use when evaluating policies for an authorization decision. This allows policies to be data-driven in that the value of an attribute can impact the access decision. For example, if access to transfer money from a bank account is based on how much money is currently in the account, an attribute retriever can be used to get a value for the current balance.

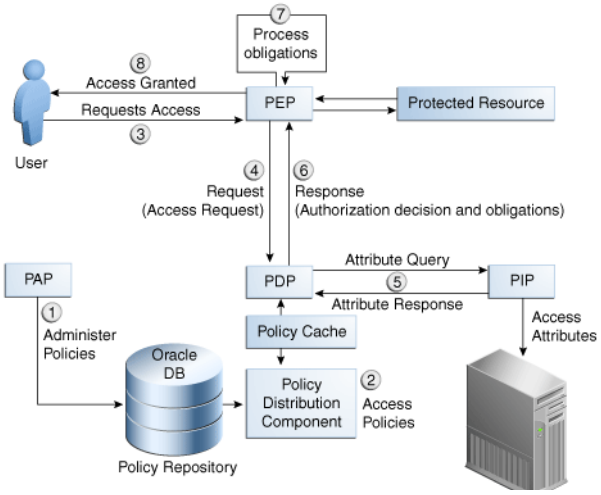
In an Oracle Entitlements Server deployment, *attribute retrievers* serve the PIP thus, the terms PIP and attribute retriever may be used interchangeably in this documentation. The Attribute Authorities component illustrated in Figure 1–1, "Components of Oracle Entitlements Server" would be considered the PIP. Out of the box, an attribute retriever is available for both, LDAP and relational database data sources. See *Oracle Fusion Middleware Developer's Guide for Oracle Entitlements Server* and Appendix B, "Configuring Attribute Retrievers Manually" for more information on attribute retrievers.

1.4 How Oracle Entitlements Server Processes Authorization Policies

The Oracle Entitlements Server authorization process involves the components described in Section 1.3, "Overview of the Oracle Entitlements Server Architecture". When a policy decision is requested, the PDP evaluates all policies related to the

request and returns a *grant* or *deny* decision to the calling application. Figure 1–7 illustrates how the data flows during the policy authorization process.

Figure 1–7 How Data Flows in the Policy Authorization Process



1. Oracle Entitlements Server (acting as a PAP) is used to create and manage policies to protect a particular resource.
2. Policies in the policy repository are pushed to a policy cache, local to the PDP, by the Policy Distribution Service. The PDP reads policies from this cache. See *Oracle Fusion Middleware Developer's Guide for Oracle Entitlements Server* for more information.
3. A request for a resource is received by the PEP protecting it. The PEP can be the application itself or a Security Module - whichever makes the authorization call to Oracle Entitlements Server.
4. The PEP makes an authorization call to the PDP.
5. The Security Module PDP queries for additional subject, resource, action and environment attributes from the appropriate data source PIP.
6. The Security Module PDP evaluates the request and returns a response (and applicable obligations) to the PEP in the form of an authorization decision to grant or deny access.
7. The PEP fulfills any obligations, if applicable. An *obligation* is information returned with the decision upon which the PEP may or may not act. For example, an obligation may contain additional information concerning a decision to deny. The PEP entity is responsible for obligation fulfillment based on its settings. Oracle Entitlements Server is only responsible for forwarding the obligation based on policy configuration.
8. If access is permitted, the PEP grants the requester access to the resource; otherwise, access is denied.

Section 2.2, "How Oracle Entitlements Server Evaluates Policies" contains more details.

1.5 About the Supported Access Control Standards

Oracle Entitlements Server supports a number of access control models. Many access control products support only one of these models but Oracle Entitlements Server has implemented a policy model with the flexibility to support many of them. You can

deploy strictly based on one model or mix and match pieces of different models. The following sections contain information on the access control models.

- Section 1.5.1, "Role-based Access Control (RBAC)."
- Section 1.5.2, "Attribute-Based Access Control (ABAC)."
- Section 1.5.3, "Java Permissions."
- Section 1.5.4, "XACML"
- Section 1.5.5, "PEP API"

1.5.1 Role-based Access Control (RBAC)

The Role-Based Access Control (RBAC) authorization model uses *roles* to define the privileges of a user. First, roles are created. Following, *permissions* to perform certain operations are assigned to the roles and, finally, *users* or *external groups* are assigned to the roles. Through role assignment, the assignee acquires the right to perform the assigned operations. Thus, RBAC makes management of individual permissions simply a matter of assigning the appropriate roles to the appropriate entity. Roles can also be combined in a hierarchy where higher-level roles subsume permissions owned by sub-roles. This model is useful for more coarse-grain authorization; for example, when your authorization requirements are not complex. Application Roles are used for modeling Oracle Entitlements Server deployments based on RBAC.

1.5.2 Attribute-Based Access Control (ABAC)

Attribute-Based Access Control (ABAC) provides the capability to define fine grained authorization using attributes. Roles need not be created. An ABAC policy specifies one or more *claims* that need to be satisfied before a user is granted access; for example, the user must be a certain age. If the user can prove this claim, access is granted. Conditions are used when modeling Oracle Entitlements Server deployments based on ABAC. For more information, see Section 4.6, "Using the Condition Builder."

1.5.3 Java Permissions

Oracle Entitlements Server is built as an extension of Oracle Platform Security Services (OPSS). The OPSS security model is based on Java Authentication and Authorization Service (JAAS) security. JAAS institutes a permission based authorization system that implements Java-based security standards to support principal based and code based policies. A Java `Permission` object represents permission to access a resource. For example, the following code creates a `FilePermission` object representing read access to a file named `abc` in the `/tmp` directory.

```
perm = new java.io.FilePermission("/tmp/abc", "read");
```

Oracle Entitlements Server supports the Java Development Kit developer version 1.6 on either the Standard Edition or Enterprise Edition platforms. For more information, see the Java documentation at `http://www.oracle.com/technetwork/indexes/documentation/index.html`.

1.5.4 XACML

The eXtensible Access Control Markup Language (XACML) is an access control model that describes how to interpret policies, and an access control policy language (written using XML). Oracle Entitlements Server implements the XACML request and response

standard as well as the architecture model (described in Section 1.3, "Overview of the Oracle Entitlements Server Architecture.") It also implements how XACML defines policies as a collection of principals, resources, actions and attributes. For more information, see the XACML specifications at http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml.

Note: Only support for XACML 3.0 data types and functions has been added to this release of Oracle Entitlements Server.

1.5.5 PEP API

Oracle Entitlements Server has implemented the PEP API, a part of the Open Az framework (http://www.openliberty.org). The `org.openliberty.openaz.azapi` package provides access from a PEP to a remote or embedded PDP. The `org.openliberty.openaz.azapi.pep` package (PEP API) has been implemented by Oracle Entitlements Server to be used by the PEP to issue authorization requests to a PDP. More information on the Open Az API can be found at http://openaz.svn.sourceforge.net/viewvc/openaz/test/doc/index.html?org/openliberty/openaz/azapi/pep/package-summary.html. More information on the PEP API implementation can be found in *Oracle Fusion Middleware Developer's Guide for Oracle Entitlements Server*.

Note: As implemented by Oracle Entitlements Server, the PEP API do not support authorization decisions for resources protected by Java permissions.

2

Understanding the Policy Model

A *policy* is a set of criteria that, when evaluated by Oracle Entitlements Server, specifies whether a user will be granted access to a particular protected resource or assignment to a particular role. This chapter contains an overview of the Oracle Entitlements Server policy model, the elements that comprise a policy and how the elements are organized in the policy store. It contains the following sections.

- Understanding Oracle Entitlements Server Policies
- How Oracle Entitlements Server Evaluates Policies
- The Policy Object Glossary
- Implementing a Policy Use Case

2.1 Understanding Oracle Entitlements Server Policies

Oracle Entitlements Server supports the creation of the following types of policies. The referenced sections contain detailed information regarding these policy types including how they are used.

- An *Authorization Policy* defines the criteria that control access to an organization's protected resources. See Section 2.1.1, "Understanding the Authorization Policy."

  Note: Resources may include software components or business objects. For more information, see Section 2.4, "Implementing a Policy Use Case."

- A *Role Mapping Policy* defines the criteria that control how external users, groups or roles are granted or denied membership to roles created using Oracle Entitlements Server. See Section 2.1.2, "Understanding Role Assignments and the Role Mapping Policy."

2.1.1 Understanding the Authorization Policy

An *Authorization Policy* is created to grant or deny access to a particular resource based on the profile of the requesting user. From a high level, the Authorization Policy defines an association between an effect (GRANT or DENY), a principal (requesting user), the target resource, the resource's allowed actions and an optional condition. An Authorization Policy is applicable to a request for access if the parameters in the request match those specified in the policy. Consider this Authorization Policy definition:

```
GRANT the SupportManagerEast role MODIFY access to the Incidents servlet
```

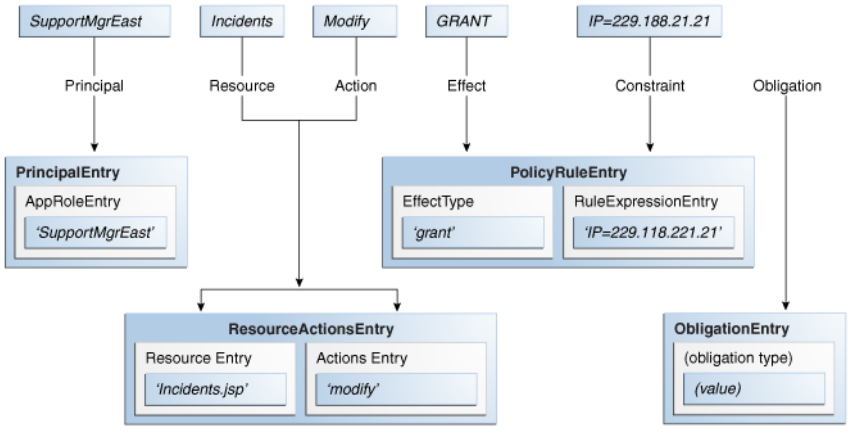
```
if the request is made from an IP address of 229.188.21.21
```

This Authorization Policy will GRANT any user that is a member of the `SupportManagerEast` role access to the Incidents servlet for the purpose of modifying it. The policy is also constrained by an optional condition - the request must be made from IP address 229.188.21.21. Thus, if the parameters in the request match the parameters in the policy (a member of the `SupportManagerEast` role wants to modify the Incidents servlet), and the request is made from IP address 229.188.21.21, the request is granted. If the parameters in the request match the parameters in the policy (a member of the `SupportManagerEast` role wants to modify the Incidents servlet) but the request is NOT made from IP address 229.188.21.21, the policy is ignored. The following list of terms and values are extracted from this policy definition and comprise the components of the Authorization Policy.

- Effect: GRANT
- Action: MODIFY
- Resource: Incidents servlet
- Principal: member of SupportManagerEast role
- Condition/Constraint: IP address 229.188.21.21

Figure 2–1 illustrates how the components of this policy map to the Oracle Entitlements Server Authorization Policy objects.

Figure 2–1 Policy Components Mapped to Authorization Policy Objects



For information on how to create, update and delete Authorization Policies, see Section 4.5.5, "Managing Authorization Policies."

2.1.2 Understanding Role Assignments and the Role Mapping Policy

An *Application Role* is a collection of users, groups, or other Application Roles; it is defined using Oracle Entitlements Server and stored in the policy store. It can be mapped to a user, group, or external role in an identity store, or another Application Role in the Oracle Entitlements Server policy store.

Note: Assigning one Application Role to another Application Role allows you to build an Application Role hierarchy.

Application Roles can be assigned to a user in either of the following ways:

- By statically granting a specific user membership in the role.
- By referencing the Application Role in a *Role Mapping Policy* that will be used to dynamically assign role membership at runtime.

A Role Mapping Policy allows you to dynamically assign (GRANT) role membership to a user or dynamically revoke (DENY) role membership from a user. Consider the following Role Mapping Policy definition:

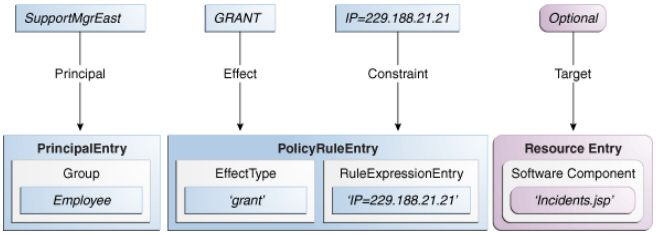
```
GRANT the Employee group application role SupportManageEast
   if the request is made from an IP address of 229.188.21.21
```

This policy grants the `SupportManagerEast` Application Role to any user that is a member of the group `Employee`. It is also constrained by an optional condition - the request must be made from IP address 229.188.21.21. Thus, if the parameters in the request match the parameters in the Role Mapping Policy (the requesting user is a member of the `Employee` group), and the request is made from IP address 229.188.21.21, the Application Role is granted. If the request is not made from the defined IP address, the Role Mapping Policy is ignored. The target is also an optional addition to a Role Mapping Policy. The following terms and values are applicable to this Role Mapping Policy definition.

- Effect: GRANT
- Application Role: SupportManagerEast
- Principal: member of Employee group
- Condition/Constraint: IP address 229.188.21.21

If a Resource or Resource Name Expression is defined as a Target for a Role Mapping Policy, the Role Mapping Policy will only be applied if it matches the resource defined in the request. Figure 2–2 illustrates how the components of this example policy might map to Oracle Entitlements Server Role Mapping Policy objects.

Figure 2–2 Policy Components Mapped to Role Mapping Policy Objects



A Role Mapping Policy can also be used to prevent specific users from being assigned an Application Role. Consider the following Role Mapping Policy definition:

```
DENY the Customers group the application role GoldCircle
   if the account balance is less then $10,000
```

This policy denies the `GoldCircle` Application Role to any members of the group Customers IF their account balance is less then $10,000. For information on how to create, update and delete Role Mapping Policies, see Section 4.5.7, "Managing Role Mapping Policies."

2.2 How Oracle Entitlements Server Evaluates Policies

During Oracle Entitlements Server runtime evaluation, the following occurs:

1. Based on the subject, a list of Application Roles is determined by:
   a. Retrieving the user's static role membership.
   b. Evaluating all applicable Role Mapping Policies with a GRANT effect and adding them to the list of roles previously determined.
   c. Evaluating all applicable Role Mapping Policies with a DENY effect and removing them from the list of roles previously determined.
2. Based on the subject and list of retrieved Application Roles, a list of Authorization Policies is evaluated to find any that might be applicable based on the grantee, target matching and conditions. (The actions allowed on the resource are defined by the Authorization Policy.)
3. A final authorization decision is based on the "DENY overrides" combining algorithm and returned to the calling application.

Section 1.4, "How Oracle Entitlements Server Processes Authorization Policies" contains additional details on this process.

2.3 The Policy Object Glossary

The policy objects defined in this section can be created, provisioned or managed using the Authorization Policy Manager Administration Console.

- **Application**

  An Application is a high-level container for managing roles, policies, resource definitions, and other policy objects; in effect, all items needed to define secure access to a particular resource. An Application may correspond to a single deployed software application, a set of deployed software applications, or components of a software application (such as an Enterprise Java Bean). You can manage more than one Application with Oracle Entitlements Server. For more information, see Section 4.5.1, "Managing Applications."

- **Application Role**

  An Application Role defines criteria for mapping users, groups, External Roles or other Application Roles to a particular Oracle Entitlements Server Application object. The Application Role can be granted to an external user, group, or role in an identity store, or another Application Role in the policy store. When creating an Application Role (in a Role Catalog) you might grant it all privileges necessary to access a given target Resource. Then, the Application Role can be assigned statically by explicitly granting membership to it, or the Application Role can be assigned dynamically by referencing it as the subject in a Role Mapping Policy. One target Application may have several different Application Roles, with each one assigned a different set of privileges offering more fine-grained access.

  One External Role can be mapped to many Application Roles. For example, the External Role `employee` (stored in LDAP-based identity store) can be mapped to the Application Role `customersupport member` (defined in one Application) and to the Application Role `IT member` (defined in another Application). A Role Catalog is the container in which Application Roles are kept. For more information, see Section 4.5.6, "Managing Application Roles in the Role Catalog."

> **Note:** An Application's Role Catalog provides the means to organize Application Roles. Search for Application Roles in the Role Catalog node of the Oracle Entitlements Server Administration Console. See Chapter 5, "Querying Security Objects" for more information.

- **Attribute**

  An *Attribute* represents data that can be used in a policy *condition*, or returned with the policy determination as an *obligation*. It can be a built-in attribute (where its value is always defined - as in current time or current user), a Resource attribute (which is associated with, and managed within, a configured Oracle Entitlements Server Resource) or a Dynamic attribute. An attribute is defined by its name, the type of data it takes as a value, and whether the value is single or multiple. An attribute value can either be passed by the protected application as part of an authorization request, or retrieved by Oracle Entitlements Server using an attribute retriever. For more information, see Section 4.5.9, "Managing Attributes and Functions as Extensions."

- **Authorization Policy**

  An Authorization Policy specifies a set of rights that an entity (the grantee, a principal or code source) is allowed on a protected resource. This might include viewing a web page or modifying a report. In short, it specifies who can do what to a resource protected by Oracle Entitlements Server. For more information, see Section 4.5.5, "Managing Authorization Policies."

  > **Note:** Search for Authorization Policies in the Default Policy Domain node (or a custom Policy Domain node, if applicable) of a configured Application. See Chapter 5, "Querying Security Objects" for more information.

- **Condition**

  A Condition is one or more boolean expressions that must evaluate to true in order for the policy to be included in the authorization decision. Adding a Condition to a policy is optional and when used, further restricts access to the protected resource. In general, a condition's boolean expression tests the value of some user, resource, or system attribute. Individual conditions can be combined with the following logical operators: AND, OR, and NOT. Conditions can define constraints based on date, time, a time range, a day of week, and so forth. For more information, see Section 4.6, "Using the Condition Builder."

- **Entitlement**

  An Entitlement (also known as a *permission set*) represents a small set of Resources and the associated actions needed to perform a task. It groups related Resources, possibly of different types, needed to perform a business function. An Entitlement is a reusable collection of access rights that can be granted to multiple principals. For more information, see Section 4.5.4, "Managing Entitlements."

- **External Role**

  An External Role is defined in an external identity store such as an LDAP server or a database. The term *external role* is often synonymous with the terms *enterprise role* or *enterprise group*; they are typically implemented as LDAP groups in the identity store. For information on adding an External Role to a policy, see Section 4.5.5.1,

"Creating an Authorization Policy" or Section 4.5.7.1, "Creating a Role Mapping Policy."

Note: Within Oracle Entitlements Server, external roles and users are read-only. They are managed with a different tool, such as Oracle Identity Manager. For more information, see *Oracle Fusion Middleware System Administrator's Guide for Oracle Identity Manager*.

- **Function**

  A *Function* represents code that can be invoked as part of the evaluation of a policy condition; the returned value will affect the evaluation of the condition. You can extend the functions supplied with Oracle Entitlements Server by creating additional, custom ones. For more information, see Section 4.5.9, "Managing Attributes and Functions as Extensions."

- **Obligation**

  An Obligation specifies optional information that is to be returned with a GRANT or DENY authorization decision. When used in a policy, an Obligation may impose an additional requirement for the policy enforcing component, or simply request useful information. For example, the reason a request for access has been denied might be returned as an Obligation. For information on adding an Obligation to a policy, see Section 4.5.5.1, "Creating an Authorization Policy."

- **Policy Domain**

  A Policy Domain is a container under an Application object that can serve as a partition to facilitate management of Resources, Entitlements and Authorization Policies. The Policy Domain is an optional management construct that can restrict an administrator's right to a particular subset of Resource, Entitlements, and Authorization Policies. The Policy Domain has no effect upon runtime policy evaluation. Multiple Policy Domains can be created and can be hierarchical. A *default policy domain* is added to each Application upon its creation. For more information, see Section 11.4, "Using Policy Domains to Delegate."

- **Policy Store**

  The Policy Store is where all Oracle Entitlements Server policy objects (including, but not limited to, Applications, Resources and various role types) are stored. A Policy Store can be a relational database (preferred) or an LDAP-based directory. For more information, see Section 3.2.3, "Accessing the Policy Store."

- **Principal**

  A Principal is the identity to which the access rights defined in a policy are granted. A principal can be a user, a group, an External Role, or an Application Role. Most frequently, it is an Application Role. For information on adding a principal to an Oracle Entitlements Server policy, see Section 4.5.5.1, "Creating an Authorization Policy" or Section 4.5.7.1, "Creating a Role Mapping Policy."

- **Resource**

  A Resource is a protected component or object to which access is granted or denied. A Resource represents the application component or business object that is secured by an Authorization Policy. At runtime, the application passes the Resource name to check for access definitions that will determine whether a Principal is authorized access. A Resource requires an associated Resource Type. For more information, see Section 4.5.3, "Managing Resources."

- **Resource Type**

  A Resource Type represents the type of a secured object. Protected software application components that share common characteristics can be represented by particular Resource Type. For example, a set of pages can be represented by one Resource Type and bank accounts by another Resource Type. A Resource Type defines eligible *resource attributes* and possible valid actions that are applicable to the protected component. For more information, see Section 4.5.2, "Managing Resource Types."

- **Role Category**

  A Role Category is an optional tag that can be associated with an Application Role; it can be used for searching. Role Categories allow administrators to organize roles in arbitrary flat collections. They have no effect upon runtime policy evaluation. For more information, see Section 4.5.8, "Managing a Role Category."

- **Role Mapping Policy**

  A Role Mapping Policy is used to determine what external subjects (users, groups or External Roles) are assigned to the applicable Application Role. The Application Role, when referenced in an Authorization Policy, defines the principals affected by the Authorization Policy. Role Mapping Policies may also include conditions. For more information, see Section 4.5.7, "Managing Role Mapping Policies."

> **Note:** Search for Role Mapping Policies under the Role Catalog node of the Oracle Entitlements Server Administration Console. See Chapter 5, "Querying Security Objects" for more information.

2.4 Implementing a Policy Use Case

Oracle Entitlements Server provides the ability to externalize policy management and policy decision making logic from an organization's resources. It secures access to the organization's resources by implementing policies that specify the users, groups, and roles that can access them. Resources can be application software components (URLs, Enterprise JavaBeans, JavaServer Pages) or enterprise business objects (customer accounts, patient records). This use case considers how the policy model can be used to secure the financial services offered by Acme Investment Bank. It is based on the concept of *hierarchical resources* - resources are organized as a tree and inherit from their parent elements.

> **Note:** Oracle Entitlements Server also supports the concept of non-hierarchical (flat) resources. See Section 4.5.2, "Managing Resource Types" for more information.

In this use case, the following conditions apply:

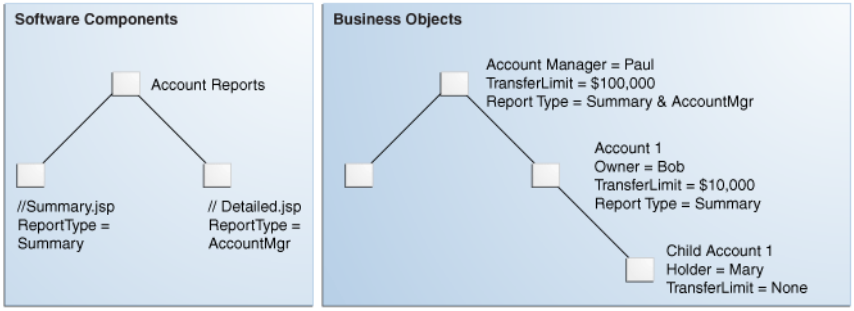
- A customer may open a family account and is considered an owner of the account.
- The customer may open a child account for family members and set transfer limits for each member. Transfer limits must be lower than the customer's own transfer limit.
- Each account has a bank employee that acts as an account manager and sets the transfer limits for the account.

- Both a Summary report and an Account Manager report are associated with each family account.

Figure 2–3 illustrates the financial services scenario by organizing the protected resources into business objects and software components. Paul is a bank employee who is an account manager and can set transfer limits up to $100,000 on the accounts that he manages. Paul manages account owner Bob's family account which has a transfer limit of $10,000. Bob manages his family account and a child account he created for Mary who may not transfer money. These accounts are considered business objects and are protected as such.

Account owner Bob has access to a Summary report generated for his family account. Account manager Paul has access to Bob's Summary report and his own Account Manager report. Child account holder Mary has access to no reports. These reports, generated as JavaServer Pages, are considered software components and are protected as such.

Figure 2–3 Use Case for Software Components and Business Objects



For any given user, a request for access to a financial services account (business object) or account report (software component) generates a decision based on the following questions:

1. Is this user an account holder, account owner or an account manager?
2. Can this user transfer funds from this account (subject to the role, transfer limit, and time of transaction)?
3. What reports can this user access?

The first two questions can be decided using policies created to protect business objects, while the last can be decided using policies created to protect software components. The following sections illustrate how to conceptualize the policies.

- Section 2.4.1, "Protecting Software Components"
- Section 2.4.2, "Protecting Business Objects"

2.4.1 Protecting Software Components

The Account Reports node in Figure 2–3, "Use Case for Software Components and Business Objects" represents the reporting application. `Summary.jsp` and `Detailed.jsp` are the software components. There are several options from which to choose when deciding how to model policies for securing these software components. One option is to set an Authorization Policy for the top node reporting application by using group membership. The following example illustrates how access is explicitly granted by naming the resource and the group of users that can access it.

```
GRANT the BankManagers group access to the AccountReports node
```

This top down Authorization Policy grants access to the Account Reports node for anyone in the BankManagers group. Because these resources are hierarchical, anyone in the allowed group has access to both the Summary and Detailed reports. But this access may be restricted using system-based or attribute-based conditions. For example, adding a condition based on time or based on the value of a specific user, group, or resource attribute would further limit access. The following example illustrates how a time-based condition restricts access of the reports to typical office hours.

```
GRANT the BankManagers group access to the AccountReports node
  IF the request is made between 09:00 and 17:00
```

Another option can set the top down Authorization Policy by defining the principal as a role rather than a group. A role is comprised of users or groups. (An LDAP role would be granted enterprise wide whereas an Application Role is specific to the Application for which it was configured.)

> **Note:** A user can be assigned to a role through membership or a Role Mapping Policy as discussed in Section 2.1, "Understanding Oracle Entitlements Server Policies."

In the following example, the resource is explicitly named and access is implicitly granted to a user if the user is assigned the defined role.

```
GRANT access to AccountReports node if user has BankManagers role
```

You can also dynamically assign the BankManagers role to users accessing the reporting application if they are a member of the BankManagers group (as illustrated below).

```
GRANT BankManagers role to members of BankManagers group
  FOR access to AccountReports node
```

Another way to define the previous Authorization Policy is to assign the role based on a user attribute value rather than group membership. (In a large enterprise, it is typically more efficient to assign users based on attributes than on group membership.) The following example assigns the BankManagers role to the requesting user if the value of the UserType attribute in the user's profile is BankManager.

```
GRANT BankManagers role to anyone defined by UserType 'BankManager'
  FOR access to the AccountReports node
```

The previous examples represent Authorization Policies that scope from the top node reporting application down to the reports but Authorization Policies can also be defined for the specific report nodes. The following example grants access to the `Summary.jsp` report to all assignees of the BankManagers and AccountOwners roles. The additional condition is that the principal requesting access must be listed on the account to which the report pertains.

```
GRANT Summary.jsp access to all members of BankManagers and AccountOwners role
  IF the requesting assignee is listed as OWNER or MANAGER on specified account
```

Another example illustrates how access to the `Detailed.jsp` report can be granted to anyone who is assigned the BankManager role.

```
GRANT Detailed.jsp access to all assignees of BankManagers role
```

The previous examples show how an Authorization Policy can be modeled for specific application software components. In a real enterprise scenario, each application may have tens or hundreds of resources so it might not be practical to write an Authorization Policy for each one. The concept of *resource attributes* has been implemented by Oracle Entitlements Server to address this proliferation of application software component resources and associated Authorization Policies.

By associating a resource with an attribute, you can grant access based on the value of the attribute. For example, *filetype* could be a resource attribute that is used to define an HTML page, an image, or a PDF. By defining a condition as `if filetype=pdf`, you can grant access to all PDF files that are associated with the resource. The following example uses a resource attribute; it allows users assigned the BankManagers and AccountOwners role access to all reports although access is granted only if the report type being requested matches the value of the UserReportType attribute in the specific user's profile.

```
GRANT users assigned BankManagers or AccountOwners roles
     access to AccountReports
   IF requested ReportType matches UserReportType attribute value
     in user profile
```

This policy grants BankManagers and AccountOwners access to all reports although access is constrained based on matching resource attribute values with user attribute values. An advantage of this approach is that the policy governing access need not change as resources are added to, or removed from, an application. As resources change, the ReportType resource attribute attached to the application continues to govern access.

2.4.2 Protecting Business Objects

There are several options from which to choose when deciding how to model Authorization Policies for securing business objects. In this banking scenario, business objects are bank accounts. Figure 2–3, "Use Case for Software Components and Business Objects" illustrates the Acme bank account structure.

Each bank account can have a manager, an owner, and a holder with each *scope* assigned a certain set of privileges (or *entitlements*). The policy evaluating what a user can do on the bank account is then based on the user's attributes rather than the resource. The following example allows anyone to transfer money but that privilege is only granted if the user is defined as owner of the account requested and the amount of money being transferred is less than or equal to the limit defined for the user.

```
GRANT anyone transfer privileges only
   IF the user is listed as OWNER on specified account
  AND transfer amount is equal to or less than the transfer limit
```

There is another option to acquire a user's entitlements. Rather than comparing a transfer request to a transfer limit, Oracle Entitlements Server can return the transfer limit amount as the output of evaluation. In this scenario, the user's ability to access the account is verified but the transfer amount is returned to the caller (in a Java object) as an *obligation*. This leaves verification that the transfer amount is within the transfer limit up to the application. The following example illustrates this model.

```
GRANT anyone transfer privileges only
   IF the user is listed as OWNER on specified account
  THEN RETURN transfer limit to calling application
```

A model where the bank account corresponds to an individual resource instance can also be used; however, this would yield a proliferation of policies (one for each account) and become unmanageable. For example, if Acme Investment Bank had 100,000 accounts, it would need at least 100,000 policies just to manage transfers. For more information on adding obligations, see Section 4.5.5, "Managing Authorization Policies."

3 Getting Started

This chapter describes how to get started using Oracle Entitlements Server, including information about how to use and navigate the graphical interface. It contains the following sections.

- Before You Begin
- Understanding The Graphical Interface
- Accessing the Administration Console
- Navigating the Administration Console
- Upgrading from Oracle Entitlements Server Basic
- Accessing Oracle Entitlements Server Examples

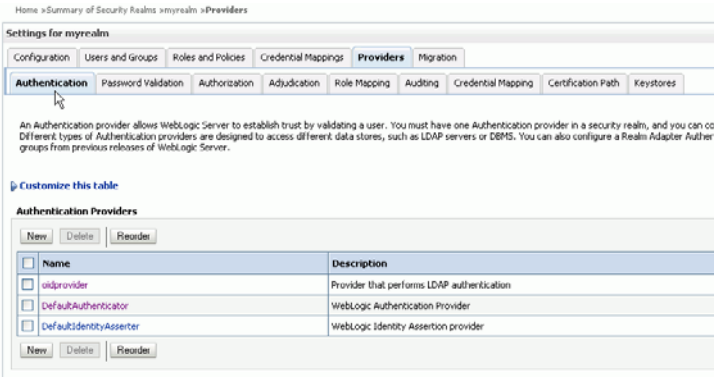
3.1 Before You Begin

Before getting started using Oracle Entitlements Server, the following tasks must be done. They include installing the product and its components (for example, remote Security Modules), and configuring features like high availability and Secure Sockets Layer (SSL), if applicable. After finishing with these tasks, you can begin with Section 3.2, "Understanding The Graphical Interface."

- Install and configure Oracle Entitlements Server according to the instructions in *Oracle Fusion Middleware Installation Guide for Oracle Identity and Access Management*.
  - The policy store managed by Oracle Entitlements Server must be a relational database.
  - The identity store associated with Oracle Entitlements Server must be an LDAP-based directory.
- After installation, the Oracle Entitlements Server identity store is associated with the WebLogic Server embedded LDAP directory. While this embedded LDAP directory is fine for development purposes, a supported LDAP directory must be used in production. The following procedure reconfigures the default identity store settings. More specific information on configuring LDAP authentication providers can be found in the *Oracle Fusion Middleware Securing Oracle WebLogic Server*.
  1. Launch the WebLogic Server console.
  2. Click Security Realms.
  3. Click the settings for *myrealm*.

4. Click the Provider tab.
5. Click the Authentication tab as displayed in Figure 3–1.

Figure 3–1 The Authentication Provider Tab

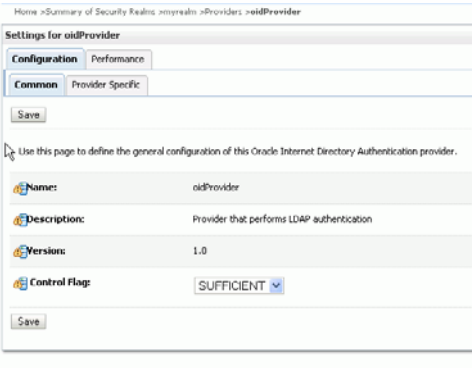


6. Click the New button to create a new provider.
7. Enter a name and select the type of LDAP-based directory.

   For example, *OracleInternetDirectoryAuthenticator*.
8. Configure the provider-specific attributes of the LDAP-based directory.

   This might include the host name and port, credentials, group search base, user search base and the like.
9. Save the provider information.
10. Change the order of the providers so that the LDAP-based directory is first.
11. Click the new provider name to configure it.
    a. Click the Configuration tab.
    b. Click the Common tab.
    c. Set the Control Flag to SUFFICIENT and click Save as displayed in Figure 3–2.

Figure 3–2 SUFFICIENT Control Flag

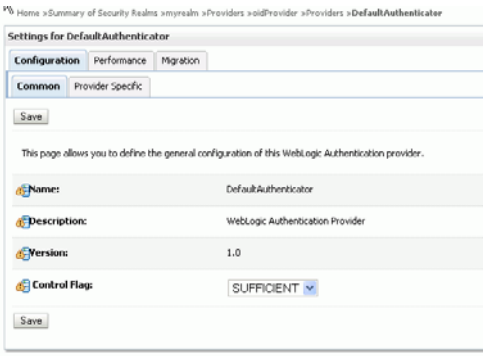


    d. Click the Provider Specific tab.

e. Enter the LDAP configuration information for your identity store and click Save.

12. Return to the Providers tab.
13. Click *DefaultAuthenticator* to change its configuration.
14. Set the Control Flag to SUFFICIENT and click Save as displayed in Figure 3–3.

Figure 3–3 DefaultAuthentciator Tab in WebLogic Server Console



15. Restart WebLogic Server.

- For information about configuring high availability for Oracle Entitlements Server, see *Oracle Fusion Middleware High Availability Guide*
- For information regarding the authentication of users, see *Oracle Fusion Middleware Securing Oracle WebLogic Server*.

> **Note:** Oracle Entitlements Server is not involved in the authentication of users. This is normally done as part of the WebLogic Server security realm configuration.

- For information about configuring one-way SSL for connections that Oracle Entitlements Server establishes with the policy store, the identity store, and the database, see *Oracle Fusion Middleware Securing Oracle WebLogic Server*. Access to Oracle Entitlements Server using a browser can also be secured through one-way SSL. These settings are similar to those of any other application running in the Oracle WebLogic Server.
- Refer to the system requirements and certification documentation for information about hardware and software requirements, platforms, databases, and other information.
  - The system requirements document covers information such as hardware and software requirements, minimum disk space and memory requirements, and required system libraries, packages, or patches:

    `http://www.oracle.com/technology/software/products/ias/files/fusion_requirements.htm`
  - The certification document covers supported installation types, platforms, operating systems, databases, JDKs, and third-party products:

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

These documents are available on Oracle Technology Network (OTN).

3.2 Understanding The Graphical Interface

Oracle Authorization Policy Manager is a sub-component of Oracle Entitlements Server that is the graphical management tool for administrators. It is a browser-based interface for managing policies and related policy objects. The following sections contain information to help understand the Authorization Policy Manager Administration Console.

- Section 3.2.1, "Assigning Oracle Entitlements Server Administrators"
- Section 3.2.2, "Using the Identity Store"
- Section 3.2.3, "Accessing the Policy Store"
- Section 3.2.4, "Displaying Oracle Platform Security Services Application Grants"

3.2.1 Assigning Oracle Entitlements Server Administrators

Only users with sufficient privileges can log in to the Oracle Entitlements Server Administration Console or use administrative command-line tools such as the WebLogic Scripting Tool (WLST). An Oracle Entitlements Server system-level Administrator Role named `SystemAdmin` is created during installation and is mapped to the WebLogic Server administrator user (`weblogic`). The password is set during installation. `SystemAdmin` has extensive privileges that includes the rights to create additional Administrative Roles and delegating administrative rights to others.

Note: At first log in to the Oracle Entitlements Server Administration Console, `SystemAdmin` must use the credentials set during installation. The identifier and password can be changed by using your identity store's management tool.

You can create separate administrative users with different access rights for administering Oracle Entitlements Server and your environment. For more information, see Section 11.6, "Managing System Administrators Using Administrator Roles.".

3.2.2 Using the Identity Store

Oracle Entitlements Server administrator and user identities are stored in an identity store, typically an LDAP directory server. Users and external roles defined in the identity store are read-only; Oracle Entitlements Server reads and displays the data but can not perform management operations on it. Management of the identity data is accomplished using the identity store's tools or an identity management product such as Oracle Identity Manager.

Note: After installation, the Oracle Entitlements Server identity store is associated with the WebLogic Server embedded LDAP directory. See Section 3.1, "Before You Begin" for the procedure to reconfigure Oracle Entitlements Server to use a supported LDAP directory.

Oracle Entitlements Server will use the Oracle Platform Security Services (OPSS) Identity Store Service to allow for the configuration and support of multiple identity

stores. The Identity Store Service allows access to, and managment of, multiple identity types (users, groups, roles) and is extensible to support new identity types. For information on the Identity Store Service, see the *Oracle Fusion Middleware Application Security Guide*. See Section 10.3, "Configuring Identity Directory Service Profiles" for details specific to Oracle Entitlements Server.

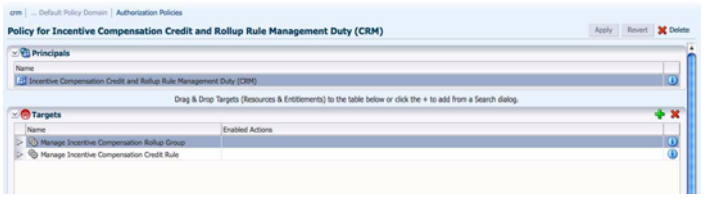
3.2.3 Accessing the Policy Store

The Policy Store used to maintain security objects and defined policies must be a relational database. (Oracle Internet Directory can be used as the policy store but has limited capabilities.) For links regarding hardware requirements, see Section 3.1, "Before You Begin." Basic information about the Policy Store can be found in the *Oracle Fusion Middleware Application Security Guide*.

3.2.4 Displaying Oracle Platform Security Services Application Grants

The Administration Console displays advanced Authorization Policies created using Oracle Entitlements Server as well as the simpler Application Grants (application policies) created using Oracle Platform Security Services (OPSS). The OPSS Application Grants can be displayed for viewing, modification and deletion only. When created using OPSS, Application Grants are defined with a principal and target only. Figure 3–4 is a screenshot of the Oracle Entitlements Server screen when an OPSS Application Grant is displayed using one of the search functions.

Figure 3–4 OPSS Application Grants Display Screen



Note the Name, Display Name and Description fields are not displayed as they would be if the Authorization Policy was created using Oracle Entitlements Server. OPSS Application Grants can only be removed or modified using Oracle Entitlements Server; they can not be created using Oracle Entitlements Server. For more information on Application Grants, see the *Oracle Fusion Middleware Application Security Guide*.

3.3 Accessing the Administration Console

The following sections contain information on the Authorization Policy Manager graphical interface (also referred to as the Administration Console).

- Section 3.3.1, "Signing In to the Administration Console"
- Section 3.3.2, "Signing Out of the Administration Console"

3.3.1 Signing In to the Administration Console

Follow this procedure to sign in to the Authorization Policy Manager Administration Console.

1. Enter the Authorization Policy Manager Administration Console URL in the address bar of your browser. For example:

https://*hostname:port*/apm/

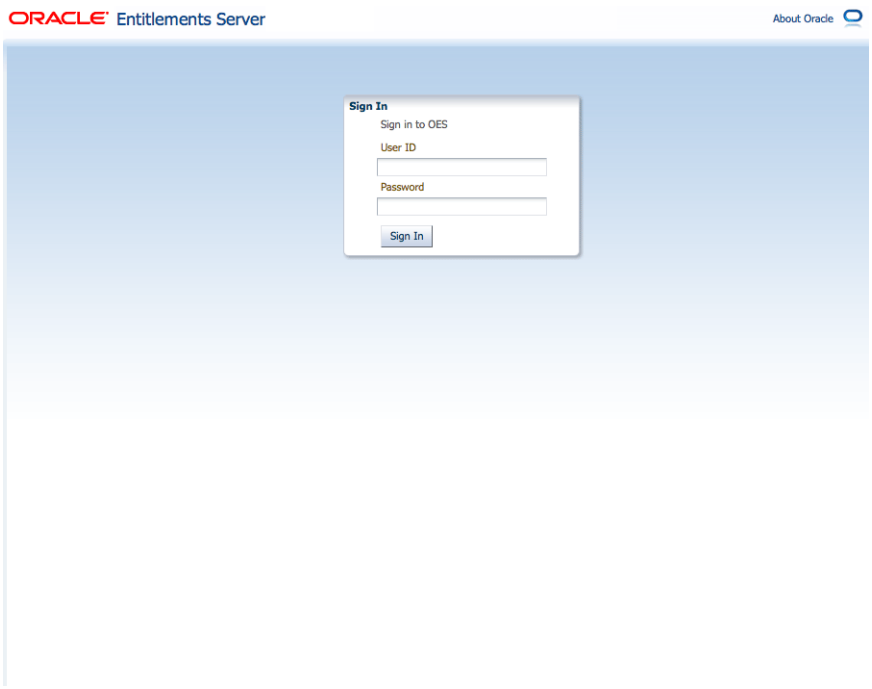
where:

- `https` represents the Hypertext Transfer Protocol (HTTP) with Secure Socket Layer (SSL) enabled to encrypt and decrypt user page requests and the pages returned by the Web server.
- *hostname* refers to the fully qualified domain name of the computer hosting the Oracle Authorization Policy Manager Administration Console.
- *port* refers to the designated bind port for the Authorization Policy Manager Administration Console. (This is the same as the bind port for the WebLogic Server Administration Console.)
- `/apm/` refers to the context-root of the Authorization Policy Manager application which redirects to the login page.

2. Enter the System Administrator credentials.

   The default system administrator identifier is `weblogic`. The password is the same one supplied during installation. Figure 3–5 is a screenshot of the Sign In page.

Figure 3–5 Administration Console Sign In Page



3. Click **Sign In**.

3.3.2 Signing Out of the Administration Console

Follow this procedure to sign out of the Authorization Policy Manager Administration Console.

1. Click the **Sign Out** link located in the upper right corner of the Administration Console.

   Figure 3–6 is a screenshot of the Sign Out link.

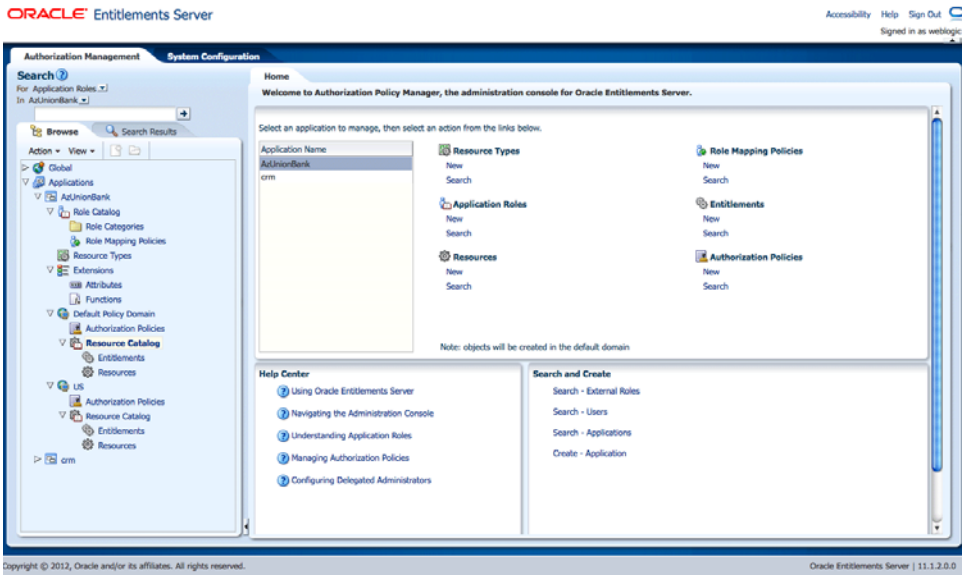
Figure 3–6 Administration Console Sign Out Link

2. Close the browser window.

3.4 Navigating the Administration Console

After a successful log in, the Authorization Policy Manager Administration Console is displayed with the Authorization Management Tab active. The Navigation Panel is on the left side and the Home tab is displayed on the right side. Objects selected in the Navigation Panel are opened and displayed in new tabs next to the Home tab on the right. Figure 3–7 is a screenshot of the Administration Console after an administrative user has successfully signed in.

Figure 3–7 Oracle Entitlements Server Administration Console



The following list contains descriptions of the top-level items displayed in Figure 3–7. See the appropriate links for more information.

- Section 3.4.1, "Understanding the Main Tabs"
- Section 3.4.2, "Using The Navigation Panel"
- Section 3.4.3, "Using the Home Tab"
- Section 3.4.4, "Accessing Help"

3.4.1 Understanding the Main Tabs

See the following sections for information on the organizational tabs used in the Administration Console. Each tab is comprised of a Navigation Panel and Home area.

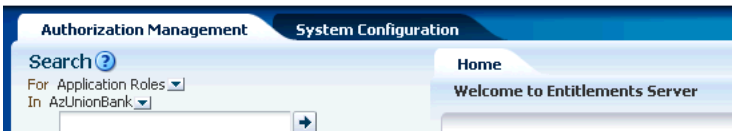
- Section 3.4.1.1, "Authorization Management Tab"

- Section 3.4.1.2, "System Configuration Tab"

3.4.1.1 Authorization Management Tab

The Authorization Management tab is used to search and manage policy objects. This tab is active upon successful log in to the Administration Console. Figure 3–8 is a screenshot of the Authorization Management tab.

Figure 3–8 Authorization Management Tab

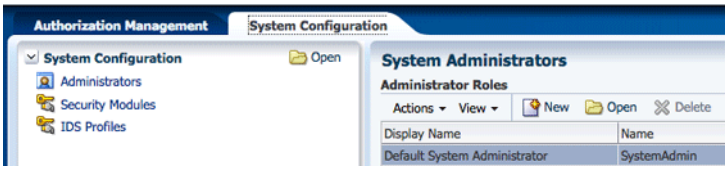


Under Authorization Management, the left side is the Navigation Panel and the right side is Home. The Home display changes based on what is selected from the Navigation Panel. For more information, see Section 3.4.2, "Using The Navigation Panel" and Section 3.4.3, "Using the Home Tab."

3.4.1.2 System Configuration Tab

The System Configuration tab is used to manage administrative and system type objects for the Oracle Entitlements Server deployment. Figure 3–9 is a screenshot of an active System Configuration tab. The object selected in the Navigation Panel is displayed using tabs in the Home area.

Figure 3–9 System Configuration Tab



The following tasks are performed under System Configuration:

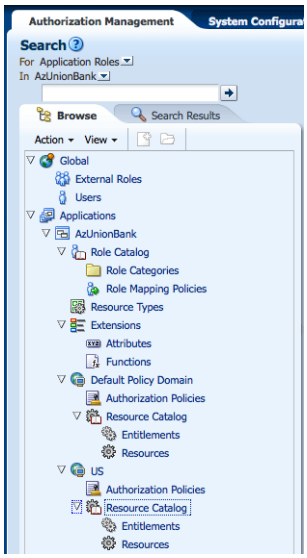
- Creating Security Module definitions and binding them to Applications
- Managing system administrators (for example, creating additional system administrator roles, assigning users to system administrator roles, and assigning rights to system administrator roles)
- Managing Identity Store profiles

For more information, see Chapter 10, "Managing System Configurations".

3.4.2 Using The Navigation Panel

The Navigation Panel is used to find security objects by browsing the Global or Applications information trees, or by conducting a simple search. It lists all Global and Application policy objects in a navigatable tree. You can browse the tree or display objects as Search Results based on defined search criteria. Figure 3–7 is a screenshot that displays the Navigation Panel with its nodes collapsed. Figure 3–10 displays the Navigation Panel with its nodes expanded and many policy objects in view.

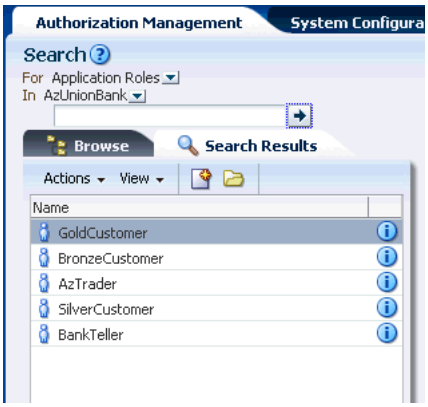
Figure 3–10 Navigation Panel Browse Tab With Nodes Expanded



The Navigation Panel contains, from top to bottom, the following elements:

- A pull-down list to select the policy object for a simple search. For more information, see Section 5.2, "Finding Objects with a Simple Search."
- A pull-down list to select the scope of a simple search. For more information, see Section 5.2, "Finding Objects with a Simple Search."
- A text box to enter the simple search string. The string is compared against both the Name and Display Name of policy objects; those that match are displayed in the Search Results tab.
- The **Browse** tab displays the following expandable and collapsible nodes:
  - The **Global** node collects global objects such as external roles and users.
  - The **Applications** node contains one or more Applications being managed by the administrator that is logged in. (Only Applications which the logged in user is authorized to access are displayed.) From any of those displayed, the administrator can access application-specific policy objects such as resource types, entitlements, resources, policies, and roles. For more information, see Chapter 10, "Managing System Configurations".
- The **Search Results** tab displays the results of the last simple search as seen in Figure 3–11.
- Action and View drop downs to select operations on the chosen policy object.

Figure 3–11 Navigation Panel Search Tab



From the Navigation Panel, there are two methods for displaying the **New** and **Open** options comprised in the Actions drop-down list.

- Locate the desired Application, expand the node, and select the desired object. Click the Actions drop-down menu and select New or Open.
- Locate the desired Application, expand the node, right-click the desired object and select New or Open.

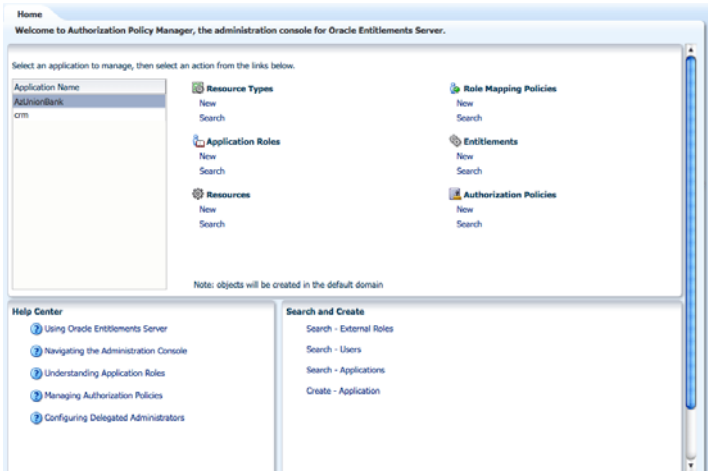
Select **New** to create a new object of the same type and select **Open** to display a search tab in the Home area. Double-clicking an object from the node also opens a Search tab in the Home area.

3.4.3 Using the Home Tab

The Home tab displays on the right side of the Navigation Panel and contains quick access links to New and Search screens for the most commonly used policy objects. As displayed in Figure 3–12, the Home tab of the Administration Console is divided into the following sections.

- The **Application** area is the upper region of the Home tab. The Application Name pane displays all applications available to the logged in user. To the right of this pane are links to screens for performing common operations such as creating new policy objects (entitlements, resources, resource types, application roles, and authorization policies) or searching defined policy objects.
- The **Global** section is the lower right region of the Home tab. This section is for objects shared across all applications and includes external role search.
- The **Entitlements Resource Center** section is the lower left region of the Home tab. It contains links to information regarding the most commonly used procedures.

Figure 3–12 The Home Tab



3.4.4 Accessing Help

To get more information while using the Administration Console, click the Help link located in the upper right corner (as seen in Figure 3–6). A separate window opens. From this window you can access both the online help and an embedded version of this book in HTML. After the window displays, select either Oracle Entitlements Server Administration Console *Online Help* or *Administrator's Guide for* Oracle Entitlements Server from the drop-down Book list. The help topics link to the corresponding section of the embedded book as do the links in the book's Table of Contents.

3.5 Upgrading from Oracle Entitlements Server Basic

Oracle Entitlements Server can be licensed separately from other Oracle products with an agreement that defines usage restrictions. Oracle Entitlements Server Basic replaces the embedded authorization engine within Oracle Platform Security Services (OPSS) and is used to define, enforce and audit basic Role Based Access Control and Java/JAAS permission based authorization policies. A Basic license allows the use of a limited set of features provided by Oracle Entitlements Server and the Oracle Entitlements Server Security Modules, and is included and available for use only with Oracle products that list this component in their respective licensing documentation. More information is in the *Oracle Fusion Middleware Licensing Information*.

In the simplest scenario, nothing other than licensed Oracle Fusion Middleware or Fusion Applications technologies that include Oracle Entitlements Server Basic need be installed. Oracle Fusion Middleware products that include Oracle Entitlements Server Basic contain:

- The Oracle Entitlements Server authorization engine (Policy Decision Point)
- A default policy store (not supported for production environments) is configured based on the `system-jazn.xml` file. Customers must configure either the Oracle Relational Database Management System or Oracle Internet Directory based policy store for production environments.

The Oracle Entitlements Server Administration Console (Authorization Policy Manager) can optionally be installed to provide centralized policy management.

3.5.1 Changing From Basic to Advanced Policy Authorization

When Authorization Policy Manager is installed for Oracle Fusion Applications, it is configured to allow for Oracle Entitlements Server Basic authorization. Basic authorization is based on the permissions policy model in which permissions are granted to users, groups, and code sources. For users and groups, the permissions determine what a user or a group member is allowed to access. For code sources, they determine what actions the code is allowed to perform. Advanced authorization allows for the use of more fine-grained policy objects including Role Mapping Policies and hierarchical resources.

Use the following procedure to reconfigure Authorization Policy Manager for advanced authorization. It assumes WebLogic Server is installed and the Fusion Middleware home directory is available.

1. Change to the Fusion Middleware home directory at `$FMW_HOME/oracle_common/common/bin`.
2. Connect to the WebLogic Server using the `wlst.sh` command.

   ```
   connect ('weblogic','weblogic1','t3://localhost:7101')
   ```

   The command takes the following arguments: user name (weblogic), password associated with the user (weblogic1) and the T3 connection URL for the Administration Server. In this example, it is running locally on port 7101.
3. Export the Authorization Policy Manager configuration file using the `exportMetadata` command.

   ```
   exportMetadata(application='oracle.security.apm', server='AdminServer',
    toLocation='/tmp/repository/',
    docs='/oracle/security/apm/config/apm-config.xml')
   ```

   This command will export the `apm-config.xml` configuration file to the `/tmp/repository` sub-directory on the machine that hosts the Administration Server. The command takes the following arguments:

   - The application owner of the document being exported; in this case, the default value is `oracle.security.apm`.
   - The name of the WebLogic Server Administration Server.
   - The directory to which `apm-config.xml` will be exported. Be sure you have access rights to this directory.
   - The document being exported; in this example, `oracle/security/apm/config/apm-config.xml` will be exported. , so please make sure you have access right to that file path in the AdminServer machine once you downloaded the documents
4. Open the `apm-config.xml` configuration file in a text editor.

   The file is in the `/tmp/repository` directory as previously specified.
5. Change the value of the `oracle.security.apm.oes.mode` attribute in this file from *basic* to *advanced*.
6. Save the changes and close the file.
7. Upload the modified file back to the repository.

   ```
   importMetadata(application='oracle.security.apm', server='AdminServer',
    fromLocation='/tmp/repository/',
    docs='/oracle/security/apm/config/apm-config.xml')
   ```

This command will import the `apm-config.xml` configuration file back to the machine that hosts the Administration Server. The command takes the following arguments:

- The application owner of the document being imported; in this case, the default value is `oracle.security.apm`.
- The name of the WebLogic Server Administration Server.
- The directory from which `apm-config.xml` will be imported.
- The document being imported; in this example, `apm-config.xml` will be imported to the oracle/security/apm/config/ directory.

8. Issue the exit command.

```
exit()
```

3.6 Accessing Oracle Entitlements Server Examples

http://www.oracle.com/technetwork/indexes/samplecode/index.html is the repository on Oracle Technology Network from which developer and administration examples for Oracle Entitlements Server can be accessed.

4

Managing Policies and Policy Objects

The Oracle Entitlements Server Administration Console is used to manage Authorization and Role Mapping Policies, and the security objects from which they are created. This chapter contains the following sections.

4.1 Introducing Policy and Policy Object Management

Oracle Entitlements Server allows administrators to perform create, read, update, and delete operations on all policy objects. This can be done in any of the following ways:

- Using the Administration Console (as described in this book)
- Using the Management API (as described in the *Oracle Fusion Middleware Developer's Guide for Oracle Entitlements Server*)
- Using the WebLogic Scripting Tool on the command line (as described in the *Oracle Fusion Middleware Developer's Guide for Oracle Entitlements Server*)

Authorization Management tasks performed in the Administration Console typically require that an administrator identify an object (by browsing or searching), select it, and choose one of the operations available for it. Objects are organized in the main groupings displayed in the Navigation Panel.

- **Application** objects include the objects used to create Role Mapping and Authorization Policies (Resources, Application Roles and the like). They apply to, and can only be used for Authorization Policies within, the Application under which they are defined. The Applications node in the Navigation Panel is the branch under which all configured Applications (and their respective objects) are organized. This chapter contains information on managing Applications and their child objects.
- **Global** objects include users and External Roles. These objects may apply to all configured Applications throughout the system. The Global node in the Navigation Panel is the branch under which all systemwide objects are organized.

> **Note:** Within Oracle Entitlements Server, External Roles (and users) are read only and managed with a tool such as Oracle Identity Manager. For more information, see *Oracle Fusion Middleware System Administrator's Guide for Oracle Identity Manager*.

System Configuration tasks performed in the Administration Console typically include Security Module profile definitions and bindings, system administrator configurations, and identity store profile definitions and bindings. These objects are discussed in Chapter 10, "Managing System Configurations."

> **Note:** Oracle Entitlements Server supports the mapping of policies to Users, External Roles, and Application Roles. However, mapping policies to Application Roles is recommended because managing authorization based on grants to individual Users and External Roles can become unmanageable as the number increases. Additionally, if the identity store changes (for example, when a move between development, test and production environments results in a new LDAP server), no changes to policy definitions are needed. All that is required is a re-mapping of the Application Roles to the Users and External Roles available in the target environment.

By default, all access to a resource is denied until an Authorization Policy is written and deployed that explicitly grants access action. If the Authorization Policy only grants an entitlement on a Resource to a role, the user must be statically assigned to it or a Role Mapping Policy must be written and deployed that assigns a user or a group to the defined role. If an Authorization Policy denies a previously granted entitlement, it takes precedence over the grant. Explicit DENY authorization policies cannot be overruled. A practical use of a DENY policy is to explicitly deny an entitlement to ensure that a user or group can never gain access to a specific resource.

4.2 Defining an Authorization Policy And Its Components

Defining a policy requires that the objects be created in a particular order. For example, a Resource can only be created after defining a Resource Type. A policy can be composed by following the sequence described below.

1. Create an Application.

   In the Navigation Panel, an Application should be created as the overall container for policies and related information that secure the components of a particular resource. You may create as many Applications as needed although it is recommended that only one is created for each application to be secured. For more information, see Section 4.5.1, "Managing Applications."

2. Create a Resource Type.

   A Resource Type specifies one or more Resource attributes, and definitions of all possible valid actions that can be performed on a particular kind of resource.

   > **Note:** Before an attribute can be added to a Resource Type it must be defined as documented in Section 4.5.9, "Managing Attributes and Functions as Extensions."

The actions can be standard actions (GET and POST to a URL) or custom actions on a business object (transfer money to or from a bank account). Consider the following Resource Types and their valid actions:

- A file may support Read, Write, Copy, Edit, and Delete actions, depending on its type. A `clearance` Resource attribute may be associated with the file to define who can see it. It might do this by defining a security level that must be held by the subject for access; values might include `unclassified`, `restricted`, `confidential`, or `topSecret`. For example, if the clearance attribute has a value of `topSecret`, an Authorization Policy can be created to grant access by adding the condition: `if securityLevel = topSecret`.
- A checking account application may support deposit, withdrawal, view account balance, view account history, transfer to savings, and transfer from savings.

Resource instances are created from Resource Types. Actions defined by the Resource Type are granted or denied when accessing a protected Resource instance created from the Resource Type.

Note: A Resource instance is defined in a Policy Domain and references the Resource Type. For more information, see Section 4.5.3, "Managing Resources."

For more information, see Section 4.5.2, "Managing Resource Types."

3. Instantiate a Resource from the Resource Type.

   A specific protected target (Resource) is instantiated from a Resource Type. A Resource represents a secured target (for example, an application) and is created under a Policy Domain in the Resource Catalog.

   Note: A Policy Domain is an optional object that is created for purposes of delegated administration and organization. If no Policy Domain is specified, the Resource instance is created under the Default Policy Domain. See Chapter 11, "Delegating With Administrator Roles."

   It is not necessary to create a Resource instance for each protected resource. A Resource Name Expression can also be used in an Authorization Policy. A Resource instance is required, though, if a Resource attribute is being created. For more information, see Section 4.5.3, "Managing Resources."

4. Build the Authorization Policy.

   This entails specifying the effect (GRANT or DENY), adding a user, group or role as the policy principal and the Resource and actions as the policy target. Optionally, you can add an Obligation or build a Condition. For more information, see Section 4.5.5, "Managing Authorization Policies."

4.3 Adding Fine-Grained Elements to an Authorization Policy

Section 4.2, "Defining an Authorization Policy And Its Components" documented the minimum components needed to create an authorization policy. The following fine-grained elements can be added to a simple policy.

- Entitlements

  An Entitlement associates an instantiated Resource with the applicable actions that can be performed on it. The set of actions for a Resource are a subset of the set of legal actions already defined in its corresponding Resource Type. For more information, see Section 4.5.4, "Managing Entitlements."

- Application Roles

  An Application Role can be assigned statically or dynamically to a user, group, or external role in an identity store, or another Application Role in the policy store. One target application may have several different Application Roles, with each role assigned a different set of privileges for more fine-grained access. For more information, see Section 4.5.6, "Managing Application Roles in the Role Catalog."

- Role Mapping Policy

  Membership in an Application Role can be granted dynamically with a Role Mapping Policy. An Application Role, referenced as a Principal in a Role Mapping Policy, could grant a user access to the defined resources but the Role Mapping Policy must be resolved before an authorization decision is reached. The resolution of the Role Mapping Policy answers the question *Can the user requesting access be assigned this Application Role?* The Role Mapping Policy returns a list of roles. The results garnered from a Role Mapping Policy (the roles granted the Principal) may be used in an Authorization Policy; the Role Mapping Policy itself can not. During runtime evaluation of a request for access, the following occurs:

  1. Based on the subject requesting access, a list of Application Roles is determined by retrieving static role membership and evaluating any applicable Role Mapping Policies.
  2. Based on the subject and determined list of Application Roles, a list of Authorization Policies is evaluated to find any that might be applicable based on the grantee, target matching and constraints evaluation. The actions allowed on the Resource are defined by the Authorization Policy.
  3. Final authorization decision is based on the “DENY overrides” combining algorithm.

  For more information, see Section 4.5.7, "Managing Role Mapping Policies."

- A Condition can be added to a policy as a way of setting an additional contigency on the policy. It is applicable to either an Authorization Policy or a Role Mapping Policy. A Condition is written in the form of an expression that resolves to true or false and has one of the following outcomes:

  – If the expression resolves to true, the policy condition is satisfied and the effect defined in the PolicyRuleEntry is applicable.

  – If the expression does not resolve to true, the policy is not applicable.

  A Condition must be true for the policy to evaluate to true. Conditions can be complex combinations of boolean expressions that test the value of some user, resource, dynamic or system attributes or, custom Java evaluation functions that evaluate complex business logic. For more information, see Section 4.6, "Using the Condition Builder."

- An Obligation specifies optional information to be evaluated during the policy enforcement phase of authorization. The obligation is returned with the corresponding policy effect (GRANT or DENY). This information may or may not be taken into account during policy enforcement based on settings defined by the application. For example, the reason a request for access has been denied might be

returned as an obligation. A different type of obligation might involve sending a message; for example, if a certain amount of money is withdrawn from a checking account, send a text message to the account holder's registered mobile phone. For more information, see Section 4.5.5, "Managing Authorization Policies."

4.4 Implementing An Authorization Policy Step by Step

In Section 2.4, "Implementing a Policy Use Case," several use cases for creating a policy are discussed. This section documents the step by step procedure to create an Authorization Policy (and the policy objects from which it is comprised) using the Administration Console. This procedure assumes you have installed Oracle Entitlements Server and a Java Security Module to protect an application.

1. Create an Application.

   The Application Name must match what is used in the application code. For example, create a `HelloOESworld` Application object to map to a `HelloOESworld` Application. See Section 4.5.1.1, "Creating an Application."

2. Create a Resource Type.

   The Resource Type Name must match what is used in the application code. For example, create a `Files` Resource Type object for use in collecting files that will be protected. (You might also create Resource attribute for the Resource Type; for example, a `filetype` attribute that may contain a value of html, image, jsp or pdf defines the file type.) Associate the *write* and *read* actions with the Resource Type. See Section 4.5.2.1, "Creating a Resource Type."

3. Create a Resource.

   A Resource Name must match what is used in the application code. Additionally, the Resource is created from the Resource Type. For example, create a `FinanceFile` Resource from the `Files` Resource Type. See Section 4.5.3.1, "Creating a Resource."

4. Create the Authorization Policy.

   In the `HelloOESworld` Application, create an Authorization Policy. Add one or more Principals (Roles or Users), one or more targets (Resources or Entitlements) and confirm the actions for the target. Optional conditions or obligations can also be added before saving. See Section 4.5.5.1, "Creating an Authorization Policy."

5. Create a Security Module definition and bind it to the Application.

   This step defines the Security Module to which this Authorization Policy is distributed once binded. See Section 10.2, "Configuring Security Module Definitions."

6. Distribute the Authorization Policy to the Security Module.

   See Chapter 6, "Managing Policy Distribution."

4.5 Managing Policy Objects in An Application

The following sections describe how to manage policy objects specific to the Applications.

- Section 4.5.1, "Managing Applications"
- Section 4.5.2, "Managing Resource Types"
- Section 4.5.3, "Managing Resources"

- Section 4.5.4, "Managing Entitlements"
- Section 4.5.5, "Managing Authorization Policies"
- Section 4.5.6, "Managing Application Roles in the Role Catalog"
- Section 4.5.7, "Managing Role Mapping Policies"
- Section 4.5.8, "Managing a Role Category"
- Section 4.5.9, "Managing Attributes and Functions as Extensions"

4.5.1 Managing Applications

An Application is created as the overall container for policies and related artifacts that secure the components of a particular application. These artifacts include (but are not limited to) roles, resources, attributes and functions. You may create as many Application instances as needed although it is recommended that only one is created for each application to be secured. The following sections describe management operations on Application instances.

- Creating an Application
- Modifying an Application
- Deleting an Application

4.5.1.1 Creating an Application

To create an Application, proceed as follows:

1. Right-click Applications in the Navigation Panel and select New from the menu.

   Note: Alternately, click Create Application under Search and Create in the Home area.

   An Untitled page with several tabs displays in the Home area. The General tab is active. You can only configure the Delegated Administrators and Policy Distribution details after the Application has been created. See Section 4.5.1.2, "Modifying an Application" for information.

2. Provide the following information for the application being created under the General tab.
   - **Display Name:** The Display Name is optional and case insensitive. Specifying a meaningful value, though, is recommended as it is displayed in the Administration Console and can be used as a search parameter.
   - **Name**: The name is required and case insensitive. It must match what is used in the application code.
   - **Description**: Although optional, it is recommended to provide useful information about the Application.
3. Select one of the following from the Save menu.
   - **Save** saves the configuration, renames the tab with the value provided for the Application's Display Name and activates the Delegated Administrators and Policy Distribution tabs.
   - **Save and Close** saves the configuration and closes the tab.

- **Save and Create Another** saves the configuration to the policy store, refreshes the information tree in the Navigation Panel and leaves an Untitled area open for you to create another Application.

4.5.1.2 Modifying an Application

To modify an Application, proceed as follows:

1. Expand the **Applications** node in the Navigation Panel.
2. Select the name of the Application to modify.
3. Right-click the Application name and select **Open** from the menu.

   Alternately, double-click the Application name. The Application page is displayed and the General tab, the Delegated Administrators tab and the Policy Distribution tab are all active.
4. Select the tab you want to modify or configure and see the appropriate section for parameter details.
   - **General** : Section 4.5.1.1, "Creating an Application"
   - **Delegated Administrators :** Chapter 11, "Delegating With Administrator Roles"
   - **Policy Distribution**: Chapter 6, "Managing Policy Distribution"
5. Apply or save as necessary.

4.5.1.3 Deleting an Application

To delete an Application instance, proceed as follows:

1. Find the Application to delete using an advanced search (as documented in Section 5.3.2, "Searching Applications").

   The Search Applications page is displayed.
2. Enter query parameters and click Search.

   The results are displayed.
3. Select the Application name from the results and click Delete.
4. Choose one of the following methods to search for the Application:

   A Delete Warning is displayed.
5. Click Delete.

   The Application is deleted. Alternately, you can expand the Applications information tree in the Navigation Panel and double click the name of the Application to delete. When the Application's tab is displayed, click Delete in the upper right corner.

4.5.2 Managing Resource Types

Resource Types specify the full scope of traits for a particular kind of protected resource. It contains one or more resource attributes, and definitions of all possible valid actions that can be performed on the particular kind of resource. An *action* represents an activity or task in your business process that can be executed on a resource. Actions can be standard (GET and POST to a URL) or custom on a specific business object (transfer to or from a bank account). A Resource instance for a specific

target is created from a Resource Type. The following sections describe management operations on Resource Types.

- Creating a Resource Type
- Modifying a Resource Type
- Deleting a Resource Type

4.5.2.1 Creating a Resource Type

To create a Resource Type, proceed as follows:

1. Display the page for creating a Resource Type by choosing from the following methods:
   - Expand the information tree in the Navigation Panel, right-click Resource Types under the particular Application in which the Resource Type will be created and select from the menu.
   - In the Home area, select the Application Name under which the Resource Type will be created and click New under Resource Types.

   An Untitled page is displayed in the Home area. Alternately, you can click New from the Simple or Advanced Search results pages. See Chapter 5, "Querying Security Objects" for information.
2. Provide the following information for the Resource Type.
   - **Display Name** : The display name is optional and case insensitive. Specifying a meaningful value, though, is recommended as it is displayed in the Administration Console and can be used as a search parameter.
   - **Name :** The name is required and case insensitive.
   - **Resource Finder** : An (optional) class that implements the `oracle.security.jps.service.policystore.entitymanager.ResourceFinder` interface. It allows resources managed outside of the Policy Store to be consumed. *(Reserved for future use.)*
   - **Description** : Although optional, it is recommended to provide useful information. The description string is case insensitive.
3. Add actions allowed by the Resource Type in the Actions section.
   a. Click **New** to display the New Action dialog
   b. Enter the name of the action.

      The string entered must match the actions for which your application is asking for authorization. If a Permission class is added, the action must be meaningful to it.
   c. Click **Save**.

   The Action list is updated with the new action.
4. Use the Find Existing Attribute dialog to add attributes to the Resource Type being created.
   a. In the Attributes section, click **Add** to display the Find Existing Attribute dialog.

      Before an attribute can be displayed, it must be defined. See Section 4.5.9, "Managing Attributes and Functions as Extensions."
   b. Select the attribute **Type** from the list.

c. Enter an (optional) string to match in the **Search** text box.

d. Click the arrow icon next to the Search text box to begin the search.

e. Select the attributes to add and click **Add**.

Use **Ctrl+click** to select multiple items from the list.

These attributes are used when instantiating a Resource. See Section 4.5.3.1, "Creating a Resource."

5. Configure the remaining fields.

The selection changes according to the Resource Type being created.

- Supports Resource Hiearchy - Select Yes or No to set the Resource Type as hierarchical. This means the following when the Resource Type is used to instantiate a Resource:
  - A policy applicable to a Resource created from a hierarchical Resource Type is also applicable to Resources that are its children.
  - Any attribute defined for a Resource created from a hierarchical Resource Type is inherited by Resources that are its children.
- Resource Name Delimiter - Only valid when Supports Resource Hierarchy is enabled. The default delimiter is `Slash (/)`.
- Evaluation Logic - Evaluation logic for a Resource Type can be either a default matching algorithm or a permission class. Select Default or Permission Class from the drop down menu.
- Permission Class - If the evaluation logic for a Resource Type is defined as Permission Class, specify a case-sensitive Permission class name. An authorization decision for permission based policies is requested using the `checkPermission` call.
- Action Name Delimiter - The specified character is used to separate actions in a list when the Resource Type represents a permission.
- All Action Keyword - If the policy's target contains the defined keyword as an action, the policy will match any action passed in with the authorization request. For example, assume that this parameter is set to `ANY` and you create the following policy:

```
GRANT user "Michael" action:"ANY" on resource:"Resource1
```

The decision for authorization requests like *Can Michael do 'write' on Resource1?* or *Can Michael do 'transfer' on Resource1?* will return ALLOW. The use of this parameter allows you to create a single Authorization Policy that would be applicable to any valid action for that Resource Type.

6. Select one of the following from the Save menu.

- **Save** saves the configuration and renames the tab with the value provided for the Resource Type's Display Name.
- **Save and Close** saves the configuration and closes the tab.
- **Save and Create Another** saves the configuration to the policy store, refreshes the information tree in the Navigation Panel and leaves an Untitled area open for you to create another Resource Type.

4.5.2.2 Modifying a Resource Type

To modify a Resource Type, proceed as follows:

1. Choose from the following methods to display the desired Resource Type.
   - Expand the information tree in the Navigation Panel to find the Resource Types node under the appropriate Application and double click it. A search dialog opens in the Home area. For information about searching in the Home area, see Section 5.3.3, "Searching Resource Types."
   - Search for Resource Types using the Navigation Panel's search function and double-click the Resource Type name in the Search Results tab. For information about searching in the Navigation Panel, see Section 5.2, "Finding Objects with a Simple Search."
   - In the Home area, select the Application Name under which the Resource Type was created and click **Search** under **Resource Types**. A search dialog opens in the Home area. For information about searching in the Home area, see Section 5.3.3, "Searching Resource Types."

   When the correct Resource Type name is displayed, select it and click Open to display the details.
2. Modify as necessary.
3. Click **Apply**.

4.5.2.3 Deleting a Resource Type

To delete a Resource Type, proceed as follows:

1. Choose from the following methods to delete the desired Resource Type.
   - Expand the information tree in the Navigation Panel to find the Resource Types node under the appropriate Application and double click it. A search dialog opens in the Home area. Enter criteria for the lookup and click Search. Select the appropriate Resource Type from the search results and click Delete. For information about searching in the Home area, see Section 5.3, "Finding Objects with an Advanced Search."
   - Search for Resource Types using the Navigation Panel's search function. Find the appropriate Resource Type from the search results and double-click it. After the Resource Type profile opens in the Home area, click Delete. For information about searching in the Navigation Panel, see Section 5.2, "Finding Objects with a Simple Search."

   A Delete Warning is displayed.
2. Click Delete.

   The Resource Type is deleted.

4.5.3 Managing Resources

A Resource represents a specific, secured target in a protected application. Each Resource belongs to a defined Resource Type and can represent software components managed by a container (URLs, EJBs, JSPs) or business objects in an application (reports, transactions, revenue charts).

> **Note:** Resources can be hierarchical (in that the child resource inherits attributes from parent resources) or non-hierarchical. When organized in a hierarchy (root down), you can add new attributes to the parent resources or overwrite any existing attributes that are inherited.

The following sections describe management operations on Resources.

- Creating a Resource
- Modifying a Resource
- Deleting a Resource

4.5.3.1 Creating a Resource

To create a Resource, proceed as follows

1. Display the page for creating a Resource by choosing from the following methods:
   - Navigate to the Resource Catalog by expanding the applicable Policy Domain node in the appropriate Application node using the Navigation Panel. Right-click **Resources** from the Resource Catalog node and select **New** from the menu.
   - Select the Application under which you will create the Resource instance from the Home area and click **New** under **Resources**.

   > **Note:** This option creates the Resource in the Application's Default Policy Domain.

   An Untitled page is displayed in the Home area. Alternately, you can click New from the Simple or Advanced Search results pages. See Chapter 5, "Querying Security Objects" for information.
2. Provide the following information.
   - **Resource Type**: Select from the list. This defines what is displayed in the Instance Attributes and Overwrites table.
   - **Display Name** : The display name is optional and case insensitive. Specifying a meaningful display name is recommended since it is displayed in the Administration Console, and provides extra information to help administrators identify objects.
   - **Name :** The name is required and case sensitive. At runtime, this is the string the application passes to determine whether a user is authorized to access this Resource.
   - **Description** : Although optional, it is recommended to provide useful information about the entitlement. The description string is case insensitive.
3. Add or remove the attributes for this Resource from those displayed in the Instance Attributes and Overwrites dialog.

   The Overwrites dialog is displayed only in the case of hierarchical Resources.
4. Select the attributes from the list (use **Ctrl+click** to select multiple items from the list) and click **Add**.

5. Select one of the following from the Save menu.
   - **Save** saves the configuration and renames the tab with the value provided for the Resource's Display Name.
   - **Save and Close** saves the configuration and closes the tab.
   - **Save and Create Another** saves the configuration to the policy store, refreshes the information tree in the Navigation Panel and leaves an Untitled area open for you to create another Resource.

4.5.3.2 Modifying a Resource

To modify a resource, proceed as follows:

1. Choose from the following methods to display the desired Resource.
   - Navigate to the Resource Catalog by expanding the applicable Policy Domain node in the appropriate Application node using the Navigation Panel and double click it. A search dialog opens in the Home area. For information about searching in the Home area, see Section 5.3.6, "Searching Resources."
   - Search for Resources using the Navigation Panel's search function and double-click the Resource name in the Search Results tab. For information about searching in the Navigation Panel, see Section 5.2, "Finding Objects with a Simple Search."
   - In the Home area, select the Application Name under which the Resource Type was created and click **Search** under **Resources**. A search dialog opens in the Home area. This search dialog will only query the Default Policy Domain. For information about searching in the Home area, see Section 5.3.6, "Searching Resources."

   When the correct Resource name is displayed, select it and click Open to display the details.
2. Modify the Resource as necessary.
3. Click **Apply**.

4.5.3.3 Deleting a Resource

To delete a Resource, proceed as follows:

1. Choose from the following methods to delete the desired Resource.
   - Navigate to the Resource Catalog by expanding the applicable Policy Domain node in the appropriate Application node using the Navigation Panel and double click it. A search dialog opens in the Home area. Enter criteria for the lookup and click Search. Select the appropriate Resource from the search results and click Delete. For information about searching in the Home area, see Section 5.3, "Finding Objects with an Advanced Search."
   - Search for Resources using the Navigation Panel's search function. Find the appropriate Resource name from the search results and double-click it. After the Resource Type profile opens in the Home area, click Delete. For information about searching in the Navigation Panel, see Section 5.2, "Finding Objects with a Simple Search."
   - In the Home area, select the Application Name under which the Resource was created and click **Search** under **Resources**. A search dialog opens in the Home area. Enter criteria for the lookup and click Search. (This search queries only in the Default Policy Domain.) Select the appropriate Resource from the search

results and click Delete. For information about searching in the Home area, see Section 5.3, "Finding Objects with an Advanced Search."

A Delete Warning is displayed.

2. Click Delete.

   The Resource is deleted. If the Resource Type is hierarchical, child resources of the Resource are also deleted.

4.5.4 Managing Entitlements

After instantiating a Resource, define the actions that can be performed on it in an Entitlement. The actions are defined using the set of legal actions defined in the Resource's parent Resource Type. The following sections describe management operations on Entitlements.

- Creating an Entitlement
- Modifying an Entitlement
- Deleting an Entitlement

> **Note:** An Entitlement may be created if there are plans to use the same list of Resource and Action pairs in multiple policies. Otherwise, the Resource and Action pair itself can be directly specified as a target when you create an Authorization Policy. See Section 4.5.5, "Managing Authorization Policies" for more information.

4.5.4.1 Creating an Entitlement

To create an Entitlement, proceed as follows.

1. Display the page for creating an Entitlement by choosing from the following methods:
   - Navigate to the Resource Catalog by expanding the applicable Policy Domain node in the appropriate Application node using the Navigation Panel. Right-click **Entitlements** from the Resource Catalog node and select **New** from the menu.
   - In the Home area, select the Application Name under which the Entitlement will be created and click **New** from **Entitlements**.

   An Untitled page is displayed in the Home area.
2. Provide the following information.
   - **Display Name** : The display name is optional and case insensitive. Specifying a meaningful display name is recommended since it is displayed in the Administration Console, and provides extra information to help administrators identify objects.
   - **Entitlement Name :** The name is required and case insensitive. At runtime, this is the string the application passes to determine whether a user is authorized to access this Resource.
   - **Description** : Although optional, it is recommended to provide useful information about the entitlement. The description string is case insensitive.
3. Choose one of the following methods to add Resources to the Entitlement.
   - Drag and drop

a. Use the Navigation Panel to list the Application's available Resources by performing a search on Resource instances. The Resources must be searched from the same Policy Domain in which the Entitlement is being created. For more information, see Section 5.2, "Finding Objects with a Simple Search".

b. Drag and drop Resources from the **Search Results** tab into the area labeled **Resources**.

- Add Targets pop up search

a. Click **Add** in the **Targets** section.

The **Add Targets** dialog displays. This will search in the current Policy Domain.

b. Search for available targets by entering a string.

The resources matching the query are displayed in **Search Results**. If no search string was entered, a list of all objects of the specified type is returned.

c. Select your choice(s) and click Add Selected.

The Target(s) are added to the Selected Targets. Use **Ctrl+click** to select multiple items from the list.

Note: Alternately, you can click the Resource Expression link under the Resources tab, select a Resource Type, enter a string expression and click Add to Targets. This will search for targets, using the defined criteria, dynamically at runtime. All Resources that belong to the selected Resource Type that contain the string expression are returned, within the context of the administrator privileges.

---

d. Click **Add Targets**.

4. Add actions to the Resources as follows:

a. Select an added resource from the Resources list to display the resource details in the **Resource Details** section.

b. Expand the selected row to see the range of actions.

Only the actions allowed for the type of the selected resource are available in this area.

c. Check the desired actions for the Resource in the **Actions** section.

d. Repeat this procedure for each Resource you have added to the Entitlement being created.

5. Select one of the following from the Save menu.

- **Save** saves the configuration and renames the tab with the value provided for the Entitlement's Display Name.
- **Save and Close** saves the configuration and closes the tab.
- **Save and Create Another** saves the configuration to the policy store, refreshes the information tree in the Navigation Panel and leaves an Untitled area open for you to create another Entitlement.

4.5.4.2 Modifying an Entitlement

To modify an entitlement, proceed as follows:

1. Choose from the following methods to display the desired Entitlement.
   - Navigate to the Resource Catalog by expanding the applicable Policy Domain node in the appropriate Application node using the Navigation Panel and double click it. A search dialog opens in the Home area. For information about searching in the Home area, see Section 5.3.7, "Searching Entitlements."
   - Search for Entitlements using the Navigation Panel's search function and double-click the Entitlement name in the Search Results tab. For information about searching in the Navigation Panel, see Section 5.2, "Finding Objects with a Simple Search."
   - In the Home area, select the Application Name under which the Entitlement was created and click **Search** under **Entitlements**. A search dialog opens in the Home area. For information about searching in the Home area, see Section 5.3.7, "Searching Entitlements."

   When the correct Entitlement name is displayed, select it and click Open to display the details.
2. Modify the entitlement as necessary.
3. Click **Apply**.

4.5.4.3 Deleting an Entitlement

To delete an Entitlement, proceed as follows:

1. Choose from the following methods to delete the desired Entitlement.
   - Expand the information tree in the Navigation Panel to find the Entitlement node under the appropriate Application's Resource Catalog and double click it. A search dialog opens in the Home area. Enter criteria for the lookup and click Search. Select the appropriate Entitlement from the search results and click Delete. For information about searching in the Home area, see Section 5.3, "Finding Objects with an Advanced Search."
   - Search for Entitlements using the Navigation Panel's search function and double-click the Entitlement name in the Search Results tab. Select the appropriate Entitlement from the search results and click Delete. For information about searching in the Navigation Panel, see Section 5.2, "Finding Objects with a Simple Search."
   - In the Home area, select the Application Name under which the Entitlement was created and click **Search** under **Entitlements**. A search dialog opens in the Home area. Enter criteria for the lookup and click Search. Select the appropriate Entitlement from the search results and click Delete. ((In this case, the search is done only within the Application's Default Policy Domain.) For information about searching in the Home area, see Section 5.3, "Finding Objects with an Advanced Search."

   A Delete Warning is displayed.
2. Click Delete.

   The Entitlement is deleted.

4.5.5 Managing Authorization Policies

The Authorization Policy is the mechanism that defines the access rights of a protected resource. A user, an Application Role or an External Role is *granted* the rights of the policy. An Authorization Policy must have:

- At least one principal which can be a user, External Role or Application Role. Code sources are not allowed as a principal.
- At least one target that can be a Resource and Action association (created within the policy) or an Entitlement (created outside the policy and added to it) but not both.
- A defined effect of PERMIT or DENY.

> **Note:** Entitlement-based policies correspond closely with business functions. They are recommended in cases in which a business function considers securing a collection of resources; an entitlement can be used in one or more grants.

The following sections describe management operations on Authorization Policies.

- Creating an Authorization Policy
- Modifying an Authorization Policy
- Deleting an Authorization Policy

4.5.5.1 Creating an Authorization Policy

To create a policy, proceed as follows:

1. Display the page for creating a policy by choosing one of the following methods:
   - Navigate to the Policy Domain under the appropriate Application node in the Navigation Panel and expand it. Right-click **Authorization Policies** from the Resource Catalog node and select **New** from the menu.
   - In the Home area, select the Application Name under which the Authorization Policy will be created and click **New** from **Authorization Policies**. (When using this option, the policy will be created in the Default Policy Domain.)

   An Untitled page is displayed in the Home area.
2. Provide the following information.
   - **Effect**: Select **Permit** if the policy will grant rights or **Deny** if the policy will deny rights.
   - **Display Name** : The display name is optional and case insensitive. Specifying a meaningful display name is recommended since it is displayed in the Administration Console, and provides extra information to help administrators identify objects.
   - **Name :** The name is required and case insensitive. At runtime, this is the string the application passes to determine whether a user is authorized to access this Resource.
   - **Description** : Although optional, it is recommended to provide useful information about the entitlement. The description string is case insensitive.
3. Choose one of the following methods to add Principals to the Authorization Policy.

- Drag and drop

  a. Use the Navigation Panel to list the Application's available Principals by performing a search on Users, External Roles or Application Roles. For more information, see Section 5.2, "Finding Objects with a Simple Search".

  b. Drag and drop Principals from the **Search Results** tab into the area labeled **Principals**.

  c. Select Any or All.

     If Any, the user must match at least one of the specified principals. For example, if the principals are roles, the user must be a member of at least one of the roles for the Authorization Policy to apply. If All, the user must match all of the specified principals. For example, if the principals are roles, the user must be a member of all of them for the Authorization Policy to apply.

- Add Principals pop-up search

  For details on how to use the pop-up search box, see Section 5.1, "Searching with the Administration Console."

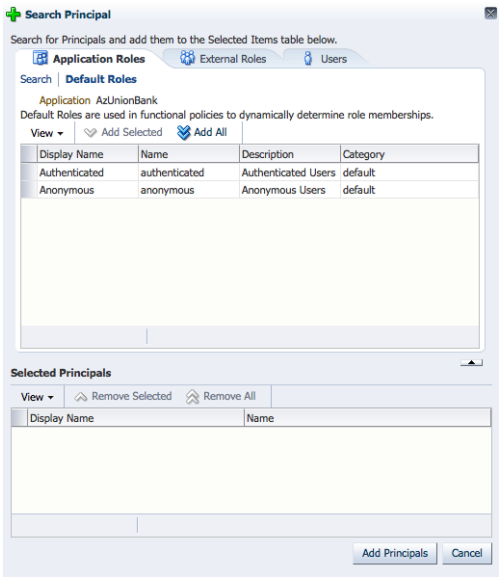
  a. Click **Add** in the **Principals** section.

     The **Add Principals** dialog displays.

  b. Select the appropriate tab to search for available Principals.

     Options are Application Roles, External Roles and Users. You can navigate between tabs and add as many selected Principal types as desired. The Application Roles tab contains two links: Search and Default Roles. The dialog opens on Search which allows a pop-up Application Role search as explained in Chapter 5, "Querying Security Objects." By clicking Default Roles, you can add an Anonymous Role or an Authenticated Role to the Principal. (See *Oracle Fusion Middleware Application Security Guide* for detailed descriptions of these roles.) Figure 4–1 is a screenshot of the Default Roles pop up box.

Figure 4–1 Add Default Roles Pop Up



c. Search for the available Principals by entering a string.

   The Principals matching the query are displayed in **Search Results**.

d. Select your choice(s) and click Add Selected.

   The Principal(s) are added to the Selected Principals. Use **Ctrl+click** to select multiple items from the list.

e. Click **Add Principals**.

f. Select Any or All.

   If Any, the user must match at least one of the specified principals. For example, if the principals are roles, the user must be a member of at least one of the roles for the Authorization Policy to apply. If All, the user must match all of the specified principals. For example, if the principals are roles, the user must be a member of all of them for the Authorization Policy to apply.

4. Choose one of the following methods to add Targets to the Authorization Policy.

   This step adds either Resource and action associations or Entitlements or both to the Authorization Policy.

   - Drag and drop

     a. Use the Navigation Panel to list the Application's available Resources or Entitlements by performing a search. (Be sure to look for these objects in the same Policy Domain to which you are adding the Authorization Policy.) For more information, see Section 5.2, "Finding Objects with a Simple Search".

     b. Drag and drop one or more Resources or Entitlements from the **Search Results** tab into the area labeled **Targets**. Expanding the added object in **Targets** allows you to associate an action with it.

   - Add Targets pop up search

     For details on how to use the pop-up search box, see Section 5.1, "Searching with the Administration Console."

     a. Click **Add** in the **Targets** section.

        The **Add Targets** dialog displays.

     b. Select the appropriate tab to search for available Targets.

        Options are Entitlements and Resources. You can navigate between tabs and add as many selected Targets as desired.

     c. Search for available targets under the Entitlements tab by entering a string.

        The resources matching the query are displayed in **Search Results**. If no search string was entered, a list of all objects of the specified type is returned.

     d. Select your choice(s) and click Add Selected.

        The Target(s) are added to the Selected Targets. Use **Ctrl+click** to select multiple items from the list.

     e. Search for available targets under the Resources tab by entering a string.

The resources matching the query are displayed in **Search Results**. If no search string is entered, a list of all objects of the specified type is returned.

Alternately, you can click the Resource Expression link under the Resources tab, select a Resource Type, enter a string expression and click Add to Targets. This will search for targets, using the defined criteria, dynamically at runtime. All Resources that belong to the selected Resource Type that contain the string expression are returned, within the context of the administrator privileges.

f. Click **Add Targets**.

You will have to select the action for any non-entitlement targets (Resource or Resource Name Expression) added.

5. Select the **Conditions** tab to add a condition.

For more information, see Section 4.6, "Using the Condition Builder."

6. Select the **Obligations** tab.

An Authorization Policy may have zero, one or more Obligations.

a. Click **New** to display the New Obligation dialog.

b. Provide a Name and an (optional) Display Name and Description for the New Obligation and click Add.

c. Click **New** in the Attributes section to add an obligation attribute.

An Obligation has a set of attributes. Each attribute is a name-value pair. The value can be either static or the value of a previously defined attribute. Each obligation should have at least one attribute. See Section 4.5.9, "Managing Attributes and Functions as Extensions" for information.

d. Provide a Name for the attribute in the New Obligation Attribute dialog.

If the obligation attribute is static, select one of the Data Types and provide a Value. If the obligation is an attribute, select **Attribute** for Data Type and choose from the list of predefined attributes.

e. Click **Add**.

7. Click **Save** to save the Authorization Policy.

4.5.5.2 Modifying an Authorization Policy

To modify a policy, proceed as follows:

1. Choose from the following methods to display the desired Authorization Policy.

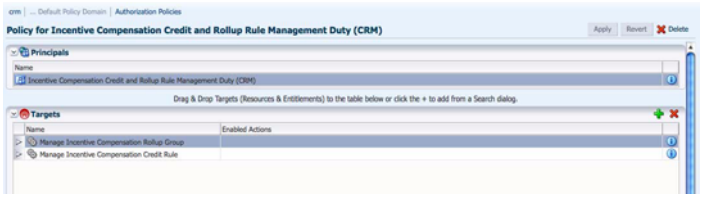
- Expand the information tree in the Navigation Panel to find the Authorization Policies node under the appropriate Application's Policy Domain and double click it. A search dialog opens in the Home area. For information about searching in the Home area, see Section 5.3.8, "Searching Authorization Policies."
- In the Home area, select the Application Name under which the Authorization Policy was created and click **Search** under **Authorization Policies**. A search dialog opens in the Home area. For information about searching in the Home area, see Section 5.3.8, "Searching Authorization Policies."

The search results are displayed.

2. Select the correct policy from the displayed search results and click Open to display its details.

   In addition to displaying advanced Authorization Policies created using Oracle Entitlements Server, the Administration Console also displays the simpler Application Grants (system policies) created using Oracle Platform Security Services (OPSS). The OPSS Application Grants can be viewed, modified or deleted only with the Administration Console. When created using OPSS, Application Grants are defined with a principal and target only. Figure 4–2 is a screenshot of the Oracle Entitlements Server screen when an OPSS Application Grant is displayed.

Figure 4–2 OPSS Application Grants Display Screen



   Note the Name, Display Name and Description fields are not displayed as they would be if the Authorization Policy was created using Oracle Entitlements Server. OPSS Application Grants can not be created using Oracle Entitlements Server. For more information on Application Grants, see the *Oracle Fusion Middleware Application Security Guide*.

3. Modify the policy as necessary.
   - Select the Principal to modify.

     For more information, see Section 4.5.5.1, "Creating an Authorization Policy."
   - Select (or expand) the Target to modify.

     For more information, see Section 4.5.5.1, "Creating an Authorization Policy."
   - Click the Conditions tab to edit conditions.

     For more information, see Section 4.6, "Using the Condition Builder."
   - Click the Obligations tab to modify the Obligation or its attributes.
     - To modify the obligation, click **Edit** from the Obligations table, make changes in the displayed dialog and click **Update**.
     - To modify an attribute, select the attribute from the Attibutes table and click Edit. Make changes in the displayed dialog and click Update.
     - To delete the Obligation, select it in the Obligations table and click Remove.

4. Click **Apply**.

4.5.5.3 Deleting an Authorization Policy

To delete an Authorization Policy, proceed as follows:

1. Choose from the following methods to display the Authorization Policy search screen.

- Expand the information tree in the Navigation Panel to find the Authorization Policies node under the appropriate Application's Policy Domain, right-click it and select Open. A search dialog opens in the Home area.
- In the Home area, select the Application Name under which the Authorization Policy was created and click **Search** under **Authorization Policies**. A search dialog opens in the Home area.

For information about searching in the Home area, see Section 5.3, "Finding Objects with an Advanced Search."

2. Enter criteria for the lookup and click Search.
3. Select the appropriate Authorization Policy from the search results and click Remove.

   The Administration Console can display advanced Authorization Policies created using Oracle Entitlements Server as well as the simpler Application Grants (system policies) created using Oracle Platform Security Services (OPSS). Either Authorization Policies or Application Grants can be deleted by clicking Remove.

4.5.6 Managing Application Roles in the Role Catalog

Application Roles are defined at the Application level (thus, its name) and created using Oracle Entitlements Server. An Application Role can be assigned to an External Role, user, or group in an identity store, or another Application Role in the policy store. One target application may have several different roles, with each assigned a different set of privileges for more fine-grained authorization. Membership can be granted statically to External Roles or individual users, or dynamically using a Role Mapping Policy that is processed at runtime.

Note: A Role Mapping Policy assigns the role to subjects and an Authorization Policy defines the role's access rights.

You can use Application Roles to control access by establishing relationships with the following procedure:

1. Define Application Roles to represent the functional roles users have in the application.
2. Map each Application Role to External Roles or Users.
3. Create Authorization Policies to provide the level of access rights (Permit/Deny) required to meet the goals of the Application Roles.
4. Add the Application Role as a Principal to one or more Authorization Policies.

Application Roles use role inheritance and hierarchy. The inheritance pattern is such that a subject assigned to a role (using a Role Mapping Policy or static role assignments) also inherits any child roles if it is not prohibited by Role Mapping Policies. When an Application Role is referenced as a Principal in a policy, access to the resource for all users assigned to the role is governed by the policy. The following sections describe management operations on Application Roles.

- Creating an Application Role
- Modifying an Application Role
- Mapping External Roles to an Application Role
- Mapping an External User to an Application Role

- Deleting an Application Role or Removing External Role Mappings
- Removing External User Mappings

4.5.6.1 Creating an Application Role

The following procedure describes the steps to create a new Application Role. You are not required to add members to the role at the same time and can return to the saved role later. To create an Application Role, proceed as follows:

1. Display the page for creating an Application Role by choosing one of the following methods:
   - Navigate to the Role Catalog under the appropriate Application node in the Navigation Panel. Right-click the **Role Catalog** node and select **New** from the menu.
   - In the Home area, select the Application Name under which the Application Role will be created and click **New** from **Application Roles**.

   An Untitled page with four tabs is displayed in the Home area: General (active), Application Role Hierarchy, External Role Mapping and External User Mapping.
2. Provide the following information under the **General** tab.
   - **Display Name** : The display name is optional and case insensitive. Specifying a meaningful display name is recommended since it is displayed in the Administration Console, and provides extra information to help administrators identify objects.
   - **Role Name :** The name is required and case insensitive. At runtime, this is the string the application passes to determine whether a user is authorized to access this Resource.
   - **Description** : Although optional, it is recommended to provide useful information about the entitlement. The description string is case insensitive.
   - Role Category : A Role Category is a tag you can assign to a role for ease of management. See Section 4.5.8, "Managing a Role Category."
3. Click **Save**.

   The page is renamed to match the entry provided for Role Name and the Application Role Hierarchy, External Role Mapping and External User Mapping tabs become active. At this point, you can create a policy with this Application Role as the Principal or find a policy with this Application Role as the Principal by clicking Create Policies or Find Policies, respectively. To define the Application Role Hierarchy continue to the next step.
4. Optionally, select the **Application Role Hierarchy** tab to define from which roles this Application Role will inherit permissions (Inherits) and to see a list of roles which are inherited by this role (Is Inherited By). Hierarchy is not required but if you choose to define it, the following example sub procedure is specific to the former option.
   a. Click **Inherits**.
   b. Click **Add**.
   c. Select the radio button that corresponds to the role to which you are adding the hierarchy.

When you add roles to the hierarchy, you can either add the roles to the role under which you are working or to a role that you can select in the Application Role Hierarchy table.

d. Complete the criteria fields in the **Add a Role** dialog and click **Search**.

The results display in the **Search Results** table. Empty strings fetch all roles.

e. Select the role from which this role will inherit permissions in the **Search Results** table.

Use **Ctrl+click** to select multiple roles.

f. Click **Add**.

The selected roles display in the **Application Role Hierarchy** tab, and the Application Role inherits permissions from them.

For information about external role mapping, see Section 4.5.6.3, "Mapping External Roles to an Application Role." For information about external user mapping, see Section 4.5.6.4, "Mapping an External User to an Application Role."

4.5.6.2 Modifying an Application Role

To modify or view an Application Role, proceed as follows:

1. Choose from the following methods to display the desired Application Role.
   - Expand the information tree in the Navigation Panel to find the Role Catalog node under the appropriate Application, right-click it and select Open. A search dialog opens in the Home area. Enter criteria for the lookup and click Search.
   - In the Home area, select the Application Name under which the Application Role was created and click **Search** under **Application Roles**. A search dialog opens in the Home area. Enter criteria for the lookup and click Search.

   When the correct Application Role is displayed, select it and click Open to display the details in the Home area. For information about searching in the Home area, see Section 5.3, "Finding Objects with an Advanced Search."
2. Select the tab that contains the parameters you want to modify and click Add.

   For information on the available tabs, see:
   - Application Role Hierarchy : Creating an Application Role
   - External Role Mapping : Mapping External Roles to an Application Role
   - External User Mapping : Mapping an External User to an Application Role

4.5.6.3 Mapping External Roles to an Application Role

To map external roles to an application role, proceed as follows:

1. Choose from the following methods to display the desired Application Role.
   - Expand the information tree in the Navigation Panel to find the Role Catalog node under the appropriate Application and double click it. A search dialog opens in the Home area. For information about searching in the Home area, see Section 5.3, "Finding Objects with an Advanced Search."
   - Search for Application Roles using the Navigation Panel's search function and double-click the Application Role name in the Search Results tab. For information about searching in the Navigation Panel, see Section 5.2, "Finding Objects with a Simple Search."

- In the Home area, select the Application Name under which the Application Role was created and click **Search** under **Application Roles**. A search dialog opens in the Home area. For information about searching in the Home area, see Section 5.3, "Finding Objects with an Advanced Search."

When the correct Application Role is displayed, select it and click Open to display the details in the Home area.

2. Select the **External Role Mapping** tab.
3. Click **Add** to display the **Add a Role** dialog.
4. Complete the query fields in the **Add a Role** dialog and click **Search**.

   Empty strings fetch all roles. The results display in the **External Role Search** table.
5. Select the external role to map to by clicking its name in the table.

   Use **Ctrl+click** to select multiple roles.
6. Click **Map Roles**.

   The selected roles display in the **External Role Mapping** tab.

4.5.6.4 Mapping an External User to an Application Role

To map an external user to an application role, proceed as follows:

1. Choose from the following methods to display the desired Application Role.
   - Expand the information tree in the Navigation Panel to find the Role Catalog node under the appropriate Application and double click it. A search dialog opens in the Home area. For information about searching in the Home area, see Section 5.3, "Finding Objects with an Advanced Search."
   - Search for Application Roles using the Navigation Panel's search function and double-click the Application Role name in the Search Results tab. For information about searching in the Navigation Panel, see Section 5.2, "Finding Objects with a Simple Search."
   - In the Home area, select the Application Name under which the Application Role was created and click **Search** under **Application Roles**. A search dialog opens in the Home area. For information about searching in the Home area, see Section 5.3, "Finding Objects with an Advanced Search."

   When the correct Application Role is displayed, select it and click Open to display the details in the Home area.
2. Select the **External Users Mapping** tab.
3. Click **Add** to display the **Add a User** dialog.
4. Complete the query fields in the **Add a User** dialog and click **Search**.

   Empty strings fetch all roles. The results display in the **External User Search** table.
5. Select the user to map by selecting its name in the table.

   Use **Ctrl+click** to select multiple roles.
6. Click **Map Users**.

   The selected roles display in the **External User Mapping** tab.

4.5.6.5 Deleting an Application Role or Removing External Role Mappings

To delete an Application Role or remove External Role Mappings from an Application Role, proceed as follows:

1. Choose from the following methods to display the desired Application Role.
   - Expand the information tree in the Navigation Panel to find the Role Catalog node under the appropriate Application, right-click it and select Open. A search dialog opens in the Home area. Enter criteria for the lookup and click Search.
   - In the Home area, select the Application Name under which the Application Role was created and click **Search** under **Application Roles**. A search dialog opens in the Home area. Enter criteria for the lookup and click Search.

   For information about searching in the Home area, see Section 5.3, "Finding Objects with an Advanced Search."
2. Select the Application Role in the Search Results table and:
   - Click Delete to remove the role.
   - Select the appropriate mapping in the External Role Mapping table and click Remove.
   - To remove External User Mappings, see Section 4.5.6.6, "Removing External User Mappings."

4.5.6.6 Removing External User Mappings

To remove External User Mappings from an Application Role, proceed as follows:

1. Choose from the following methods to display the desired Application Role.
   - Expand the information tree in the Navigation Panel to find the Role Catalog node under the appropriate Application, right-click it and select Open. A search dialog opens in the Home area. Enter criteria for the lookup and click Search.
   - In the Home area, select the Application Name under which the Application Role was created and click **Search** under **Application Roles**. A search dialog opens in the Home area. Enter criteria for the lookup and click Search.

   For information about searching in the Home area, see Section 5.3, "Finding Objects with an Advanced Search."
2. Select the Application Role in the Search Results table and click Open.

   The Application Role details open in a new tab.
3. Click the External User Mappings tab.
4. Select the appropriate User and click Remove.

   A dialog box is displayed asking for confirmation.
5. Click Delete to remove the External User Mapping.

   You can also remove External Role Mappings by clicking the External Role Mappings tab.

4.5.7 Managing Role Mapping Policies

Membership to an Application Role can be granted statically or dynamically with a Role Mapping Policy. An Application Role, referenced in a Role Mapping Policy,

could grant a user access to the defined resources. The following sections describe management operations on Role Mapping Policies.

- Creating a Role Mapping Policy
- Modifying a Role Mapping Policy
- Deleting a Role Mapping Policy

4.5.7.1 Creating a Role Mapping Policy

To create a Role Mapping Policy, proceed as follows:

1. Display the page for creating a Role Mapping Policy by choosing one of the following methods:
   - Navigate to the appropriate Application node in the Navigation Panel and expand the Role Catalog branch. Right-click **Role Mapping Policies** and select **New** from the menu.
   - In the Home area, select the Application Name under which the Role Mapping Policy will be created and click **New** from **Role Mapping Policies**.

   An Untitled page is displayed in the Home area.
2. Provide the following information.
   - **Effect**: Select **Permit** if the policy will grant membership in the Application Role or **Deny** if the policy will deny membership in the Application Role.
   - **Display Name** : The display name is optional and case insensitive. Specifying a meaningful display name is recommended since it is displayed in the Administration Console, and provides extra information to help administrators identify objects.
   - **Name :** The name is required and case insensitive.
   - **Description** : Although optional, it is recommended to provide useful information about the policy. The description string is case insensitive.
3. Choose one of the following methods to add Application Roles.
   - Drag and drop
     a. Use the Navigation Panel to list the Application's available Application Roles by performing a search. For more information, see Section 5.2, "Finding Objects with a Simple Search".
     b. Drag and drop Application Roles from the **Search Results** tab into the area labeled **App Role**.
   - Add Application Roles dialog
     a. Click **Add** in the **App Role** section.

        The **Search Application Roles** dialog displays.
     b. Search for the available Application Roles by entering a string.

        The resources matching the query are displayed in **Search Results**.
     c. Select the principals to add and click **Add Application Roles**.

        Use **Ctrl+click** to select multiple items from the list.

Note: For this release, this dialog displays the Search Principals title and Add Principals button.

4. Choose one of the following methods to add Principals.
   - Drag and drop
     a. Use the Navigation Panel to list the Application's available Users and External Roles by performing a search. For more information, see Section 5.2, "Finding Objects with a Simple Search".
     b. Drag and drop Users and External Roles from the **Search Results** tab into the area labeled **Principals**.
   - Add Principals dialog
     a. Click **Add** in the **Principals** section.

        The **Search Principals** dialog displays.
     b. Search for the available Principals (in this case, Users or External Roles) by entering a string.

        The resources matching the query are displayed in **Search Results**.
     c. Select the principals to add and click **Add Principals**.

        Use **Ctrl+click** to select multiple items from the list.
5. Optionally, choose one of the following methods to add Resources (also referred to as Targets).
   - Drag and drop
     a. Use the Navigation Panel to list the Application's available Resources by performing a search. For more information, see Section 5.2, "Finding Objects with a Simple Search".
     b. Drag and drop one or more Resources from the **Search Results** tab into the area labeled **Resources**.
   - Add Targets pop up search
     a. Click **Add** in the **Resources** section.

        The **Add Targets** dialog displays.
     b. Choose the Policy Domain that contains the Resource (if applicable).
     c. Enter a string and click Search.

        The resources matching the query are displayed in **Search Results**. If no search string was entered, a list of all objects of the specified type is returned.
     d. Select the appropriate Targets to add and click **Add Selected**.

        The Target(s) are added to the Selected Targets. Use **Ctrl+click** to select multiple items from the list.
     e. Click the Resource Expression link to add an expression as a Target.

        Select a Resource Type, enter a string expression and click Add to Targets. This will search for targets, using the defined criteria, dynamically at runtime. All Resources that belong to the selected Resource Type that con-

tain the string expression are returned, within the context of the administrator privileges.

f. Click **Add Targets**.

6. See Section 4.6, "Using the Condition Builder" for information on using the Condition Builder.
7. Click Save.

4.5.7.2 Modifying a Role Mapping Policy

To modify a Role Mapping Policy, proceed as follows:

1. Choose from the following methods to display the desired Role Mapping Policy.
   - Expand the information tree in the Navigation Panel to find Role Mapping Policies under the Role Catalog node of the appropriate Application and double click Role Mapping Policies. A search dialog opens in the Home area. For information about searching in the Home area, see Section 5.3, "Finding Objects with an Advanced Search."
   - In the Home area, select the Application Name under which the Application Role was created and click **Search** under **Role Mapping Policies**. A search dialog opens in the Home area. For information about searching in the Home area, see Section 5.3, "Finding Objects with an Advanced Search."

   When the correct Role Mapping Policy is displayed, select it and click Open to display the details in the Home area.
2. Modify the policy as necessary.
3. Click Apply.

4.5.7.3 Deleting a Role Mapping Policy

To delete a Role Mapping Policy, proceed as follows:

1. Choose from the following methods to display the desired Role Mapping Policy.
   - Expand the information tree in the Navigation Panel to find Role Mapping Policies under the Role Catalog node of the appropriate Application and double click Role Mapping Policies. A search dialog opens in the Home area. For information about searching in the Home area, see Section 5.3, "Finding Objects with an Advanced Search."
   - In the Home area, select the Application Name under which the Application Role was created and click **Search** under **Role Mapping Policies**. A search dialog opens in the Home area. For information about searching in the Home area, see Section 5.3, "Finding Objects with an Advanced Search."

   When the correct Role Mapping Policy is displayed, select it and click Open to display the details in the Home area.
2. Double-click the Role Mapping Policy to delete.

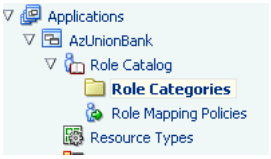
   The Role Mapping Policy displays in the Home area.
3. Click Delete in the upper right corner of the Home area.

4.5.8 Managing a Role Category

A Role Category is a tag you can assign to a role for ease of management. You can create or delete a Role Category but you cannot modify them. To create a Role

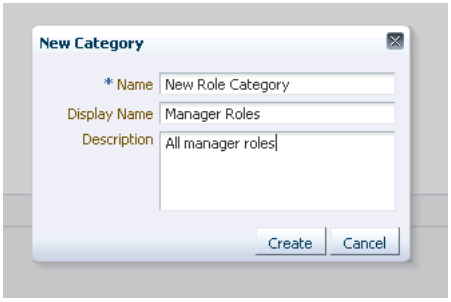
Category, proceed as follows. Instructions to delete a Role Category are detailed after the final step.

1. Expand the appropriate Application node in the Navigation Panel and double-click the **Roles Categories** node.



   The Role Categories page opens in the Home area.

2. Click **New** to display the **New Category** dialog.
3. Provide the following information.
   - **Name**
   - **Display Name**
   - **Description**



4. Click **Create**.

   The new category displays in the Role Categories list.

To delete a Role Category, expand the appropriate Application node in the Navigation Panel and double-click the **Roles Categories** node. Select the Role Category to delete and click Delete.

4.5.9 Managing Attributes and Functions as Extensions

Attributes and Functions are definitions organized under the Extensions node of the Application for which they were created. Attribute and function definitions can be used in a Condition or an Obligation. In regards to a Condition, attribute and function definitions can be used to make an optional expression that can be added to a policy to further restrict access to the protected resource. In regards to an Obligation, this optional set of name-value pairs returns additional information, with a policy decision, to the calling application. There are two ways to define an Obligation:

- Statically where an attribute with an absolute value is returned.
- Dynamically where an attribute value, or a custom function, is evaluated at runtime and the output is returned.

An Attribute can be a value dynamically defined at runtime (for example, the locality of the user) or a value based on the type of protected resource (for example, creation date of a text file). During policy evaluation, attribute values can be passed in by the application or Oracle Entitlements Server can retrieve it using a custom attribute retriever. Attributes must have a defined type. Boolean, integer, date, time and string are Oracle Entitlements Server predefined types. An attribute may be singular or a multi-valued list. A Function is a definition of externally implemented logic. It can be added to a policy as a condition on the policy's outcome. The following sections describe management operations on Attributes and Functions.

- Creating an Attribute
- Modifying an Attribute
- Deleting an Attribute
- Creating a Function
- Modifying a Function
- Deleting a Function

4.5.9.1 Creating an Attribute

To create an attribute, proceed as follows:

1. Navigate to, and expand, Extensions under the appropriate Application node in the Navigation Panel.
2. Right-click the **Attributes** node and select **New** from the menu.

   An Untitled page is displayed in the Home area.
3. Provide the following information for the attribute.
   - **Display Name** : The display name is optional and case insensitive. Specifying a meaningful display name is recommended since it is displayed in the Administration Console, and provides extra information to help administrators identify objects.
   - **Name :** The name is required and case insensitive. At runtime, this is the string the application passes to determine whether a user is authorized to access this Resource.
   - **Description** : Although optional, it is recommended to provide useful information about the entitlement. The description string is case insensitive.
   - **Category**: Select from Resource and Dynamic as a value for this required parameter. A Resource attribute is defined and maintained using Oracle Entitlements Server. All other attributes would be Dynamic.
   - **Type**: Select from Boolean, Date, Date Time, Integer, String, Time, Base64 Binary, Day Time Duration, DNS Name, Double, Email, Hex Binary, IP Address, URI, X500 Name or Year Month Duration.
   - **Input Values**: Select from Single and Multiple.
4. Select one of the following from the Save menu.
   - **Save** saves the configuration and renames the tab with the value provided for the Display Name.

- **Save and Close** saves the configuration and closes the tab.
- **Save and Create Another** saves the configuration to the policy store, refreshes the information tree in the Navigation Panel and leaves an Untitled area open for you to create another Attribute.

4.5.9.2 Modifying an Attribute

To modify an attribute, proceed as follows:

1. Choose from the following methods to display the desired Attribute.
   - Navigate to, and expand, Extensions under the appropriate Application node in the Navigation Panel. Double-click Attributes to open a search dialog in the Home area. For information about searching in the Home area, see Section 5.3.9, "Searching Attributes."
   - Search for Attributes using the Navigation Panel's search function and double-click the Attribute name in the Search Results tab. For information about searching in the Navigation Panel, see Section 5.2, "Finding Objects with a Simple Search."

   When the correct Attribute is displayed, select it and click Open to display the details in the Home area.
2. Modify the attribute as necessary.
3. Click **Apply**.

4.5.9.3 Deleting an Attribute

To delete an attribute, proceed as follows:

1. Choose from the following methods to display the Attribute.
   - Navigate to, and expand, Extensions under the appropriate Application node in the Navigation Panel. Right-click Attributes and select Open to display a search dialog in the Home area. Enter criteria for the lookup and click Search. For information about searching in the Home area, see Section 5.3, "Finding Objects with an Advanced Search."
   - Search for Attributes using the Navigation Panel's search function, right-click the Attribute name in the Search Results tab and select Open to display the Attribute in the Home area. For information about searching in the Navigation Panel, see Section 5.2, "Finding Objects with a Simple Search."
2. Select the Attribute and click **Delete**.

   A Delete Warning is displayed.
3. Click Yes.

   If the Attribute is being used, it can not be deleted.

4.5.9.4 Creating a Function

To create a function, proceed as follows:

1. Navigate to, and expand, Extensions under the appropriate Application node in the Navigation Panel.
2. Right-click the **Functions** node and select **New** from the menu.

   An Untitled page is displayed in the Home area.
3. Provide the following information for the function.

- **Display Name** : The display name is optional and case insensitive. Specifying a meaningful display name is recommended since it is displayed in the Administration Console, and provides extra information to help administrators identify objects.
- **Name :** The name is required and case insensitive. At runtime, this is the string the application passes to determine whether a user is authorized to access this Resource.
- **Description** : Although optional, it is recommended to provide useful information about the entitlement. The description string is case insensitive.
- **Function Class Name**: The name of the class that provides the functionality.

  If creating a custom function implemented from the `InspectableOesFunction` interface, Oracle Entitlements Server will load the metadata based on the interface and dynamically evaluate the class to ensure its viablility. See the *Oracle Fusion Middleware Developer's Guide for Oracle Entitlements Server* for details.
- **Input Parameter**: A list of the types of parameters passed to the function.
- **Return Type**: Select the data type returned by the function.
- **Syntax Preview** displays a preview of the function's syntax.

4. Select one of the following from the Save menu.
   - **Save and Close** saves the configuration and renames the page with the value provided for the Display Name.
   - **Save and Create Another** saves the configuration to the information tree in the Navigation Panel but leaves the Untitled area open for you to create another.

4.5.9.5 Modifying a Function

To modify a function, proceed as follows:

1. Navigate to, and expand, Extensions under the appropriate Application node in the Navigation Panel.
2. Double-click Functions to open a search dialog in the Home area.
3. Enter search criteria to display the Function.

   For information about searching in the Home area, see Section 5.3.10, "Searching Functions."
4. Select the Function from the Search Results and click Open.

   The Function's details are displayed in the Home area.
5. Modify the Function as necessary.
6. Click **Apply**.

4.5.9.6 Deleting a Function

To delete a Function, proceed as follows:

1. Choose from the following methods to display the Function.
   - Navigate to, and expand, Extensions under the appropriate Application node in the Navigation Panel. Right-click Functions and select Open to display a search dialog in the Home area. Enter criteria for the lookup and click Search.

Select the appropriate Function from the Search Results. For information about searching in the Home area, see Section 5.3, "Finding Objects with an Advanced Search."

- Search for Functions using the Navigation Panel's search function, right-click the Function name in the Search Results tab and select Open to display the Function in the Home area. For information about searching in the Navigation Panel, see Section 5.2, "Finding Objects with a Simple Search."

2. Click Delete.

   A Delete Warning is displayed.

3. Click Yes.

   If the Function is being used, it can not be deleted.

4.6 Using the Condition Builder

An optional Condition in a policy rule can be used to further evaluate the applicability of an authorization decision returned in response to a request for access. For example, a Condition can be used to grant access to a resource only on the condition that the request was issued from a specific location or at a specific time.

Note: Conditions in Role Mapping Policies provide the same functionality, and take the same format, as conditions in Authorization Policies.

A Condition is written in the form of an expression that resolves to either true or false. If the expression resolves to true, the condition is satisfied and the policy is applicable. If the expression does not resolve to true, the policy is not applicable. The expression can operate on attributes, functions or literals. Oracle Entitlements Server contains predefined attributes and functions that can be inserted or you can create custom ones. The literals belong to the supported data types and are constants.

Note: All Attributes and Functions (both custom and predefined) are created, collected and further managed under the Extensions node of the Application. For more information, see Section 4.5.9, "Managing Attributes and Functions as Extensions."

The Condition Builder allows an administrator to quickly create Condition expressions that can then be added to an Authorization Policy or a Role Mapping Policy. The following procedure illustrates how to use the Condition Builder to create a Condition for your policy. To create a Condition, you either create or modify an Authorization Policy or a Role Mapping Policy. Following one of these procedures will bring you to a step in which you can build a Condition.

- Creating an Authorization Policy or
- Modifying an Authorization Policy
- Creating a Role Mapping Policy
- Modifying a Role Mapping Policy

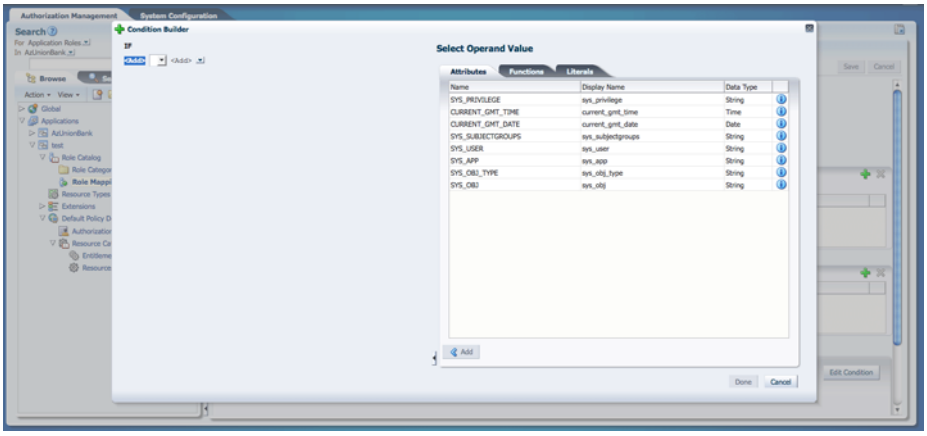
When you get to the appropriate screen, follow this procedure.

1. Click the Condition tab.

2. Click Edit Condition.

   The Condition Builder (as displayed in Figure 4–3) displays. Note the Condition expression on the left contains two **Add** replaceables and an operator drop down (which is empty until an operand has been added). Build the expression by adding components from the Attributes, Functions and Literals tabs on the right.

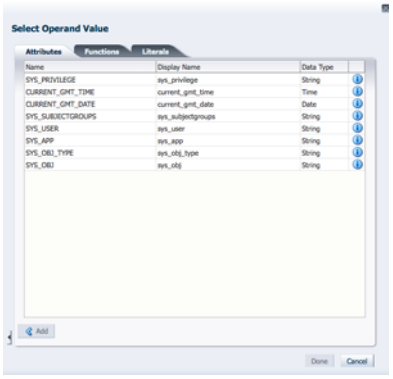
Figure 4–3 The Condition Builder



3. Click the apropriate Operand Value tab that contains the component type you want to add to the Condition.

   Figure 4–4 is a screen shot of the Attributes, Functions and Literals tabs. The Attributes and Functions listed in these tabs are filtered based on the Application in which the policy is being created. For example, a custom Function created within Application 1 will not be visible when the Condition Builder is activated to create a policy within Application 2.

Figure 4–4 Operand Value Tabs



Note: If accessing the Functions tab to add a custom function implemented from the `InspectableOesFunction` interface, Oracle Entitlements Server will dynamically evaluate the values entered based on how the interface is implemented. See the *Oracle Fusion Middleware Developer's Guide for Oracle Entitlements Server* for details.

---

4. Select the line that contains the component you want to add to the Condition and click Add.

   Click the blue **i** to display a Details box with more information regarding the component. Figure 4–5 is a screenshot after having added a SYS_APP attribute which takes a string value.

Figure 4–5 Adding a Literal to the Condition

5. Populate the value on the right of the expression by selecting the appropriate Operand Value and click Add.
6. Specify the operator on the right of the Condition Builder by clicking the drop down and selecting your choice.

   The operator options are dependent on the Operand Value.
7. Add additional expressions by clicking the last arrow in the expression and selecting AND, OR or NOT from the crop down menu, if applicable.

   REMOVE will clear the expression of all components so you may begin again.
8. Select components for the additional expression from the appropriate Operand Value tabs, if applicable.

   You may add as many expressions (and components) as necessary by clicking the last arrow in the current expression and selecting from the Operand Value tabs.
9. Click Done to complete the Condition.

The following points should be taken into account as you navigate the Condition Builder to create your expression.

- The Condition Builder contains Tool Tips on most fields for additional details.
- Click the appropriate blue **i** for information on the Operand Value.
- At the minimum, an expression must contain two operands and an operator.
- You can compare an Attribute and an Attribute, an Attribute and a Function, an Attribute and a Literal, a Function and a Function, and a Function and a Literal.
- The input parameters for Functions can be Attributes, Literals or Functions.
- The choice of operators displayed is directly related to the first operand chosen. For example, you cannot do less than or equal to on a string.
- The choice of a second Operand Values displayed within an expression is also directly related to the first operand chosen.
- REMOVE clears the expression to which it is tied of all components so you may begin again. It does not clear the entire Condition.
- The completed Condition (expression) is evaluated by Oracle Entitlements Server at runtime. The interpretation is governed by the rules of precedence.
- The outcome of this Condition must be a boolean.

The following sections contain procedures for more complex conditions.

- Building a Complex Expression
- Passing Parameters to Functions

4.6.1 Building a Complex Expression

This procedure explains how you might build a complex expression using parenthesis.

1. Follow one of these procedures to bring you to the Condition Builder.
   - Creating an Authorization Policy or
   - Modifying an Authorization Policy
   - Creating a Role Mapping Policy
   - Modifying a Role Mapping Policy
2. Click the Condition tab.
3. Click Edit Condition.

   The Condition Builder displays as in Figure 4–3.
4. Click the Attributes tab.
5. Select the `DateAttr` custom attribute and click Add.

   `DateAttr` is not a predefined Oracle Entitlements Server attribute so this step assumes a custom attribute has been defined as documented in Section 4.5.9, "Managing Attributes and Functions as Extensions." DateAttr is added to the left of the operator.
6. Select the equal sign (=) as the operator.
7. Select the `CURRENT_GMT_DATE` predefined attribute and click Add.

   `CURRENT_GMT_DATE` is a predefined Oracle Entitlements Server attribute and can be viewed under the Attributes tab. It is added to the right of the operator.
8. Add more complexity to the Condition by selecting the appropriate AND, OR or NOT operation at the end of the line of code.

   Parentheses must match; there must be an equal number of open and closing parentheses. If you select an operation at the end of a line of code, the operation will involve the code itself. If you select an operation at the end of the entire Condition, it will allow you to add on to the Condition as a whole.
9. Add additional conditions by choosing values from Attributes, Functions or Literals as necessary.
10. Click Done when finished.

4.6.2 Passing Parameters to Functions

This procedure describes how to pass parameters into a Function.

Note: If adding a custom function implemented from the `InspectableOesFunction` interface, Oracle Entitlements Server will evaluate the class on the fly to ensure its viablility. See the *Oracle Fusion Middleware Developer's Guide for Oracle Entitlements Server* for details.

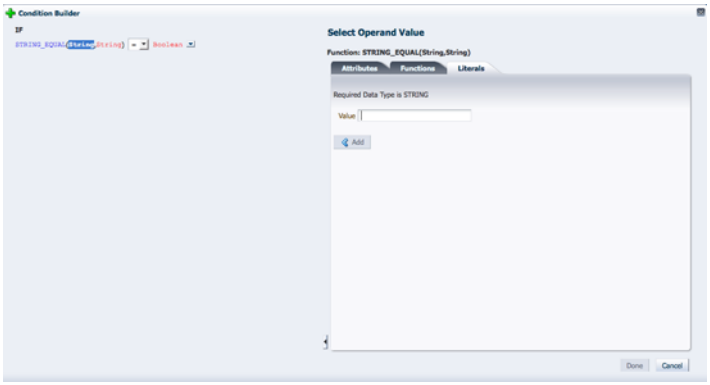
---

1. Follow one of these procedures to bring you to the Condition Builder.
   - Creating an Authorization Policy or
   - Modifying an Authorization Policy
   - Creating a Role Mapping Policy
   - Modifying a Role Mapping Policy
2. Click the Condition tab.
3. Click Edit Condition.

   The Condition Builder displays as in Figure 4–3.
4. Click the Functions tab.
5. Select STRING_EQUAL and click Add.

   Figure 4–6 illustrates an added Function and contains placeholders for the two parameters that must be passed to it. This Function will compare the two strings (one the value of a predefined attribute).

Figure 4–6 Adding a Function



6. Select the first parameter if not already.
7. Click the Attributes tab.
8. Select SYS_USER and click Add.

   The second parameter is highlighted and the Literal tab is activated.
9. Enter a value for the second parameter and click Add.

   For this example, joe. The boolean to the right of the operator is highlighted and the Literal tab is activated.
10. Choose the appropriate operator.
11. Click the Boolean replaceable and select whether this function output should be true or false.
12. Add Additional operands as you see fit.
13. Click Done when finished.

5

Querying Security Objects

Oracle Entitlements Server enables querying for policies and policy objects from within the Oracle Entitlements Server Administration Console. This chapter explains the types of search functionalities and for what purposes they can be used. It contains the following topics:

- Searching with the Administration Console
- Finding Objects with a Simple Search
- Finding Objects with an Advanced Search
- Understanding Case Sensitivity in Object Names

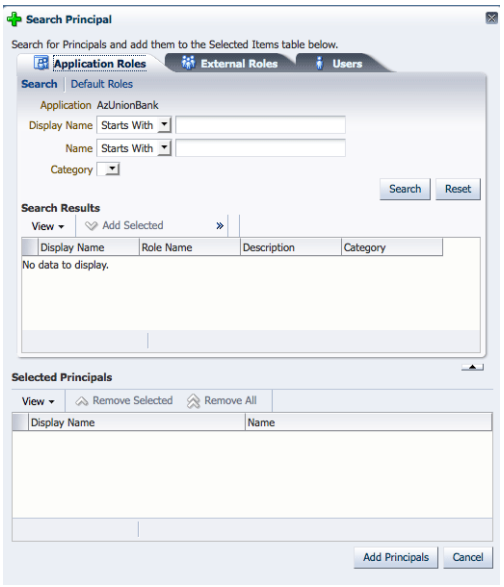
5.1 Searching with the Administration Console

Oracle Entitlements Server enables different kinds of search queries using the Administration Console.

- A *simple search* matches names and display names only. The search is generated from the top of the Navigation Panel and results are displayed in the Navigation Panel. For more information, see Section 5.2, "Finding Objects with a Simple Search."
- An *advanced search* uses operators that enable more sophisticated matching. The advanced search screen is launched by double-clicking an object in the Navigation Panel, or from the Home area. The search box opens in the Home area and results are also displayed there. For more information, see Section 5.3, "Finding Objects with an Advanced Search."
- A *blind search* will search objects without specifying search criteria. This can be done as a simple search or an advanced search. A blind search will display no more than 300 objects in the system. Oracle Entitlements Server will not display more than 300 rows in the search results.
- A *pop-up search* opens from within the Authorization Policy or Role Mapping Policy screens, when the policy is being created or modified, by clicking the green Add button (plus sign). The pop-up search box uses a shopping cart paradigm. You add choices selected from the multiple, displayed tabs on the top of the search box to the Selected box on the bottom of the search box. All choices in the Selected box are added when you click Add.

  Figure 5–1 is a screen shot of the pop-up search box for adding a Principal. You can click between the three tabs (Application Roles, External Roles, and Users), selecting one or more policy subjects and adding them to the Selected Principals box. When you click Add Principals, all choices added from all tabs will then be added to the policy.

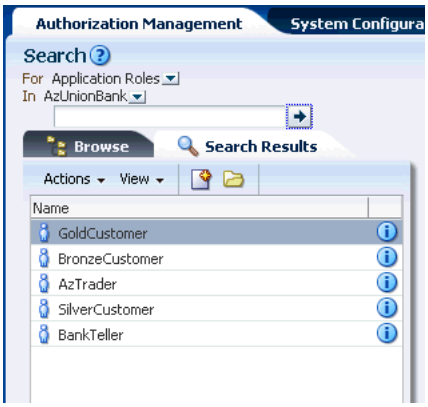
Figure 5–1 Pop-up Search Box



5.2 Finding Objects with a Simple Search

A simple query matches names and display names only. The fields in the top portion of the Authorization Management tab in the Navigation Panel, as shown in Figure 5–2, are used to specify simple queries.

Figure 5–2 Simple Search Fields and Results Tab in Navigation Panel



To specify a simple search, proceed as follows:

1. Select the policy object for which you are searching from the **For** list.

   The following object types are available:

   - Application Roles
   - External Roles
   - Users
   - Resources
   - Resource Types

- Entitlements
- Attributes

2. Select the search scope from the **In** list.

   The *search scope* defines the level at which the search will take place. When searching for Application Roles, Resources, Resource Types, Entitlements and Attributes, the search scope is an Application. For External Roles and Users, the search scope can be Global (the default option) or the name of an Application bound to a particular Identity Directory Service profile.

   Note: In the latter case, the search will be in the identity data store that corresponds to the Identity Directory Service profile to which the Application is bound. See Section 10.3, "Configuring Identity Directory Service Profiles" for more information.

   For Entitlements and Resources, the search scope is the Policy Domain within an Application. If performing a Resource search, you also select the Resource Type from the **Type** list.

3. Optionally, enter a string to match in the text box.

   Wildcard characters percent (%) and asterisk (*) are supported for a simple search.

4. Click the arrow icon next to the text box to begin the search.

   Names and display names matching the specified criteria are returned and displayed in the **Search Results** tab. If no search string was entered, a list of all objects of the specified type is returned.

5. Double-click the object to edit, right click the object and select New to create, or click the object's information icon for details.

   For more information on managing policy objects, see Chapter 4, "Managing Policies and Policy Objects."

5.3 Finding Objects with an Advanced Search

An advanced search is generally initiated by double-clicking the object name in the Navigation Panel, or from the Search link for the object in the Home area. An advanced search can use the following operators:

- Starts with
- Ends with
- Contains
- Equal to

There is no support for wildcard characters in an advanced search. In particular, the asterisk (*) or percent (%) characters are treated as plain text in any advanced search parameter. The following sections have information on searching for policy objects with an advanced search.

- Searching External Roles
- Searching Applications
- Searching Resource Types
- Searching Application Roles

- Searching Role Mapping Policies
- Searching Resources
- Searching Entitlements
- Searching Authorization Policies
- Searching Attributes
- Searching Functions
- Searching for Users

5.3.1 Searching External Roles

To search External Roles, proceed as follows:

1. Select from the following methods to display the Search External Roles page:
   - In the Navigation Panel, expand Global and double-click External Roles.

     Alternately, right-click External Roles and select Open.
   - In the Home area, click **Search - External Roles** from the Search and Create section.
2. Enter the following query parameters:
   - **Name**: Select an operator from the list and enter a string to match.
   - **Display Name**: Select an operator from the list and enter a string to match.

   Optionally, select from the **Saved Search** drop-down list of previously saved searches. Its query parameters automatically populate the search fields. Select **Personalize...** to set options for previously saved searches.
3. Optionally, click **Save...** to name the current query parameters.

   The named search is added to the **Saved Search** list.
4. Click **Search**.

   The results are displayed in Search Results.

5.3.2 Searching Applications

To search applications, proceed as follows:

1. Select from the following methods to display the Search Applications page:
   - In the Navigation Panel, double-click Applications to display the Search Applications page.

     Alternately, right-click Applications and select Open.
   - In the Home area, click **Search - Applications** from the Search and Create section.
2. Enter the following query parameters:
   - **Name**: Select an operator from the list and enter a string to match.
   - **Display Name**: Select an operator from the list and enter a string to match.

   Optionally, select from the **Saved Search** drop-down list of previously saved searches. Its query parameters automatically populate the search fields. Select **Personalize...** to set options for previously saved searches.

3. Optionally, click Save... to save the current query parameters as a Saved Search.

   The search is added to the Saved Search list.

4. Click Search.

   The results are displayed in Search Results.

5.3.3 Searching Resource Types

To search Resource Types, proceed as follows:

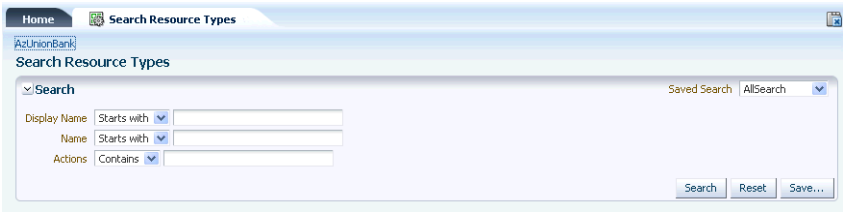
1. Select from the following methods to display the Search Resource Types page as in Figure 5–3.

   - In the Navigation Panel, expand the Application node and double-click Resource Types.

     Alternately, right-click Resource Types and select Open.

   - In the Home area, select the appropriate Application Name and click Search under Resource Types.

Figure 5–3 Searching for Resource Types



2. Enter the following query parameters:

   - **Display Name:** Select an operator from the list and enter a string to match.
   - **Name**: Select an operator from the list and enter a string to match.
   - **Actions**: Select an operator from the list and enter a string to match.

   Optionally, select from the Saved Search drop-down list of previously saved searches. Its query parameters automatically populate the search fields. Select Personalize... to set options for previously saved searches.

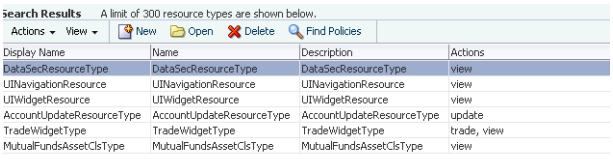
3. Optionally, click Save... to save the current query parameters as a Saved Search.

   The search is added to the Saved Search list.

4. Click Search.

   All results matching the query specifications are displayed in the **Search Results** table as illustrated in Figure 5–4.

Figure 5–4 Resource Type Search Results



5.3.4 Searching Application Roles

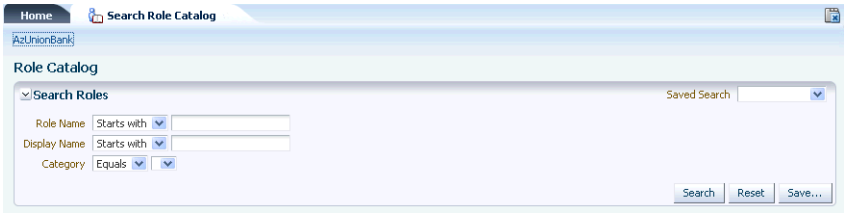
To search Application Roles, proceed as follows:

1. Select from the following methods to display the Search Role Catalog page.
   - In the Navigation Panel, expand Applications and the named Application node applicable to the search, and double-click Role Catalog.

     Alternately, right-click Role Catalog and select Open.
   - In the Home area, select the Application Name and click Search from Application Roles.

   The Search Role Catalog tab is displayed as in Figure 5–5.

Figure 5–5 Searching for Application Roles in a Role Catalog



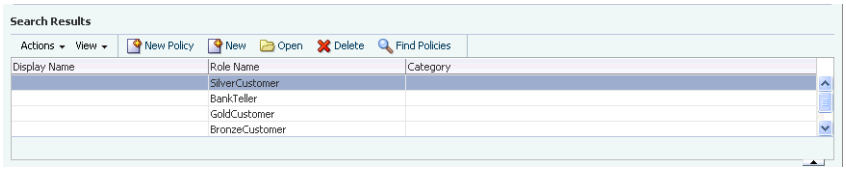
2. Enter the following query parameters:
   - **Role Name:** Select an operator from the list and enter a string to match.
   - **Display Name:** Select an operator from the list and enter a string to match.
   - **Category**: Select a Role Category from the list. (Oracle Entitlements Server only supports an *equals* search for Role Category.)

   Optionally, select from the Saved Search drop-down list of previously saved searches. Its query parameters automatically populate the search fields. Select Personalize... to set options for previously saved searches.
3. Optionally, click Save... to save the current query parameters as a Saved Search.

   The search is added to the Saved Search list.
4. Click Search.

   All results matching the query specifications are displayed in the **Search Results** table as in Figure 5–6.

Figure 5–6 Application Role Search Results



5.3.5 Searching Role Mapping Policies

1. Select from the following methods to display the Search Role Mapping Policies page:

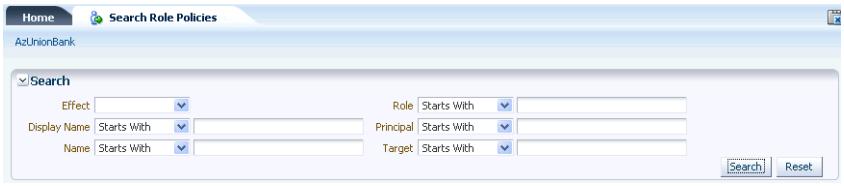
- In the Navigation Panel, expand Applications and the named Application node applicable to the search, and double-click Role Mapping Policies.

  Alternately, right-click Role Mapping Policies and select Open.

- In the Home area, select the Application Name and click Search from Role Mapping Policies.

The Search Role Policies page is displayed as in Figure 5–7.

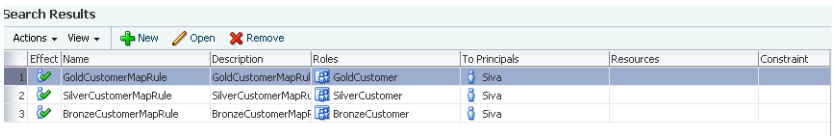
Figure 5–7 Searching for Role Mapping Policies



2. In the Search section, enter the query parameters as follows:
   - **Effect**: Select the policy effect (Grant/Deny) from the list.
   - **Display Name:** Select an operator from the list and enter a string to match.
   - **Name**: Select an operator from the list and enter a string to match.
   - **Role**: Select an operator from the list and enter a string to match.
   - **Principal**: Select an operator from the list and enter a string to match.
   - **Target**: Select an operator from the list and enter a string to match.
3. Click Search.

   All results matching the query specifications are displayed in the **Search Results** table as in Figure 5–8.

Figure 5–8 Role Mapping Policy Search Results



5.3.6 Searching Resources

A Resource can be hierarchical (a scenario in which the sub resource inherits attributes from the parent resource) or non-hierarchical. If a Resource is hierachical, its tiered-organization is shown in the Search results. To search Resources, proceed as follows:

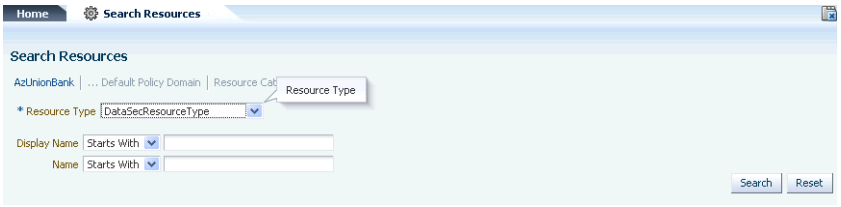
1. Select from the following methods to display the Search Resources page:
   - In the Navigation Panel, expand Applications and the named Application node applicable to the search. Expand the appropriate Policy Domain and Resource Catalog and double-click Resources.

     Alternately, right-click Resources and select Open.

- In the Home area, select the Application Name and click Search from Resources.

The Search Resources page is displayed as in Figure 5–9.

Figure 5–9 Searching for Resources



2. Enter the following query parameters:
   - **Resource Type**: Select a resource type from the list. This parameter is required.
   - **Display Name:** Select an operator from the list and enter a string to match.
   - **Name**: Select an operator from the list and enter a string to match.
3. Click Search.

   All results matching the query specifications are displayed in the **Search Results** table.

5.3.7 Searching Entitlements

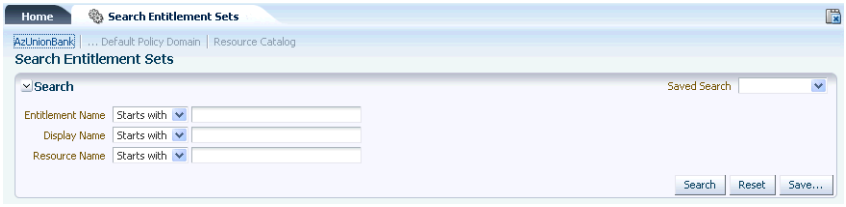
To search Entitlements, proceed as follows:

1. Select from the following methods to display the Search Entitlements page:
   - In the Navigation Panel, expand Applications and the named Application node applicable to the search. Expand the appropriate Policy Domain and Resource Catalog and double-click Entitlements.

     Alternately, right-click Entitlements and select Open.
   - In the Home area, select the Application Name and click Search from Entitlements. (In this case, the search is done only within the Default Policy Domain.)

   The Search Entitlements tab is displayed in the Home area as in Figure 5–10.

Figure 5–10 Searching for Entitlements



2. Enter the following query parameters:
   - **Entitlement Name:** Select an operator from the list and enter a string to match.
   - **Display Name:** Select an operator from the list and enter a string to match.

- **Resource name**: Select an operator from the list and enter a string to match.

Optionally, select from the Saved Search drop-down list of previously saved searches. Its query parameters automatically populate the search fields. Select Personalize... to set options for previously saved searches.

3. Optionally, click Save... to save the current query parameters as a Saved Search.

   The search is added to the Saved Search list.

4. Click Search.

   All results matching the query specifications are displayed in the Search Results table.

5.3.8 Searching Authorization Policies

Authorization Policies can be searched by specifying a policy name, a principal, or a target. To search Authorization Policies, proceed as follows:

1. Select from the following methods to display the Search Policies page:

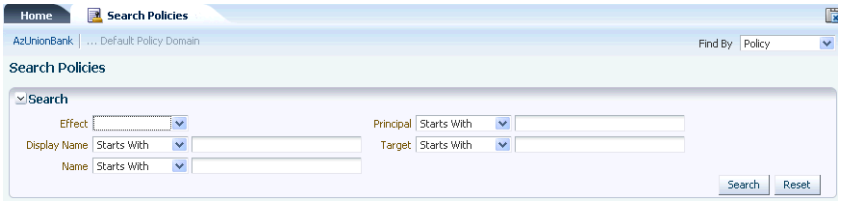
   - In the Navigation Panel, expand Applications and the named Application node applicable to the search. Expand the appropriate Policy Domain and Resource Catalog and double-click Authorization Policies.

     Alternately, right-click Authorization Policies and select Open.

   - In the Home area, select the Application Name, and click Search under Authorization Policies. (In this case, the search is done within the Default Policy Domain.)

   The Search Policies tab is displayed in Figure 5–11.

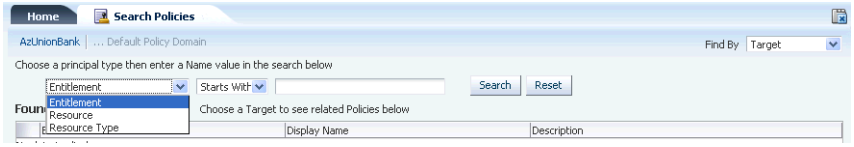
Figure 5–11 Searching Policies



2. Select the search type from the Find By list.

   The query parameters change according to the selection. Options include Policy, Principal or Target. Figure 5–11 is a screenshot in which Policy is selected. Figure 5–12 is a screenshot in which Target is selected.

Figure 5–12 Searching Polices by Target



3. Search using the option based on your previous selection.

   - To Find By: Policy, enter the following query parameters.

- **Effect:** Select the policy effect (Grant/Deny) from the list.
- **Display Name:** Select an operator from the list and enter a string to match.
- **Name:** Select an operator from the list and enter a string to match.
- **Principal:** Select an operator from the list and enter a string to match.
- **Target:** Select an operator from the list and enter a string to match.

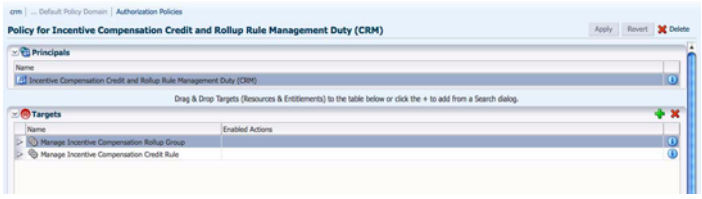
- To Find By: Principal or Find By: Target, select an operator from the list, and enter a string to match.

  A Resource Type must be provided if the Resource or Resource Type operator is selected.

4. Click Search.

   The Administration Console can display Authorization Policies created using Oracle Entitlements Server as well as the simpler Application Grants (system policies) created using Oracle Platform Security Services (OPSS). The OPSS Application Grants can be displayed for viewing, modification and deletion only. When created using OPSS, Application Grants are not given a policy name or description; they are defined with a principal and target only. Figure 5–13 is a screenshot of the Oracle Entitlements Server screen when an OPSS Application Grant is displayed.

Figure 5–13 OPSS Application Grants Display Screen



   Note the Name, Display Name and Description fields are not displayed as they would be if the Authorization Policy was created using Oracle Entitlements Server. OPSS Application Grants can only be removed or modified with Oracle Entitlements Server; they can not be created using Oracle Entitlements Server. For more information on Application Grants, see the *Oracle Fusion Middleware Application Security Guide*.

5.3.9 Searching Attributes

To search Attributes, proceed as follows:

1. In the Navigation Panel, expand Applications and the named Application node applicable to the search.
2. Expand Extensions and double-click Attributes to display the Search Attributes page.

   Alternately, right-click Attributes and select Open.
3. Enter the following query parameters.
   - **Display Name**: Select an operator from the list and enter a string to match.
   - **Name**: Select an operator from the list and enter a string to match.
   - **Type**: Select an operator from the list and enter a string to match.

Optionally, select from the Saved Search drop-down list of previously saved searches. Its query parameters automatically populate the search fields. Select Personalize... to set options for previously saved searches.

4. Optionally, click Save... to save the current query parameters as a Saved Search.

   The search is added to the Saved Search list.

5. Click Search.

5.3.10 Searching Functions

To search application functions, proceed as follows

1. In the Navigation Panel, expand Applications and the named Application node applicable to the search.
2. Expand Extensions and double-click Functions to display the Search Functions page.

   Alternately, right-click Functions and select Open.

3. Enter the following query parameters.

   - **Name**: Select an operator from the list and enter a string to match.
   - **Display Name**: Select an operator from the list and enter a string to match.

   Optionally, select from the Saved Search drop-down list of previously saved searches. Its query parameters automatically populate the search fields. Select Personalize... to set options for previously saved searches.

4. Optionally, click Save... to save the current query parameters as a Saved Search.

   The search is added to the Saved Search list.

5. Click Search.

5.3.11 Searching for Users

To search for Users, proceed as follows:

1. Select from the following methods to display the Search External Roles page:

   - In the Navigation Panel, expand Global and double-click Users. (Alternately, right-click Users and select Open.)
   - In the Home area, click Search - Users from the Search and Create section.

2. Enter the following query parameters:

   - **User Name**: Select an operator from the list and enter a string to match.
   - **Display Name**: Select an operator from the list and enter a string to match.

   Optionally, select from the Saved Search drop-down list of previously saved searches. Its query parameters automatically populate the search fields. Select Personalize... to set options for previously saved searches.

3. Optionally, click Save... to save the current query parameters as a Saved Search.

   The search is added to the Saved Search list.

4. Click Search.

   The results are displayed in Search Results.

5.4 Understanding Case Sensitivity in Object Names

This section provides information regarding the case sensitivity of names that define policy objects. The objects below are case sensitive. Those not listed are case insensitive.

- Principal (defined for an Administration Role or an Application Role)
- Grant Action
- Permission Class Name
- Resource Name
- Resource Type
- Resource Action
- Resource Name Expression
- Resource Type Resource Matcher
- Policy Action
- Policy Grantee

See Chapter 2, "Understanding the Policy Model" for information on the policy objects.

6 Managing Policy Distribution

Policy distribution is the process of distributing configured policies and policy objects from the Oracle Entitlements Server Administration Server to specified Security Modules. After a successful policy distribution, each Security Module to which the data was distributed can evaluate requests for access without the Adminministration Server being online. Evaluation of the policies will produce a *grant* or *deny* authorization decision in answer to a request for access. This chapter contains the following sections.

6.1 Defining Distribution Modes

The Oracle Entitlements Server Policy Distribution Component handles the task of distributing policies to individual Security Modules that protect applications and services. Policy data is distributed in either a *controlled* manner or a *non-controlled* manner. The distribution mode is defined in the `jps-config.xml` configuration file for each Security Module. The specified distribution mode is applicable for all Application objects bound to that Security Module. The following sections have more information.

6.1.1 Controlled Distribution

Controlled distribution is initiated by the Policy Distribution Component, ensuring that the PDP client (Security Module) receives policy data that has been created or modified since the last distribution. In this respect, distribution is controlled by the policy administrator who takes explicit action to distribute the new or updated policy data. (The Policy Distribution Component maintains a versioning mechanism to keep track of policy changes and distribution.) When controlled distribution is enabled, the Security Module cannot request distribution of the Policy Distribution Component directly.

Note: The exception is when a Security Module starts and registers itself with the Policy Distribution Component with a Configuration ID. As long as the appropriate Application bound to the Security Module is marked as (policies are) Ready For Distribution, the policies are distributed to the Security Module based on this registration.

With controlled distribution, the Policy Distribution Component distributes new and updated policy data to the Security Module where the data is stored in a local persistent cache, a file-based cache maintained by the PDP to store policy objects and provide independence from the policy store. The Policy Distribution Component does not maintain constant live connections to its Security Module clients; it will establish a connection before distributing policy to it. Thus, the Security Module is not dependent on the policy store for making policy decisions; it can use its own local cache if the policy store is offline. When the Security Module starts, it will check if the policy store is available. If it is not available, the Security Module will use policy data from the local persistent cache. With controlled distribution, if any policy distribution operation fails, the entire policy distribution fails.

Caution: Controlled distribution is supported only on database type policy stores - not on LDAP-based policy stores. If the distribution API is invoked for an LDAP policy store, it will be non-operable.

6.1.2 Non-controlled Distribution

When the PDP client (Security Module) periodically retrieves (or pulls) policies and policy modifications from a policy store, it is referred to as non-controlled distribution. Non-controlled distribution makes policy changes available as soon as they are saved to the policy store. Non-controlled distribution is initiated by the Security Module and may retrieve policies that are not yet complete. The policy store must be online and constantly available for non-controlled distribution. Non-controlled distribution is supported on any policy store type including a relational database, an LDAP directory or XML.

6.2 Understanding Policy Distribution

Managing policies and distributing them are distinct operations. Policy management operations are used to define, modify and delete policies in the policy store. The Policy Distribution Component then makes the policies available to a Security Module where the data is used to grant or deny access to a protected resource. Policies can not be enforced until they are distributed. Conceptually, policy distribution may include any or all of the following actions:

- Reading policies from the policy store.
- Caching policy objects in the in-memory policy cache maintained by the Security Module for use during authorization request processing.
- Perserving policy objects in a file-based persistent cache, local to the Policy Distribution Component, that provides independence from the policy store.

Both the central Oracle Entitlements Server Administration Console and the locally-installed (to the protected application) Security Module contain the Policy Distribution Component. This architecture allows two deployment scenarios: the first involves a centralized Policy Distribution Component that can communicate with

many Security Modules while the second involves a Policy Distribution Component that is local to, and communicates with, only one Security Module.

> **Note:** For details on configuring the Security Module parameters for policy distribution, see Section A.1, "Policy Distribution Configuration." For details on creating definitions and binding Security Modules, see Section 10.2, "Configuring Security Module Definitions."

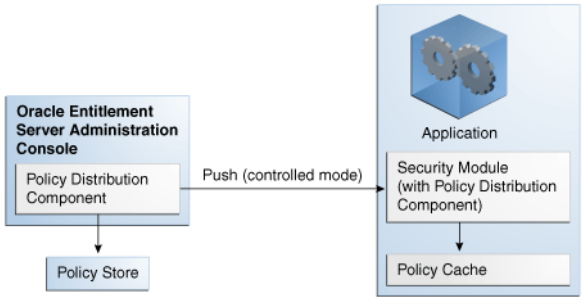
The following sections contain more information.

- Section 6.2.1, "Using a Central Policy Distribution Component"
- Section 6.2.2, "Using a Local Policy Distribution Component"

6.2.1 Using a Central Policy Distribution Component

The centralized Policy Distribution Component scenario involves the use of the Policy Distribution Component (within the Administration Console) to act as a server communicating with the Security Module's Policy Distribution Component client. Figure 6–1 illustrates how, in this scenario, the Security Module's Policy Distribution Component client does not communicate with the policy store. The distribution of policies is initiated by the Oracle Entitlements Server administrator and *pushed* to the Policy Distribution Component client. Currently, data can only be pushed in a *controlled* manner as described in Section 6.1.1, "Controlled Distribution." This scenario allows for a central Policy Distribution Component that can communicate with many Security Modules.

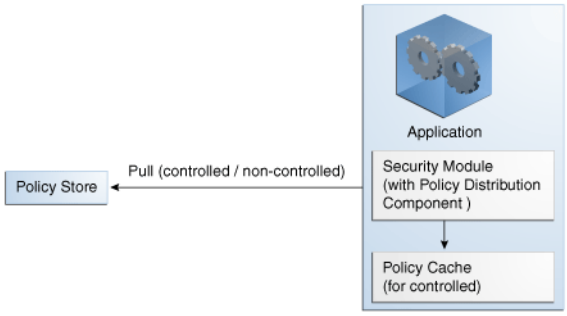
Figure 6–1 Using Oracle Entitlements Server Policy Distribution Component



6.2.2 Using a Local Policy Distribution Component

The local (to the Security Module) scenario involves the Security Module's Policy Distribution Component communicating directly with the policy store. This scenario allows for a local Policy Distribution Component to communicate with one Security Module only. The application administers management operations and decides when the Security Module instance of the Policy Distribution Component will distribute policies or policy deltas. In this deployment, as illustrated in Figure 6–2, the Policy Distribution Component *pulls* data from the policy store (by periodically checking the policy store for data to be distributed) and sends policy data from the policy store, making it available to the PDP after administrator-initiated policy distribution.

Figure 6–2 Using Security Module Policy Distribution Component



Currently, data can be pulled in either a controlled manner as described in Section 6.1.1, "Controlled Distribution" or a non-controlled manner as described in Section 6.1.2, "Non-controlled Distribution."

6.3 Distributing Policies

From a high level, the following steps are needed to get to the point where you can distribute policies. Distributing policies using these steps (from the Administration Console or with the API) will work for Controlled Distribution only. Security Modules configured for Non-controlled Distribution retrieve policy data from the policy store directly and don't require explicit distribution.

1. Create a Security Module definition.

   See Section 10.2.1, "Creating a Security Module Definition."

2. Bind the definition to the appropriate Application.

   See Section 10.2.2, "Binding an Application to a Security Module." To unbind the Security Module, see Section 10.2.3, "Unbinding an Application From a Security Module."

3. Open the Application in the Home area.

   See Section 6.3.1, "Distributing Policies Using the Administration Console."

4. Distribute the policies.

   See Section 6.3.1, "Distributing Policies Using the Administration Console."

6.3.1 Distributing Policies Using the Administration Console

Policies are distributed from within an Application. The following procedure documents how to distribute policies using the Administration Console.

1. Expand the Applications node in the Navigation Panel.
2. Select the Application which contains the policies for distribution.
3. Right-click the Application name and select Open from the menu.

   The General tab, the Delegated Administrators tab and the Policy Distribution tab are all active.

4. Click the Policy Distribution tab.
5. Click Distribute.

   Policies are distributed to all Security Modules bound to the Application.

6. Click Refresh to update the distribution progress.

6.4 Using Default or Third Party Digital Certificates

Secure Sockets Layer (SSL) uses public key encryption technology to establish and verify server identity and trust. With public key encryption, a public key and a private key are generated for the server. Data encrypted with the public key can only be decrypted with the corresponding private key and data encrypted with the private key can only be decrypted with the corresponding public key. The private key is protected so that only the owner can decrypt messages encrypted using the public key. The public key is embedded in a digital certificate with additional information describing the owner of the public key (such as name, street address and e-mail address).

Note: The data embedded in the digital certificate is verified by a certificate authority (CA) and digitally signed with the CA's trusted digital certificate. The CA's trusted digital certificate establishes trust for the application's digital certificate itself.

An application participating in an SSL connection is authenticated when the other party evaluates and accepts its digital certificate. Web browsers, servers, and other SSL-enabled applications generally accept as genuine any digital certificate that is signed by a trusted certificate authority and is otherwise valid.

During the creation of a Security Module instance (as documented in the Oracle Fusion Middleware Installation Guide for Oracle Identity and Access Management), the default Oracle Entitlements Server Security Module (client) digital certificate is stored in *your_oes_sm_folder*/oes_sm_instances/*your_oes_sm*/security/identity.jks and the trusted Certificate Authority (CA) digital certificates are stored in *your_oes_sm_folder*/oes_sm_instances/*your_oes_sm*/security/trust.jks. Both are Java Key Stores (JKS) with the passwords defined during the creation of the Security Module instance. The password will be encrypted and stored in the standard Oracle Wallet (with autologon). The default Oracle Entitlements Server client keys are generated by itself and signed during the enrollment process. The following sections document how to use third party certificates with the applicable security module.

- Using a Third Party Certificate with a WebLogic Server Security Module
- Using a Third Party Certificate with a Web Services or Java Security Module
- Using a Third Party Certificate with a Websphere Application Server Security Module
- Using a Third Party Certificate with a Tomcat or JBoss Security Module

Note: By default, an installation of WebLogic Server creates and uses DemoIdentity.jks and DemoTrust.jks as the default identity and trust stores, respectively. The full paths to the locations of these stores can be found using the WebLogic Server console. The default password for the trust store is *DemoTrustKeyStorePassPhrase*.

6.4.1 Using a Third Party Certificate with a WebLogic Server Security Module

The following procedure illustrates how to use a third party certificate with a WebLogic Server Security Module.

1. Import the third party generated certificate to the WebLogic Server trust store using this command.

   ```
   keytool -import -alias keyAlias -file cert_file -keystore ./DemoTrust.jks
    -storepass DemoTrustKeyStorePassPhrase -storetype JKS -keypass password
   ```

   WebLogic Server uses `DemoIdentity.jks` and `DemoTrust.jks` as the default identity and trust stores, respectively.

2. Import the administration CA certificate to the client trust store using the `keytool` command line interface.

   The following command imports the CA certificate in the default trust store to a client trust store. The absolute path to the client trust store is defined in the WebLogic Administration Console. Modify the parameters in the command to reflect your deployment.

   ```
   keytool -importkeystore -srckeystore ./DemoTrust.jks
    -destkeystore yourTrustStore -srcstorepass DemoTrustKeyStorePassPhrase
    -deststorepass passwordForYourTrustStore
    -srcalias wlscertgencab -destalias wlscertgencab
   ```

   This step is not needed if using the default trust store.

3. Configure a third party key store using the WebLogic Server Administration Console in the WebLogic Server domain for the appropriate Security Module.

   a. In the left pane of the Administration Console, expand Environment and select Servers.

   b. Click the name of the server for which you want to configure the identity and trust keystores.

   c. Click Configuration -> Keystores.

   d. Select the method for storing and managing private key/digital certificate pairs and trusted CA certificates in the Keystores field. The available options are described in Table 6–1.

Table 6–1 Storing and Managing Certificate Options

Demo Identity and Demo Trust: The demonstration identity and trust keystores (located in the *WLS_HOME*\server\lib directory) and the JDK cacerts keystore (located in the *JAVA_HOME*\jre\lib directory) are configured by default. Use for development only.
Custom Identity and Java Standard Trust: A keystore you create and the trusted CAs defined in the cacerts file (located in the *JAVA_HOME*\jre\lib directory) directory are configured.
Custom Identity and Custom Trust: Identity and trust keystores you create are configured.
Custom Identity and Command Line Trust: An identity keystore you create and command-line arguments that specify the location of the trust keystore are configured.

   e. Define values for the identity keystore attributes in the Identity section. Table 6–2 defines the attributes.

Table 6–2 Using the Identity Keystore Attributes

Custom Identity Keystore: The fully qualified path to the identity keystore.
Custom Identity Keystore Type: The type of the keystore. Generally, this attribute defines a Java KeyStore (JKS); if left blank, it defaults to JKS.

Table 6–2 (Cont.) Using the Identity Keystore Attributes

Custom Identity Keystore Passphrase: The password to be entered when reading or writing to the keystore. This attribute is optional or required depending on the type of keystore. All keystores require the passphrase in order to write to the keystore however, some keystores do not require the passphrase to read from the keystore. WebLogic Server only reads from the keystore so whether or not you define this property depends on the requirements of the keystore. Note: The passphrase for the Demo Identity keystore is DemoIdentityKeyStorePassPhrase.

f. Define values for the trust keystore attributes in the Trust section.

If you chose Java Standard Trust as your keystore (as documented in Table 6–1), specify the password defined when creating the keystore and confirm it. If you chose Custom Trust, define the attributes documented in Table 6–3.

Table 6–3 Using Trust Keystore Attributes

Custom Trust Keystore: The fully qualified path to the trust keystore.
Custom Trust Keystore Type: The type of the keystore. Generally, this attribute is JKS; if left blank, it defaults to JKS.
Custom Trust Keystore Passphrase: The password you will enter when reading or writing to the keystore. This attribute is optional or required depending on the type of keystore. All keystores require the passphrase in order to write to the keystore. However, some keystores do not require the passphrase to read from the keystore. WebLogic Server only reads from the keystore so whether or not you define this property depends on the requirements of the keystore.

g. Click Save.

h. Click Activate Changes in the Change Center of the Administration Console.

Some changes require a server restart.

6.4.2 Using a Third Party Certificate with a Web Services or Java Security Module

The following procedure illustrates how to use a third party certificate with a Web Services or Java Security Module.

1. Import the third party generated certificate to the administration trust store using this command.

```
keytool -import -alias keyAlias -file cert_file -keystore ./DemoTrust.jks
 -storepass DemoTrustKeyStorePassPhrase -storetype JKS -keypass password
```

By default, the administration server uses `DemoIdentity.jks` and `DemoTrust.jks` as the default identity and trust stores, respectively.

2. Import the administration CA certificate to the client trust store using the `keytool` command line interface.

The following command imports the CA certificate in the default trust store to a client trust store. The absolute path to the client trust store is defined in the WebLogic Administration Console. Modify the parameters in the command to reflect your deployment.

```
keytool -importkeystore -srckeystore ./DemoTrust.jks
 -destkeystore yourTrustStore -srcstorepass DemoTrustKeyStorePassPhrase
 -deststorepass passwordForYourTrustStore
 -srcalias wlscertgencab -destalias wlscertgencab
```

This step is not needed if using the default trust store.

3. Configure the following properties in `jps-config.xml`.
   - `oracle.security.jps.pd.client.ssl.identityKeyStoreFileName`
   - `oracle.security.jps.pd.client.ssl.trustKeyStoreFileName`
   - `oracle.security.jps.pd.client.ssl.identityKeyStoreKeyAlias`

   See Appendix A, "Installation and Configuration Parameters" for details.
4. Run `oesPassword.sh` in the appropriate *`SM-instance`*`/bin` directory to save the trust store password, the key store password and the key password into the credential store.
   - If the trust store, key store and key all have the same password, run `./oesPassword.sh`.
   - If the trust store, key store and key all have different passwords, run `./oesPassword.sh -diffpass`. You will be asked to enter (and confirm) the passwords for the `identity.jks` key store, the `trust.jks` trust store and the key.

The password for the default key store (identity store), each of the key store's disparate key entries, and the default trust store is *welcome1*.

6.4.3 Using a Third Party Certificate with a Websphere Application Server Security Module

The following procedure illustrates how to use a third party certificate with a Websphere Application Server Security Module.

1. Import the third party generated certificate to the administration trust store using this command.

   ```
   keytool -import -alias keyAlias -file cert_file -keystore ./DemoTrust.jks
    -storepass DemoTrustKeyStorePassPhrase -storetype JKS -keypass password
   ```

   By default, the administration server uses `DemoIdentity.jks` and `DemoTrust.jks` as the default identity and trust stores, respectively.
2. Import the administration CA certificate to the client trust store using the `keytool` command line interface.

   The following command imports the CA certificate in the default trust store to a client trust store. The absolute path to the client trust store is defined in the WebLogic Administration Console. Modify the parameters in the command to reflect your deployment.

   ```
   keytool -importkeystore -srckeystore ./DemoTrust.jks
    -destkeystore yourTrustStore -srcstorepass DemoTrustKeyStorePassPhrase
    -deststorepass passwordForYourTrustStore
    -srcalias wlscertgencab -destalias wlscertgencab
   ```

   This step is not needed if using the default trust store.
3. Configure a third party key store and trust store using the Websphere Application Server console located at *`servername`*`:`*`port`*`/ibm/console`.

   Information on configuring the IBM WebSphere Security Module itself is documented in the *Oracle Fusion Middleware Installation Guide for Oracle Identity and Access Management*.

a. Export the WebLogic Server `DemoTrust.jks` keystore or Oracle Entitlements Server `trust.jks` into a .DER file using the `keytool` command line interface.

```
keytool -exportcert -keystore yourtruststore -alias wlscertgencab
 -file ~/was.der
```

b. Import generated `was.der` file into the WebSphere default trust keystore and cell default trust keystore nodes.

   a. Access the WebSphere Administration Server console Signer certificates page by navigating through Security -> SSL certificate and key management -> Key stores and certificates -> *choose the appropriate <NodeDefaultTrustStore> or <CellDefaultTrustStore> name* -> Signer certificates.

   b. Click Add and enter an alias; for example, wlscertgencab.

   c. Choose the exported `was.der` file and select data type as DER.

c. Import the issued private key into the IBM WebSphere node default keystore.

   a. Access the WebSphere Administration Server console Personal certificates page by navigating through Security -> SSL certificate and key management -> Key stores and certificates -> NodeDefaultKeyStore -> Personal certificates.

   b. Click Import.

   c. Select Keystore and enter the path to your keystore file.

   d. Select JKS as the type and enter the password used to create the keystore file.

   e. Enter a certificate alias name.

d. Enable Inbound SSL for the server on which the WebSphere Security Module is running.

   a. Access the WebSphere Administration Server console Manage endpoint security configurations page by navigating through Security -> SSL certificate and key management -> Manage endpoint security configurations.

   b. Expand the inbound tree by navigating through Inbound -> DefaultCell(CellDefaultSSLSettings) -> nodes -> DefaultCellFederatedNode -> servers and select the *server name on which the IBM WebSphere Security Module is running*.

   c. Select Override inherited values on the General Properties page.

   d. Select NodeDefaultSSLSettings from the SSL configuration list.

   e. Choose the new, imported private key alias in the Certificate alias.

      See *Oracle Fusion Middleware Installation Guide for Oracle Identity and Access Management* for details.

   f. Click Apply.

e. Enable outbound SSL for the server on which the WebSphere Security Module is running.

   a. Access the WebSphere Administration Server console Manage endpoint security configurations page by navigating through Security -> SSL

certificate and key management -> Manage endpoint security configurations.

b. Expand the outbound tree by navigating through Outbound -> DefaultCell(CellDefaultSSLSettings) -> nodes -> DefaultCellFederatedNode -> servers and select the *server name on which the IBM WebSphere Security Module is running*.

c. Select Override inherited values on the General Properties page.

d. Select NodeDefaultSSLSettings from the SSL configuration list.

e. Click the Update certificate alias list button and choose the new, imported private key alias in the Certificate alias from the key store list.

f. Click Apply.

6.4.4 Using a Third Party Certificate with a Tomcat or JBoss Security Module

The following procedure illustrates how to use a third party certificate with a Tomcat or JBoss Application Server Security Module.

1. Import the third party generated certificate to the administration trust store using this command.

```
keytool -import -alias keyAlias -file cert_file -keystore ./DemoTrust.jks
 -storepass DemoTrustKeyStorePassPhrase -storetype JKS -keypass password
```

   By default, the administration server uses `DemoIdentity.jks` and `DemoTrust.jks` as the default identity and trust stores, respectively.

2. Import the administration CA certificate to the client trust store using the `keytool` command line interface.

   The following command imports the CA certificate in the default trust store to a client trust store. The absolute path to the client trust store is defined in the WebLogic Administration Console. Modify the parameters in the command to reflect your deployment.

```
keytool -importkeystore -srckeystore ./DemoTrust.jks
 -destkeystore yourTrustStore -srcstorepass DemoTrustKeyStorePassPhrase
 -deststorepass passwordForYourTrustStore
 -srcalias wlscertgencab -destalias wlscertgencab
```

   This step is not needed if using the default trust store.

3. Set the keystoreFile and truststoreFile properties for the SSL connector in the `server.xml` file. They include:

   - keystoreFile : location of your key store
   - keystorePass : password for your key store
   - keyAlias : certificate alias in the key store
   - keyPass : password for the certificate alias
   - truststoreFile : location of your trust store
   - truststorePass : password for the trust store

   For the Tomcat Application Server Security Module, `server.xml` is located in the *`TOMCAT_HOME`*`/conf/` directory. For the JBoss Application Server Security Module, the location is dependent on which profile is used; for example, *`JBOSS_HOME`*`/server/`*`jbossProfileName`*`/deploy/jbossweb.sar/server.xml`.

6.5 Debugging Policy Distribution

Information on debugging the Policy Distribution Component is documented in Section 13.8.3, "Debugging Policy Distribution."

7

Deploying the Policy Decision Point

This appendix contains information on why and how the Policy Decision Point (PDP) is deployed. The following sections have more information.

- Understanding the PDP Deployment Models
- Using the Security Module Proxy Mode
- Using the XACML Gateway

7.1 Understanding the PDP Deployment Models

Oracle Entitlements Server supports deployment models that differ in the placement of the Policy Decision Point (PDP), depending on where the request for access will be evaluated. Java clients, for example, have the option of making a local call to the Security Module embedded with the application, or making remote calls to a Remote Method Invocation (RMI) or Web Services Security Module via a proxy. These models are not mutually exclusive; any combination of them is possible.

The PDP can be embedded locally with the application or it can be installed elsewhere (outside the application) for remote communication. Real time or near real time applications (stock trading or banking applications, respectively) would typically use the embedded model (where Security Modules are distributed) for a good response time. It should also be used if the application needs to work offline (no connectivity to the back end). A remote PDP (where one central Security Module commuicates with mulitple clients) can be used when there is a need to minimize the number of PDPs or if security concerns prevent deploying the PDP next to the application (like in a DMZ). Additionally, it can be used if overhead related to remote invocation is acceptable or if the PDP is invoked from an application written in a programming language other than Java (C, C++, C#).

> **Note:** Proxy Mode is used to configure a Security Module to send authorization requests to a remote PDP. It provides the ability to issue a request for access and receive the decision of GRANT or DENY. See Section 7.2, "Using the Security Module Proxy Mode."

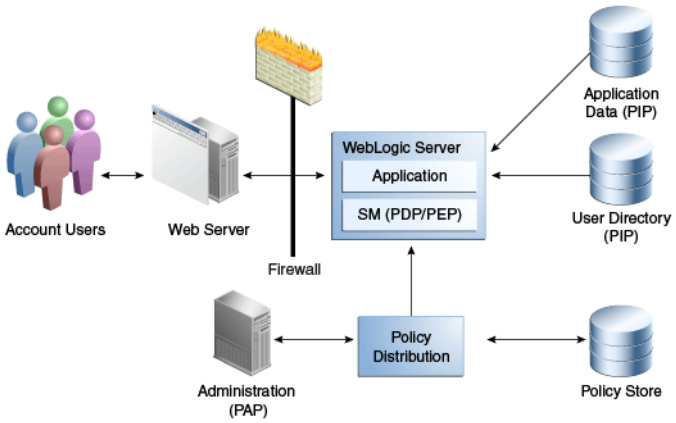
The following sections have additional information.

- Section 7.1.1, "Embedding the PDP Locally"
- Section 7.1.2, "Locating the PDP Remotely"

7.1.1 Embedding the PDP Locally

In the distributed deployment model, the Security Module/PDP is embedded in the protected application's container and runs in the same process. Policies are distributed to the Security Module and authorization decisions are made and cached locally so no network calls to a central server are needed. Additionally, no changes to the application itself are required. A typical embedded PDP deployment is illustrated in Figure 7–1.

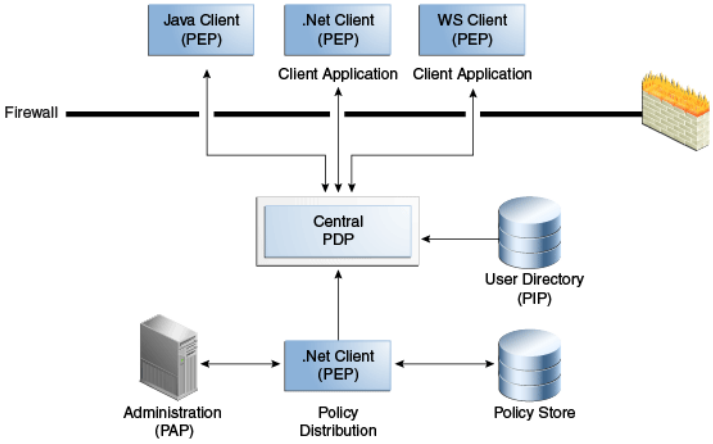
Figure 7–1 Embedded PDP Deployment



7.1.2 Locating the PDP Remotely

In the centralized deployment model, the PDP runs in a process that is separate from the protected application and is hosted in a location that is remote from the protected resources. The central PDP (Oracle Entitlements Server or a Security Module) receives authorization requests from multiple clients and then evaluates the applicable policies, returning the authorization decision (GRANT or DENY) to the PEP to enforce. Figure 7–2 illustrates this centralized deployment model.

Figure 7–2 Remote PDP Deployment

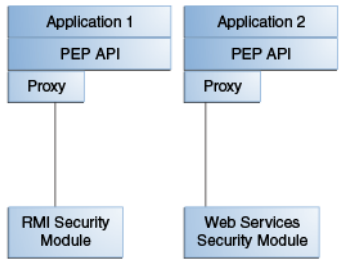


The centralized PDP is invoked using the XACML Gateway, RMI, or SOAP. It maintains local caches of roles, policies, policy decisions and attributes which can be updated on demand or at startup.

7.2 Using the Security Module Proxy Mode

Oracle Entitlements Server supports a Proxy Mode that allow clients to invoke authorization services remotely. When the Oracle Entitlements Server PEP API are used by an application to make authorization calls, a proxy can be set up on the application's host to communicate with the remote Security Module (on another machine) for access decisions. Web services or Remote Method Invocation (RMI) are the proxy's communication options. Figure 7–3 illustrates this topology.

Figure 7–3 Proxy Mode Deployment

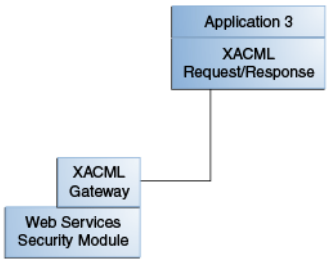


The Oracle Entitlements Server PEP API can be called from an application regardless of whether that application is calling an embedded PDP or a remote PDP. Proxy Mode can be configured to provide security services locally (including authorization caching, logging and failover) and can communicate with the PDP using RMI or SOAP.

7.3 Using the XACML Gateway

The XACML Gateway is a SOAP endpoint that uses web services functionality and provides for communication using the XACML request/response protocol. The XACML Gateway is with the PDP (not the client) and is available with the instantiation of the Web Services Security Module. No special configuration is required to use the XACML Gateway. The Web Services Security Module will use a web services call when instantiated locally with the client. If it sees a SOAP envelope at its endpoint, it uses the XACML Gateway. Figure 7–4 illustrates a XACML Gateway deployment.

Figure 7–4 XACML Gateway Deployment



Note: If the application is using the XACML request/response protocol, it is not using the PEP API.

8

Managing Security Module Configurations

The Security Module Configuration User Interface (SMConfig UI) is used to configure a Security Module after the OES Client has been successfully installed and a Security Module type has been instantiated. Basic configurations are accomplished during the installation and instantiation processes but the SMConfig UI allows for additional configurations or modifications to a Security Module profile. This chapter contains the following sections.

8.1 Before You Begin

The SMConfig UI assumes the following Oracle Entitlements Server installation procedures have already been completed.

- For information on installing Oracle Entitlements Server, see *Oracle Fusion Middleware Installation Guide for Oracle Identity and Access Management*.
- For information on installing the OES Client and instantiating a Security Module, see *Oracle Fusion Middleware Installation Guide for Oracle Identity and Access Management*.

The OES Client installs the Security Module bits and the SMConfig Tool. The SMConfig Tool will define the Security Module type and create a directory structure for each Security Module instance. The instance's home directory is created at *$OES_CLIENT_HOME*`/oes_sm_instances/`*SM_Name*`/` and contains the configuration files used to run the applicable Security Module SMConfig UI script; `oessmconfig.sh` and `oessmconfig.bat` are located in the `/bin` directory of the Security Module instance's home directory.

> **Note:** Beginning with Oracle Entitlements Server 11gR2, a Security Module can be installed in a Fusion Middleware domain, version 11gR1 PS5 (11.1.1.6.0) or above. Also, when an Oracle Entitlements Server Security Module is installed into a JRF domain, only non-controlled mode is supported.

8.2 Starting the SMConfig UI

The SMConfig UI is a standalone Java-based tool distributed with the OES Client installation package. The tool configures the Policy Decision Point (PDP), the Policy Information Point (PIP), and the policy store, among other data. The SMConfig UI script is available in the directory created during the instantiation of a Security Module; for example, *$OES_CLIENT_HOME*/oes_sm_instances/*SM_Name*/bin/.

> **Note:** The SMConfig UI script will run only after SMConfig Tool has exited. See Section 8.1, "Before You Begin" for details.

In a Linux environment the SMConfig UI script is named `oessmconfig.sh` and in a Windows environment it is called `oessmconfig.bat`. To start the SMConfig UI change to the `bin` directory in the appropriate Security Module instance directory and run the script on the command line.

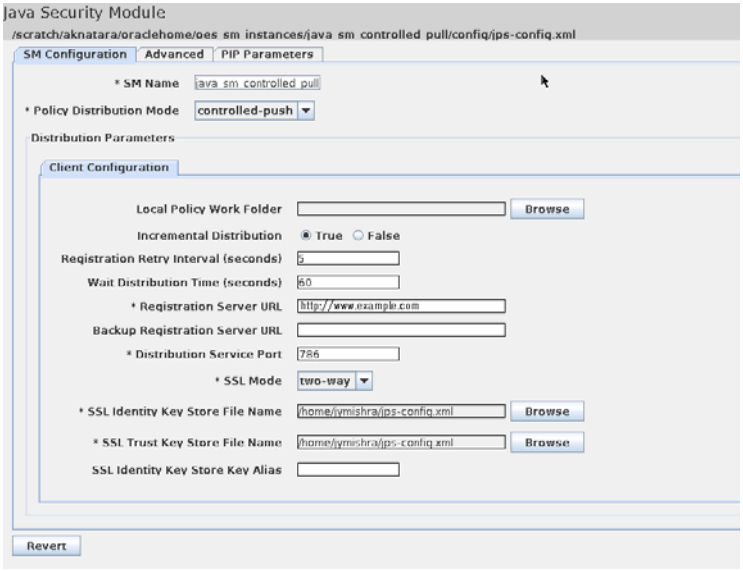
```
cd $OES_CLIENT_HOME/oes_sm_instances/SM_Name/bin/
./oessmconfig.sh
```

The SMConfig UI runs on the same host as the Security Module. The script will modify the Security Module's `jps-config.xml` file. The location of the `jps-config.xml` can be specified as an input parameter when first creating the Security Module. If this parameter is not specified, the default file (located in the Security Module's home directory at *$OES_CLIENT_HOME*/oes_sm_instances/*SM_Name*/config) is used.

> **Note:** For the WebLogic Server Secuity Module, specify the `jps-config.xml` file located under `user_projects/domains/`*domain_name*`/config/oeswlssmconfig/AdminServer` when using `oessmconfig.sh`. Pass the file's location to the tool using the `-jpsconfig` parameter. For more information, enter the following:
>
> ```
> ./oessmconfig.sh -help
> ```

Figure 8–1 is a screenshot of the SMConfig UI for a Java Security Module.

Figure 8–1 SMConfig UI for Java Security Module



Depending on the type of Security Module, you can expect to configure some or all of the following:

- If installed in non-controlled mode, configure the Policy Store Type as either LDAP, Oracle DB, or XML.
  - If a database, define a location URL and database credentials, and verify database connectivity.
  - If an LDAP store, define a location URL and LDAP credentials, and verify LDAP connectivity.
- If installed in controlled push mode, configure the host, distribution port, listener port, username and password for Policy Distribution.
- If installed in Proxy Mode, configure the communication protocol, and host and distribution point of the Security Module.

8.3 Modifying Security Module Configurations

The SMConfig UI allows an administrator to fine-tune parameter values for a specific Security Module instantiation. This includes the following groups of parameters.

- PIP parameters refer to information that is used with Attribute Retrievers. PIP parameter modification includes managing values for predefined LDAP and RDBMS Attribute Retrievers as well as custom Attribute Retrievers and individual attributes that may be created and associated with a particular Attribute Retriever. See Table 8–5, " Java Security Module PIP Parameters (Attribute Retrievers)" and Table 8–6, " Java Security Module PIP Parameters (Attributes)" for details.

Note: If the Security Module is configured in Proxy Mode, PIP parameter modification is disabled.

- Many Security Module parameters are tailored to a specific Security Module type. These parameters are documented in Section A.2, "Security Module Configuration."
- Parameters associated with policy distribution are documented as follows:
  - If distributing policies in uncontrolled mode, see Section A.4, "Policy Store Service Configuration."
  - If distributing policies in a controlled push mode to a Java, RMI or Web Services Security Module, see Section A.1.2.1, "Policy Distribution Component Client Java Standard Edition Configuration (Controlled Push Mode)."
  - If distributing policies in a controlled push mode to a WebLogic Server, Websphere Application Server, JBoss or Tomcat Security Module, see Section A.1.2.2, "Policy Distribution Component Client Java Enterprise Edition Container Configuration (Controlled Push Mode)."
  - If distributing policies in controlled pull mode, see Section A.4, "Policy Store Service Configuration."
- Enabling Proxy Mode entails specifying the oracle.security.jps.pdp.PDPTransport parameter. For Proxy Mode parameters, see Section A.3, "PDP Proxy Client Configuration."

> **Note:** The SMConfig UI can not change the type of Security Module. To create another type of Security Module, execute the SMConfig Tool and create a new Security Module instance.

8.4 Configuring Security Modules Post-Instantiation

The following sections have information regarding the parameters, specific to the Security Module type, that can be modified post-instantiation. See Appendix A, "Installation and Configuration Parameters" for detailed descriptions.

- Section 8.4.1, "Configuring the Java Security Module"
- Section 8.4.2, "Configuring the RMI Security Module"
- Section 8.4.3, "Configuring the Web Services Security Module"
- Section 8.4.4, "Configuring the WebLogic Server Security Module"
- Section 8.4.5, "Configuring the SharePoint Server (MOSS) Security Module"
- Section 8.4.6, "Configuring the .NET Security Module"
- Section 8.4.7, "Configuring the WebSphere, Tomcat and JBoss Security Modules"
- Section 8.4.8, "Configuring the Oracle Service Bus Security Module"

8.4.1 Configuring the Java Security Module

Table 8–1 documents the controlled-push Client Configuration properties for the Java Security Module.

Table 8–1 Java Security Module Controlled-Push Client Configuration

Property Name	jps-config.xml Property
SM Name	oracle.security.jps.runtime.pd.client.sm_name
Policy Distribution Mode	oracle.security.jps.runtime.pd.client.policyDistributionMode

Table 8–1 (Cont.) Java Security Module Controlled-Push Client Configuration

Property Name	jps-config.xml Property
Client Configuration	■ Local Policy Work Folder: oracle.security.jps.runtime.pd.client.localpolicy.work_folder ■ Incremental Distribution: oracle.security.jps.runtime.pd.client.incrementalDistribution ■ Registration Retry Interval: oracle.security.jps.runtime.pd.client.registrationRetryInterval ■ Wait Distribution Time: oracle.security.jps.runtime.policyDistributionWaitTime ■ Registration Server URL: oracle.security.jps.runtime.pd.client.RegistrationServerURL ■ Backup Registration Server URL: oracle.security.jps.runtime.pd.client.backupRegistrationServerURL ■ Distribution Service Port: oracle.security.jps.runtime.pd.client.DistributionServicePort ■ SSL Mode: oracle.security.jps.pd.client.sslMode ■ SSL Identity Key Store File Name: oracle.security.jps.pd.client.ssl.identityKeyStoreFileName ■ SSL Trust Key Store File Name: oracle.security.jps.pd.client.ssl.trustKeyStoreFileName ■ SSL Identity Key Store Key Alias: oracle.security.jps.pd.client.ssl.identityKeyStoreKeyAlias

Table 8–2 documents the controlled-pull Client Configuration properties for the Java Security Module. The configuration properties are organized under the Client Configuration tab and the Policy Store tab.

Table 8–2 Java Security Module Controlled-Pull Client and Store Configuration

Property Name	jps-config.xml Property
SM Name	oracle.security.jps.runtime.pd.client.sm_name
Policy Distribution Mode	oracle.security.jps.runtime.pd.client.policyDistributionMode
Client Configuration	■ Local Policy Work Folder: oracle.security.jps.runtime.pd.client.localpolicy.work_folder ■ Incremental Distribution: oracle.security.jps.runtime.pd.client.incrementalDistribution ■ Registration Retry Interval: oracle.security.jps.runtime.pd.client.registrationRetryInterval ■ Wait Distribution Time: oracle.security.jps.runtime.policyDistributionWaitTime ■ Polling Timer: oracle.security.jps.pd.client.PollingTimerEnabled ■ Polling Timer Interval: oracle.security.jps.pd.client.PollingTimerInterval

Table 8–2 (Cont.) Java Security Module Controlled-Pull Client and Store Configuration

Property Name	jps-config.xml Property
Policy Store	■ Policy Store Type: policystore.type takes a value of DB_ORACLE as controlled distribution works only with a database ■ Database Configuration Through URL ■ JDBC URL: jdbc.url ■ JDBC Driver: jdbc.driver ■ Username: security.principal ■ Password: security.credential ■ Database Configuration Through JNDI Name ■ Datasource JNDI Name: datasource.jndi.name ■ Maximum Search Filter Length: max.search.filter.length defines the maximum length of a search filter. Takes as a value an integer; for example, 1024. ■ Farm Name: oracle.security.jps.farm.name ■ Resource Type Enforcement Mode: oracle.security.jps.policystore.resourcetypeenforcementmode

Table 8–3 documents the non-controlled distribution properties for the Java Security Module.

Table 8–3 Java Security Module Non-controlled Policy Store Configuration

Property Name	jps-config.xml Property
SM Name	oracle.security.jps.runtime.pd.client.sm_name
Policy Distribution Mode	oracle.security.jps.runtime.pd.client.policyDistributionMode

Table 8–3 (Cont.) Java Security Module Non-controlled Policy Store Configuration

Property Name	jps-config.xml Property
Policy Store	Policy Store Type: policystore.type takes a value of OID or DB_ORACLE. **DB** ■ Database Configuration Through URL ■ JDBC URL: jdbc.url ■ JDBC Driver: jdbc.driver ■ Username: security.principal ■ Password: security.credential ■ Database Configuration Through JNDI Name ■ Datasource JNDI Name: datasource.jndi.name ■ Maximum Search Filter Length: max.search.filter.length defines the maximum length of a search filter. Takes as a value an integer defining the maximum length; for example, 1024. ■ Farm Name: oracle.security.jps.farm.name ■ Resource Type Enforcement Mode: oracle.security.jps.policystore.resourcetypeenforcementmode **LDAP** ■ LDAP URL: ldap.url defines the location of the LDAP policy store ■ Maximum Search Filter Length: max.search.filter.length defines the maximum length of a search filter. Takes as a value an integer; for example, 1024. ■ LDAP Root Name: oracle.security.jps.ldap.root.name ■ Farm Name: oracle.security.jps.farm.name ■ Resource Type Enforcement Mode: oracle.security.jps.policystore.resourcetypeenforcementmode ■ Username: security.principal ■ Password: security.credential **FILE** ■ Policy Store File: location of the file used as the policy store

Table 8–4 documents the Advanced configuration properties for the Java Security Module.

Table 8–4 Java Security Module Advanced Properties

Property Name	jps-config.xml Property
Rolemember Cache Type	oracle.security.jps.policystore.rolemember.cache.type
Rolemember Cache Strategy	oracle.security.jps.policystore.rolemember.cache.strategy
Rolemember Cache Size	oracle.security.jps.policystore.rolemember.cache.size
Rolemember Cache Warmup Enable	oracle.security.jps.policystore.rolemember.cache.warmup.enable

Table 8–4 (Cont.) Java Security Module Advanced Properties

Property Name	jps-config.xml Property
Policy Lazy Load Enable	oracle.security.jps.policystore.policy.lazy.load.enable
Policy Cache Strategy	oracle.security.jps.policystore.policy.cache.strategy
Policy Cache Size	oracle.security.jps.policystore.policy.cache.size
Policy Cache Updatable	oracle.security.jps.policystore.cache.updateable
Refresh Enable	oracle.security.jps.policystore.refresh.enable
Refresh Purge Timeout	oracle.security.jps.policystore.refresh.purge.timeout
Refresh Purge Interval	oracle.security.jps.ldap.policystore.refresh.interval
Missing App Policy Query TTL	oracle.security.jps.pdp.missingAppPolicyQueryTTL
Decision Cache Enable	oracle.security.jps.pdp.AuthorizationDecisionCacheEnabled
Decision Cache Eviction Capacity	oracle.security.jps.pdp.AuthorizationDecisionCacheEvictionCapacity
Decision Cache Eviction Percentage	oracle.security.jps.pdp.AuthorizationDecisionCacheEvictionPercentage
Decision Cache TTL	oracle.security.jps.pdp.AuthorizationDecisionCacheTTL
Anonymous Role Enable	oracle.security.jps.pdp.anonymousrole.enable
Authenticated Role Enable	oracle.security.jps.pdp.authenticatedrole.enable

Table 8–5 documents the parameters for defining an attribute retriever as the Policy Information Point (PIP). After clicking New, the display contains the parameters for each attribute retriever type.

Table 8–5 Java Security Module PIP Parameters (Attribute Retrievers)

Name	Definition
Attribute Retriever	Type of attribute retriever is chosen from the drop down menu. Options include LDAP, DB and Custom.

LDAP	■ Name: the Attribute Retriever's name as defined in the `serviceInstance` tag
	`<serviceInstance name="dbname"`
	`provider="pip.service.provider">`
	■ Description: an optional description as defined in the `serviceInstance` tag
	`<serviceInstance name="description"`
	`value="dbdescription">`
	■ LDAP URL: ldap.url defines the location of the LDAP policy store. Valid in JEE and JSE applications and only applies to LDAP stores. Takes as a value a URI in the format `ldap://`*host:port*
	■ Failed Server Retry Interval: interval of time defined for the `failed.server.retry.interval` property when attempting to reach a failed server again
	■ Username: security.principal
	■ Password: security.credential

Table 8–5 (Cont.) Java Security Module PIP Parameters (Attribute Retrievers)

Name	Definition
DB	■ Name: the predefined Attribute Retriever's name ■ Description: an optional description ■ Database Configuration Through URL ■ JDBC URL: jdbc.url defines the location of the database policy store. Must be defined when using the Java Database Connectivity (JDBC) API to connect to a database. Takes as a value a list of comma-delimited URLs where the first is treated as primary and so on. For example, jdbc:oracle:thin:@scl58116.domainexample.com:1521:orcl ■ JDBC Driver: jdbc.driver defines the location of the driver when using JDBC API to connect to a database. Takes as a value `oracle.jdbc.driver.OracleDriver`, for example. ■ Username: security.principal ■ Password: security.credential ■ Database Configuration Through JNDI Name ■ Datasource JNDI Name: Data source JNDI name if you want the PIP instance working through data source rather than directly through JDBC. The data source scenario is supported on WebLogic Server and WebSphere Application Server only. Takes as a value the JNDI name of the pre-defined data source object. ■ Failed Server Retry Interval: After communication with a primary repository has failed, this attribute defines the interval of time during which the backup repository is used before switching back to the primary repository. Takes as a value the number of seconds; the default value is 15.
Custom	■ Name: the custom Attribute Retriever's name ■ Description: an optional description ■ Class Names: one or more class names

Table 8–6 documents the parameters that define the attribute which should be retrieved from the applicable Policy Information Point (PIP). It includes the parameters for both LDAP and database stores.

Table 8–6 Java Security Module PIP Parameters (Attributes)

Name	jps-config.xml Property
Attribute Retriever	Select the defined Attribute Retriever from the drop down menu.

Table 8–6 (Cont.) Java Security Module PIP Parameters (Attributes)

Name	jps-config.xml Property
Name	Name of the attribute as defined in the policy store. If using the predefined LDAP Attribute Retriever, the attribute name defined for Oracle Entitlements Server must be the same as the attribute name defined in the LDAP store. Currently, there is no name mapping functionality.

Query	This is an example of an LDAP query:
	`<property name="query" value="(cn=xUSERATTR)"/>`
	This is an example of a database query:
	`<property name="query" value="select description from bookstore where author='jimmy'"/>`

Search Base	The search base for an LDAP store; not displayed for a database
Time To Live	Time-to-live (in seconds) of any cached attribute values when cached is enabled.
Attribute Cached	Enables the caching of attribute values.

8.4.2 Configuring the RMI Security Module

The following links document the parameters when configuring the RMI Security Module post-instantiation.

- Table 8–1, " Java Security Module Controlled-Push Client Configuration" documents the controlled-push Client Configuration properties for the RMI Security Module.
- Table 8–2, " Java Security Module Controlled-Pull Client and Store Configuration" documents the controlled-pull Client and Store Configuration properties for the RMI Security Module.
- Table 8–3, " Java Security Module Non-controlled Policy Store Configuration" documents the non-controlled Policy Store Configuration properties for the RMI Security Module.
- Table 8–4, " Java Security Module Advanced Properties" documents the Advanced configuration properties for the RMI Security Module.
- Table 8–5, " Java Security Module PIP Parameters (Attribute Retrievers)" and Table 8–6, " Java Security Module PIP Parameters (Attributes)" document the the PIP Parameter properties for the RMI Security Module.

See Appendix A, "Installation and Configuration Parameters" for descriptions of the properties.

> **Tip:** When using a JBOSS Security Module in Proxy Mode as the PEP and the RMI Security Module as the PDP, `jbossx.jar` must be added to the CLASSPATH of the RMI Security Module.

8.4.3 Configuring the Web Services Security Module

The following links document the parameters when configuring the Web Services Security Module post-instantiation.

- Table 8–1, " Java Security Module Controlled-Push Client Configuration" documents the controlled-push Client Configuration properties for the Web Services Security Module.

Note: The SSL properties in Table 8–1 are not displayed or configurable for the Web Services Security Module.

- Table 8–2, " Java Security Module Controlled-Pull Client and Store Configuration" documents the controlled-pull Client and Store Configuration properties for the Web Services Security Module.
- Table 8–3, " Java Security Module Non-controlled Policy Store Configuration" documents the non-controlled Policy Store Configuration properties for the Web Services Security Module.
- Table 8–4, " Java Security Module Advanced Properties" documents the Advanced configuration properties for the Web Services Security Module.
- Table 8–5, " Java Security Module PIP Parameters (Attribute Retrievers)" and Table 8–6, " Java Security Module PIP Parameters (Attributes)" document the the PIP Parameter properties for the Web Services Security Module.

See Appendix A, "Installation and Configuration Parameters" for descriptions of the properties. See Section 8.4.4, "Configuring the WebLogic Server Security Module" if using the Web Services Security Module on a WebLogic Server.

Tip: When using a JBOSS Security Module in Proxy Mode as the PEP and the Web Services Security Module as the PDP, `jbossx.jar` must be added to the CLASSPATH of the Web Services Security Module.

8.4.4 Configuring the WebLogic Server Security Module

The following links document the parameters when configuring the WebLogic Server Security Module post-instantiation. These parameters are also valid when using the Web Services Security Module on a WebLogic Server.

Note: If installed on a WebLogic Server, there is no need to configure the Web Service entry point.

- Table 8–1, " Java Security Module Controlled-Push Client Configuration" documents the controlled-push Client Configuration properties for the WebLogic Server Security Module. The following additional WLS Configuration Properties are specific to the WebLogic Server when the security providers have been enabled (as documented in Section 9.4.1, "Integrating with WebLogic Server") and can be defined as *permit*, *abstain* or *deny*.
  - Undefined Application Effect
  - No Applicable Policy Effect
- Table 8–2, " Java Security Module Controlled-Pull Client and Store Configuration" documents the controlled-pull Client and Store Configuration properties for the WebLogic Server Security Module. The following additional WLS Configuration Properties are specific to the WebLogic Server when the security providers have been enabled (as documented in Section 9.4.1, "Integrating with WebLogic Server") and can be defined as *permit*, *abstain* or *deny*.
  - Undefined Application Effect
  - No Applicable Policy Effect

- Table 8–3, " Java Security Module Non-controlled Policy Store Configuration" documents the non-controlled Policy Store Configuration properties for the WebLogic Server Security Module. The following additional WLS Configuration Properties are specific to the WebLogic Server when the security providers have been enabled (as documented in Section 9.4.1, "Integrating with WebLogic Server") and can be defined as *permit*, *abstain* or *deny*.
  - Undefined Application Effect
  - No Applicable Policy Effect
- Table 8–4, " Java Security Module Advanced Properties" documents the Advanced configuration properties for the WebLogic Server Security Module.
- Table 8–5, " Java Security Module PIP Parameters (Attribute Retrievers)" and Table 8–6, " Java Security Module PIP Parameters (Attributes)" document the the PIP Parameter properties for the WebLogic Server Security Module.

See Appendix A, "Installation and Configuration Parameters" for descriptions of the properties.

8.4.5 Configuring the SharePoint Server (MOSS) Security Module

The SMConfig UI does not configure parameters for the MOSS Security Module itself. It configures the Web Services Security Module that serves as the remote PDP for the Sharepoint Security Module. See Section 8.4.3, "Configuring the Web Services Security Module" for information. The *Oracle Fusion Middleware Installation Guide for Oracle Identity and Access Management* contains additional information on configuring the MOSS Security Module.

8.4.6 Configuring the .NET Security Module

The SMConfig UI does not configure parameters for the .NET Security Module itself. It configures the Web Services Security Module that serves as the remote PDP for the .NET Security Module. See Section 8.4.3, "Configuring the Web Services Security Module" for information. The *Oracle Fusion Middleware Installation Guide for Oracle Identity and Access Management* contains additional information on configuring the MOSS Security Module.

8.4.7 Configuring the WebSphere, Tomcat and JBoss Security Modules

The following links document the parameters when configuring the WebSphere, Tomcat and JBoss Security Modules post-instantiation.

- Table 8–1, " Java Security Module Controlled-Push Client Configuration" documents the controlled-push Client Configuration properties.
- Table 8–2, " Java Security Module Controlled-Pull Client and Store Configuration" documents the controlled-pull Client and Store Configuration properties.
- Table 8–3, " Java Security Module Non-controlled Policy Store Configuration" documents the non-controlled Policy Store Configuration properties.
- Table 8–4, " Java Security Module Advanced Properties" documents the Advanced configuration properties.
- Table 8–5, " Java Security Module PIP Parameters (Attribute Retrievers)" and Table 8–6, " Java Security Module PIP Parameters (Attributes)" documents the PIP Parameter properties.

Tip: In order for the Security Modules to recognize JBOSS Principals, add *JBOSS_HOME*/common/lib/jbosssx.jar - which contains org.jboss.security.SimplePrincipal - into the CLASSPATH of the RMI/WS server.

See Appendix A, "Installation and Configuration Parameters" for descriptions of the properties.

8.4.8 Configuring the Oracle Service Bus Security Module

The Oracle Service Bus (OSB) Security Module works only when the Oracle Entitlements Server authorization providers are enabled using the WebLogic Server console (as defined in Section 9.4.1, "Integrating with WebLogic Server"). See Table 8.4.4, "Configuring the WebLogic Server Security Module" for details on configuring the OSB Security Module parameters post-instantiation.

Note: When an Oracle Entitlements Server Security Module is installed into a JRF domain, only non-controlled mode is supported.

The following procedure is used to configure the OSB Security Module.

1. After installing and starting the OSB server and Oracle Entitlements Server, install the OESClient.

   See *Oracle Fusion Middleware Installation Guide for Oracle Identity and Access Management*.

2. Apply Opatch to the installed OESClient and the oracle_common directory.

   The oracle_common directory is created during OSB installation.

3. Modify the smconfig.prp file.

   smconfig.prp is located in the *OES_CLIENT_HOME*/oessm/SMConfigTool directory. See *Oracle Fusion Middleware Installation Guide for Oracle Identity and Access Management*.

4. Create an OSB Security Module using the SMConfig Tool.

   Ensure the Oracle Entitlements Server is runing. See *Oracle Fusion Middleware Installation Guide for Oracle Identity and Access Management*.

   a. Change to the bin directory.

      cd OES_CLIENT_HOME/oessm/bin

   b. Run the SMConfig Tool.

      ```
      ./config.sh -smType wls -smConfigId osbSM -serverLocation
      ../wlserver_10.3/
      ```

   c. Input the name and password of the user with permission to access the Oracle Entitlements Server policy store in the command window.

   d. Select *Oracle Entitlements Server Security Module On Service Bus -11.1.2.0[oesclient]* when prompted by the Fusion Middleware Configuration Wizard to Select Extension Source and click Next.

5. Modify the policy store configuration in the jps-config.xml file in the OSB Domain.

6. Enable the Oracle Entitlements Server authorization and role mapping providers using the WebLogic Server console.

See Section 9.4.1, "Integrating with WebLogic Server."

7. Restart the OSB server.

8.5 Configuring the PDP Proxy Client for RMI or Web Services

Oracle Entitlements Server supports a Proxy Mode that allow clients to invoke authorization services remotely. When the Oracle Entitlements Server PEP API are used by an application to make authorization calls, a PDP Proxy Client can be set up on the application's host to communicate with the remote Security Module (on another machine) for access decisions. The PDP Proxy Client provides local security services, including caching, logging, and failover support. A Remote Method Invocation (RMI) or Web Services call can be used as the method of communication between the PDP Proxy Client and the remote Security Module. The RMI and Web Services Security Modules are the only ones to invoke security services remotely.

Note: In XACML terminology, the proxy and remote Security Module are analogous to the PDP Proxy and PDP, respectively.

The Oracle Entitlements Server PEP API can be called from an application regardless of whether that application is calling an embedded PDP or a remote PDP. Proxy Mode can be configured to provide security services locally (including authorization caching, logging and failover) and can communicate with the PDP using RMI or SOAP.

There are configurations involved with both the local PDP Proxy Client and the remote Security Module. On the proxy side, client configurations are consolidated in the PDP Service instance inside the `jps-config.xml` configuration file. On the server side, configuration parameters for both the RMI and Web Services Security Modules are also consolidated in the PDP Service instance inside the `jps-config.xml` configuration file. See Appendix A, "Installation and Configuration Parameters" for details on the configuration parameters.

9

Securing Environment Specific Resources

This chapter contains information on how to secure resources using specific Security Modules. The following sections contain detailed information.

- Choosing a Security Module Type
- Securing Microsoft Office SharePoint Server Resources
- Securing Oracle Service Bus Resources
- Securing WebLogic Server Resources

9.1 Choosing a Security Module Type

An Oracle Entitlements Server administrator chooses a Security Module type based on the type of resource being protected. The Security Module type is defined when the Security Module is instantiated using the OES Client's SMConfig Tool (as documented in Chapter 8, "Managing Security Module Configurations").

> **Note:** See the *Oracle Fusion Middleware Installation Guide for Oracle Identity and Access Management* for OES Client installation instructions.

Instantiation of a Security Module using the SMConfig Tool defines parameters such as the Security Module type, configuration ID and a path to the appropriate container. Table 9–2 lists the Security Module types that can be instantiated. (The SMType attribute value is defined in parentheses in the Security Module column.)

The Security Module types documented in Table 9–1 do not protect resources specific to the environment. They are a Policy Decision Point (PDP) that receives Java calls for authorization. The Security Modules simply provide the authorization API for the application to call.

Table 9–1 General Protection Security Module Types

Security Module	Purpose	Proxy Mode	Container Support
Java (java)	Policy Decision Point (PDP) that receives Java calls for authorization	Supported	Used for Java, Standard Edition (JSE) applications and complex integrations
Websphere (was)	Java Security Module that receives authorization requests directly from the application	Supported	WebSphere Application Server

Table 9–1 (Cont.) General Protection Security Module Types

Security Module	Purpose	Proxy Mode	Container Support
Web Services (ws)	Multi-Protocol Security Module that accepts Web Services calls; configure other Security Modules (Java, WLS) in proxy mode to communicate with Web Services Security Module using the XACML Gateway or SOAP.	Not supported	Supported as a JSE standalone process or as a web service running on a WebLogic Server container; can be used by other Security Modules in proxy mode; can be used as long as a Web Services request can be built or when an application is built in a non-standard programming language (Python, for example)
RMI (rmi)	Multi-Protocol Security Module is a Java Security Module that is enhanced to accept Remote Method Invocation (RMI) calls	Not supported	Use only for JSE applications if Web Services is too heavy
.NET (dotnet)	Allows applications written in C# to send authorization requests to the Web Services Security Module by calling the PEP API in C#	Supported in Proxy Mode only; .NET Security Module serves as a proxy that communicates with the Web Services Security Module	.NET containers
JBoss (jboss)	Java Security Module that receives authorization requests directly from the application	Supported	JBoss Application Server
Tomcat (tomcat)	Java Security Module that receives authorization requests directly from the application; does not support Oracle Platform Security Services authentication/login module or `checkPermission`	Supported	Apache Tomcat Application Server

The Security Modules documented in Table 9–2 allow the protection of resources that are specific to the environment in which they are deployed in addition to processing direct authorization calls from the application.

Table 9–2 Environment Specific Security Module Types

Security Module	Purpose	Proxy Mode	Container Support
Sharepoint (moss)	Protects Microsoft Sharepoint Server resources by intercepting Sharepoint requests for content; see Section 9.2, "Securing Microsoft Office SharePoint Server Resources"	Supported only in Proxy Mode; Sharepoint Security Module itself serves as a proxy (written in C#) that communicates with the Web Services Security Module	.NET containers

Table 9–2 (Cont.) Environment Specific Security Module Types

Security Module	Purpose	Proxy Mode	Container Support
WebLogic Server (wls)	Security Module that behaves exactly as Java Security Module unless the Oracle Entitlements Server security providers are enabled (as documented in Section 9.4.1, "Integrating with WebLogic Server") in which case the Security Module can also process WebLogic Server calls; see Section 9.4, "Securing WebLogic Server Resources"	Supported - unless the Oracle Entitlements Server security providers are enabled to intercept WebLogic Server requests.	WebLogic Server
Oracle Service Bus (osb)	WebLogic Security Module with Oracle Entitlements Server security providers enabled that intercepts authorization requests from the Oracle Service Bus; see Section 9.3, "Securing Oracle Service Bus Resources"	Supported	

See Chapter 8, "Managing Security Module Configurations" for details on these Security Modules and how to configure them. Details on how they work can be found in Section 1.3, "Overview of the Oracle Entitlements Server Architecture" and Chapter 7, "Deploying the Policy Decision Point."

9.2 Securing Microsoft Office SharePoint Server Resources

Oracle Entitlements Server enables enterprises to manage MOSS portal environments. Integration with MOSS is provided through the MOSS Security Module plug-in that intercepts authorization calls within the SharePoint Server and sends them to its integrated Web Services Security Module (the PDP). The PDP then returns the authorization decision back to the MOSS Security Module plug-in and the decision is enforced. The following sections have more information.

- Section 9.2.1, "Protecting SharePoint Resources"
- Section 9.2.2, "Instantiating the MOSS and Web Services Security Modules"
- Section 9.2.4, "Configuring for SharePoint Security"

9.2.1 Protecting SharePoint Resources

SharePoint components that can be secured include web sites, web pages, web parts, list items, navigation bar items and the like. Based on the component, the resource is protected differently. SharePoint resources are categorized according to the following list.

- *Items* are the smallest SharePoint components; for example, a Document, a Task, a Contact, a Page or an Announcement.
- *Lists* are a collection of a single type of SharePoint component. Document Lists, Contacts Lists, Task Lists and the like can be created.
- *Folders* exist in Lists and serve as a container for multiple Items and sub Folders.
- *Sites* are a collection of Lists. For example, the default SharePoint *Document Center* Site is made up of three Lists: Announcements, Documents and Tasks.

- *Navigation Bar Items* on SharePoint site pages can be used to manipulate MOSS components.

> **Note:** There is only one Resource Type for all MOSS resources. In this section, we use the name `MossResourceType`.

The following sections contain more information.

- Section 9.2.1.1, "Protecting Web Sites and Web Pages"
- Section 9.2.1.2, "Protecting Web Parts"
- Section 9.2.1.3, "Protecting Lists"
- Section 9.2.1.4, "Protecting Sensitive Content Within Web Pages"

9.2.1.1 Protecting Web Sites and Web Pages

MOSS web sites are composed of one or more web pages. An organization generally organizes one web site in MOSS to denote one department in the company. MOSS comes with a main Web Site within which there are default sub sites. Sub sites appear on the top or side navigation bar or as links on other web pages. All these web sites have their own unique URLs.

The URL of a MOSS Web site or Web page defines the Resource instance created in Oracle Entitlements Server. In the case of a URL defined as `http://`*Sharepoint_Server_Name*`/TestSite`, the corresponding Resource is created by defining a `/TestSite` Resource as an instance of the `MossResourceType` Resource Type under the `MossApp` Application. Policies are then created using the Oracle Entitlements Server objects.

A Custom HTTP Module is implemented by Oracle Entitlements Server to secure the MOSS web sites. When a user tries to access a protected component, the request is intercepted by the Custom HTTP Module and forwarded to Oracle Entitlements Server for policy evaluation. The decision is returned to the Custom HTTP Module and if the user is denied access, a Custom Error Page with a message indicating a lack of permissions to view this location is displayed.

> **Note:** Custom HTTP Modules are enabled by defining the `HttpModules` elements in the `web.config` MOSS Site configuration file. See Section 9.2.4, "Configuring for SharePoint Security."

9.2.1.2 Protecting Web Parts

A MOSS Web Part is similar to a portlet in that it is used to publish content within web pages. A Web page may contain one or more Web Parts. Web Parts are represented in Oracle Entitlements Server by defining their unique MOSS Display Name as the Resource instance name. The Resource instance is created as a child of the parent web page's Resource.

When a user tries to access these protected components the request is intercepted by a MOSS Delegate Control created for Oracle Entitlements Server. In short, a Delegate Control allows you to put any custom .NET code into a SharePoint page without modifying the page itself. This custom code is used to retrieve the decision from Oracle Entitlements Server and remove unauthorized Web Parts from the page. There are no error messages displayed in this case.

Note: Delegate OES Authorization Control is explicitly added to the Web Part pages or implicitly defined in the web site's master page. See Section 9.2.4, "Configuring for SharePoint Security."

9.2.1.3 Protecting Lists

A MOSS List is a collection of items within a Web Part on a Web page. When creating a MOSS Web site, a set of lists is also created depending upon the template used. Each list item is identified by a URL and represented as an Oracle Entitlements Server Resource. These lists are incorporated into Oracle Entitlements Server based on whether they are *document* lists or *non-document* lists.

- Document Lists can be displayed by going to `http://`*Sharepoint_Server_Name*`/TestSite/SharedDocuments/Forms/AllItems.aspx`. Create a top-level Resource named `/TestSite/SharedDocuments/Forms/AllItems.aspx`. Next create individual Resource objects for each item on the list as a sub resource to the top-level Resource. For example, a sub Resource named `/TestSite/SharedDocuments/Scott.sql` can be created for an item on the list named `Scott.sql`.
- Non-document Lists can be displayed by going to `http://`*Sharepoint_Server_Name*`/TestSite/Lists/Announcements/AllItems.aspx`. Create a Resource named `/TestSite/Lists/Announcements/AllItems.aspx`. Next create a `/TestSite/Lists/Announcements/EditForm.aspx` Resource and a `/TestSite/Lists/Announcements/DispForm.aspx` Resource at the same level. Now click on any item in the list; the URL appears as `http://`*Sharepoint_Server_Name*`/web1/Lists/Announcements/DispForm.aspx?ID=2&Source=http%3A%2F%2Fsharepoint01%2FTestSite%2FLists%2FAnnouncements%2FAllItems%2Easpx`. Note the ID defined as a URL parameter in the URL. This ID will be used as the name of the non-document item and is created as a sub Resource of both `EditForm.aspx` and `DispForm.aspx`. This must be done for all items within a Non-document List. Alternately, hover the mouse over the link of the item and note the ID from the URL displayed in the status bar of the browser.

Note: For list items only, you don't need to write policy on `EditForm.aspx`. You may grant view or ANY on the same `DispForm.aspx`. If view, ReadOnly access is granted; if ANY, full access (edit, delete, and the like) is granted.

9.2.1.4 Protecting Sensitive Content Within Web Pages

The SharePoint server allows administrators to publish custom pages which have sensitive information that need access control. The developer may enclose the sensitive information within ASP tags corresponding to the Oracle Entitlements Server Tag Library. The tag library communicates with Oracle Entitlements Server to retrieve the access decision and, as a result, the content is shown only to authorized users.

Note: The ASP tag library is a server side web control used by a MOSS page developer who registers the namespace, tag-prefix and assembly in the MOSS page and uses the tag to enclose the sensitive content. The library is invoked when an end user tries to access a custom content page on which the tags have been used to provide access control.

9.2.2 Instantiating the MOSS and Web Services Security Modules

The MOSS Security Module works with the Web Services Security Module to provide fine grained authorization for MOSS resources. Before instantiating the Security Modules, ensure that the pre-requisite MOSS environment is already setup. This includes installation of the MOSS and creation of the web application to be protected.

Note: Environment details of the web application (port number, URL and the like) will be needed in this procedure.

The MOSS Security Module and the Web Services Security Module can be deployed on the same or different servers. Instantiation of both Security Modules is achieved using the SMConfig Tool found in the `$ORACLE_CLIENT_HOME/oessm/SMConfigTool/bin` directory.

- To instantiate both the MOSS Security Module and the Web Services Security Module at once, run `config.sh` with the parameter `-smType mossws`.
- To instantiate the Security Modules separately, run `config.sh` twice, first with the parameter `-smType moss` and then with -smType ws.

Additionally, run the SMConfigTool based on your deployment choices. For example, to instantiate the Security Modules when they are deployed on the same Windows machine as the MOSS, use the following command:

```
config.sh -smType mossws -prpFileName file_name -mossprpFileName file_name
   -smConfigId -WSListeningPort -pdServer -pdPort
```

where `prpFileName` refers to the `smconfig.prp` used to create the Web Services Security Module and `mossprpFileName` refers to the properties file used to configure the MOSS server.

Note: The `mossprpFileName` template is located at `$ORACLE_CLIENT_HOME/oessm/mosssm /adm/configtool/moss_config.properties`. `moss_config.properties` has mandatory properties that must be defined according to your environment and optional properties that, if not defined, use default values.

To instantiate the Security Modules when they are on separate Windows machines, first instantiate the Web Services Security Module (no special instructions). Then use the following command to configure the MOSS Security Module.

```
config.sh -smType moss -prpFileName file_name -mossprpFileName file_name
```

See the *Oracle Fusion Middleware Installation Guide for Oracle Identity and Access Management* for detailed OES Client installation instructions.

9.2.3 Integrating and Disintegrating the MOSS Security Module

The value of the `moss.enableOES` property in the `mossprpFileName` template can be used to integrate Oracle Entitlements Server with, or disintegrate Oracle Entitlements Server from, the MOSS application.

- If `moss.enableOES=true`, executing `config.sh` with the `-smType moss` parameter will integrate the MOSS application with Oracle Entitlements Server.

- If `moss.enableOES=false`, executing `config.sh` with the `-smType moss` parameter will configure the MOSS application to use its default authorization process.

9.2.4 Configuring for SharePoint Security

The following procedure documents the steps to manually configure authorization for a MOSS application rather than the automatic configuration initiated by SMConfigTool. These steps are the same as those automatically executed by the `config.sh` script in Section 9.2.2, "Instantiating the MOSS and Web Services Security Modules" and can be used instead, by those who prefer not to use the SMConfigTool.

This procedure assumes that the Security Modules have been instantiated. After instantiation, note the generated configuration ID and service registry URL. In the following procedure, the configID is **MOSS** and the service registry URL is `http://`*`hostname:port`*`/ServiceRegistry`.

1. Use the Oracle Entitlements Server Administration Console to complete the following steps.
   - Create the Web Services Security Module definition.

     See Chapter 10, "Managing System Configurations."
   - Create an Application policy object to represent the MOSS application being protected.

     The name of the Application must be consistent with the value of the `moss.app.name` property defined in the `moss_config.properties` file.

     See Chapter 4, "Managing Policies and Policy Objects."
   - Bind this Application to the Web Services Security Module profile.

     See Chapter 10, "Managing System Configurations."
2. Drag and drop `OES.SharePoint.dll` and `log4net.dll` from the *$OES_CLIENT_HOME*`/oes_sm_instances/`*MOSS_SM_Name*`/lib` directory to the `C:/WINDOWS/assembly` directory.

   This registers the assemblies in the Windows Global Assembly Cache and makes them available to all .NET applications on the host machine.
3. If using MOSS 2007 (IIS 6), declare the Oracle Entitlements Server Delegate Control by adding the following code to the HTML HEAD section of the `default.master` file.

   ```
   <SharePoint:DelegateControl runat="server" ControlId="PageHeader"/>
   ```

   The default.master file is located in the *`C:\Program Files\Common Files\Microsoft Shared\web server extensions\12\`*`TEMPLATE\GLOBAL\` directory. This step is not required when using MOSS 2010 (IIS 7).

   Caution: When `default.master` is opened with Wordpad, question mark (?) characters sporadically replace existing characters. Ensure that this is corrected before saving your modified file. Alternately, open the file with Notepad.
4. Add the Custom Error Page to display a message when the user is not authorized to access the MOSS component.

CustError.aspx and custError2010.aspx are the custom Oracle Entitlements Server error pages for MOSS. They are located in the *$OES_CLIENT_HOME*/oes_sm_instances/*MOSS_SM_Name*/adm/pages directory.

- If using MOSS 2007 (IIS 6), copy custError.aspx to C:\ Program Files\Common Files\Microsoft Shared\web server extensions\12\template\layouts.
- If using MOSS 2010 (IIS 7), copy custError2010.aspx to C:\Program Files\Common Files\Microsoft Shared\web server extensions\14\template\layouts and change the name to custError.aspx.

5. Edit the Sharepoint Server web.config configuration file to enable Oracle Entitlements Server-MOSS integration.

   web.config is located in the virtual directory of the SharePoint application; for example, C:\Inetpub\wwwroot\wss\VirtualDirectories*port-number* where *port-number* is the application's port.

   Caution: When web.config is opened with Wordpad, question mark (?) characters sporadically replace existing characters. Ensure that this is corrected before saving your modified file. Alternately, open the file with Notepad.

   a. Add the properties documented in Table 9–3 to the appSettings section. Example 9–1 illustrates the appSettings section. Values are taken from the file defined as mossprpFileName.

Example 9–1 appSettings Section of Sharepoint web.config File

```
<add key="SsmUrl" value="${moss.SmUrl}/ServiceRegistry"/>
<add key="SsmId"
     value="${oracle.security.jps.runtime.pd.client.sm_name}"/>
<add key="ApplicationID" value="${application.id}"/>
<add key="PolicyDomain" value="${policy.domain}"/>
<add key="ResourceType" value="${moss.resourcetype}"/>
<add key="log4NetXmlfile" value="${moss.log4NetXmlfile}”/>
<add key="sharepointSite" value="${moss.sharepointSite}"/>
<add key="EnableOES" value="${moss.EnableOES}"/>
<add key="IgnoredExtensions" value="${moss.IgnoredExtensions}"/>
<add key="IgnoredURLExpression" value=”${moss.IgnoredURLExpression}"/>
```

Table 9–3 appSettings Properties for the MOSS Application

Property	Value
SsmUrl	Registry URL of the Web Service SM; for example, http://oesw2k8:9400/ServiceRegistry
SsmId	The name of this Security Module; for example, MOSS
IdentityAsserterName	The name of the identity asserter configured in Oracle Entitlements Server. At this time, only OESIdentityAssertion is supported.
ApplicationID	The name of the configured Oracle Entitlements Server Application that represents the protected MOSS resource.

Table 9–3 (Cont.) appSettings Properties for the MOSS Application

Property	Value
ResourceType	The Resource Type of all MOSS resources; for example, `MossResourceType`. The Resource Type of all MOSS components to be protected must be consistent with the value of the `moss.resource.type` property defined in the `moss_config.properties` file.
log4NetXmlfile	Fully qualified path to the `log4Net.xml` configuration file. The log file defined in the `log4Net.xml` file must be located in an existing directory for which Read/Write access has been given to Everyone to allow log messages to be written.
sharepointSite	Top level SharePoint site; for example, `http://alesw2k8:9581/`
Enable OES	Flag to enable the OES integration; takes true or false as a value
IgnoredExtensions	A comma-separated list of file extension patterns to be ignored by OES Access Control; for example, `png,js,css,axd` Access will always be granted to these resources when requested.
IgnoredURLExpression	A comma-separated list of file name patterns to be ignored by OES Access Control; for example, `/_layouts/Authenticate.aspx,/_login/default.aspx,/_forms/default.aspx` Access will always be granted to these resources when requested.

b. Add the `SafeControl Assembly` entries documented in Table 9–2 to the `SafeControls` section.

Example 9–2 SafeControl Assembly Entries

```
<SafeControls>
      ...
<SafeControl Assembly="OES.Sharepoint, Version=1.0.0.0, Culture=neutral,
  PublicKeyToken=68b08a2fa869dfdc" Namespace="OES.Sharepoint.Controls"
  TypeName="*" Safe="True" />
<SafeControl Assembly="OES.Sharepoint, Version=1.0.0.0, Culture=neutral,
  PublicKeyToken=68b08a2fa869dfdc" Namespace="OES.Sharepoint.Modules"
  TypeName="*" Safe="True" />
</SafeControls>
```

c. Define custom `httpModules` based on the server used.

If using MOSS 2007 (IIS 6), add the following to the `httpModules` section.

```
<add name="CustHTTPModule" type="OES.Sharepoint.Modules.CustHTTPModule,
  OES.Sharepoint, Version=1.0.0.0, Culture=neutral,
  PublicKeyToken=68b08a2fa869dfdc " />
```

If using MOSS 2010 (IIS 7), add Example 9–3 to the `assemblies` section and Example 9–4 to the `modules` section (after the last `<remove>` and before the first `<add>`).

Example 9–3 add assembly Entry

```
<assemblies>
…
```

```
<add assembly="OES.Sharepoint, Version=1.0.0.0, Culture=neutral,
  PublicKeyToken=68b08a2fa869dfdc" />
…
</assemblies>
```

Example 9–4 add name Entry

```
<modules runAllManagedModulesForAllRequests="true">
…
<add name="CustHTTPModule" preCondition="integratedMode"
  type="OES.Sharepoint.Modules.CustHTTPModule, OES.Sharepoint,
  Version=1.0.0.0, Culture=neutral, PublicKeyToken=68b08a2fa869dfdc" />
…
</modules>
```

d. Update the `PageParserPaths` (in the `SafeMode` section) with the virtual path to which custom content is required to be published. The custom content may be authorized via the tag library provided with the solution. Example 9–5 is an example.

Example 9–5 PageParserPaths Entry

```
<PageParserPaths>
  <PageParserPath VirtualPath="/Pages/*" CompilationMode="Always"
     AllowServerSideScript="true" IncludeSubFolders="true"/>
</PageParserPaths>
```

e. Replace the MOSS PortalSiteMapProvider details (illustrated in Example 9–6) with the Oracle Entitlements Server PortalSiteMapProvider details (illustrated in Example 9–7).

The custom Oracle Entitlements Server PortalSiteMapProvider secures the Navigation Bar items.

Example 9–6 MOSS PortalSiteMapProvider

```
<SiteMap>
 <Providers>
     …
<add name="GlobalNavigation" description="Provider for MOSS Global Navigation"
  type="Microsoft.SharePoint.Publishing.Navigation.PortalSiteMapProvider,
  Microsoft.SharePoint.Publishing, Version=14.0.0.0, Culture=neutral,
  PublicKeyToken=71e9bce111e9429c" NavigationType="Combined" Version="14" />
<add name="CurrentNavigation" description="Provider for MOSS Current Navigation"
  type="Microsoft.SharePoint.Publishing.Navigation.PortalSiteMapProvider,
  Microsoft.SharePoint.Publishing, Version=14.0.0.0, Culture=neutral,
  PublicKeyToken=71e9bce111e9429c" NavigationType="Current" Version="14" />
     …
 </Providers>
</SiteMap>
```

Example 9–7 Oracle Entitlements Server PortalSiteMapProvider

```
<SiteMap>
 <Providers>
     …
<add NavigationType="Combined" Version="1" description="Provider for MOSS
  Global Navigation" name="GlobalNavigation"
  type="OES.Sharepoint.Controls.OESPortalSiteMapProvider, OES.Sharepoint,
```

```
  Version=1.0.0.0, Culture=neutral, PublicKeyToken=68b08a2fa869dfdc"/>
<add NavigationType="Current" Version="1" description="Provider for MOSS
  Current Navigation" name="CurrentNavigation"
  type="OES.Sharepoint.Controls.OESPortalSiteMapProvider, OES.Sharepoint,
  Version=1.0.0.0, Culture=neutral, PublicKeyToken=68b08a2fa869dfdc"/>
    …
 </Providers>
</SiteMap>
```

f. Restart IIS server for above changes to reflect in IIS server.

6. Copy the `OESAuthorizationFeature` directory to the MOSS `FEATURES` directory.

- If using MOSS 2007 (IIS 6), copy the *`$OES_CLIENT_HOME`*`/oes_sm_instances/`*`MOSS_SM_Name`*`/lib/OESAuthorizationFeature` directory to the `C:\Program Files\Common Files\Microsoft Shared\web server extensions\12\TEMPLATE\FEATURES` directory.
- If using MOSS 2010 (IIS 7), copy the *`$OES_CLIENT_HOME`*`/oes_sm_instances/`*`MOSS_SM_Name`*`/lib/OESAuthorizationFeature2010` directory to the `C:\Program Files\Common Files\Microsoft Shared\web server extensions\14\TEMPLATE\FEATURES` directory and change the directory name to `OESAuthorizationFeature`.

7. Install and activate OESAuthorizationFeature for the specified site using one of the following commands.

The OES Authorization Feature can be activated separately for each web and sub-web site by going to Site Settings ->Modify All Site Settings-> Site Features. If it is activated against a sub-web, all web parts in the web pages inside the sub-web may be access controlled.

- If using MOSS 2007 (IIS 6), open a command prompt and execute the following commands.

```
"C:\Program Files\Common Files\Microsoft Shared\web server
extensions\12\BIN\STSADM.EXE" -o installfeature -name
OESAuthorizationFeature

"C:\Program Files\Common Files\Microsoft Shared\web server
extensions\12\BIN\STSADM.EXE" -o activatefeature -name
OESAuthorizationFeature -url http://alesw2k3:9581
```

- If using MOSS 2010 (IIS 7), open a command prompt and execute the following commands:

```
"C:\Program Files\Common Files\Microsoft Shared\web server
extensions\14\BIN\STSADM.EXE" -o installfeature -name
OESAuthorizationFeature

"C:\Program Files\Common Files\Microsoft Shared\web server
extensions\14\BIN\STSADM.EXE" -o activatefeature -name
OESAuthorizationFeature -url http://aleswin2k8:9581
```

8. Restart the IIS Server.

9. Obtain a list of all the SharePoint server protected resources using `MOSSResourceDiscovery.exe` located in the *`%OES_CLIENT_HOME%`*`\oessm\mosssm\lib` directory.

> **Note:** MOSS resources are mapped hierarchically to resources in Oracle Entitlements Server. Thus, all discovered resources need not to be defined in the policy store. For example, rather than copying 10,000 individual document names, copy the name of the folder in which these documents are located and write policies using a Resource Name Expression; for example, `/lib/*`.

The MOSS Security Module contains this executable to generate a plain text file named `object1` and an XML file named `discovered-jazn-data.xml`. Both files define the MOSS resources. The `MOSSResourceDiscovery.exe` executable prompts for the following information.

- The path to a directory in which the files will be created; for example, `c:\inetpub\wwwroot\wss\VirtualDirectories\9581\policy`. This directory must be created beforehand.
- The path to the directory in which the Admin Url file is located; for example, *$OES_CLIENT_HOME*`/oessm/mosssm/adm/Discovery/AdmUrls.txt`.
- The Sharepoint Server site URL; for example, `http://amw2k8:9581`. Do not append the URL with a forward slash (/).
- The name of the Oracle Entitlements Server Application object that represents the MOSS application; for example, `MossApp`.
- The name of the Oracle Entitlements Server Resource Type; this value should always be `MossResourceType`.

The XML file can be used by the policy migration tool. See Section 13.5, "Migrating Policies" for details. The text file is used to import the resources into the Oracle Entitlements Server policy store in the next step.

10. Import the MOSS resources into the Oracle Entitlements Server policy store by using the text file as input to `manage-policy.cmd|sh`, the policy management tool.

    `manage-policy.cmd|sh` is located in the *`%OES_CLIENT_HOME%`*`\oessm\bin` directory. The import appends the MOSS resources to the policy store; any existing MOSS Applications (and related policies) will not be deleted. The input values (Application name, Resource Type and generated resource file) should be consistent with the input used by MOSSResourceDiscovery.exe in the previous step.

    Before running `manage-policy.cmd|sh`, modify the script as follows:

    - Change the `OES_CLIENT_HOME` and `OES_INSTANCE_NAME` variables to reflect the user's environment.
    - Configure the `jps-config.xml` Policy Store attributes as defined in Appendix A.4, "Policy Store Service Configuration."

    This tool is only run once. New resources are manually created using the Administration Console.

11. Distribute the policies using the Administration Console.

9.3 Securing Oracle Service Bus Resources

Oracle Service Bus (OSB) is designed to centrally manage and control many distributed service endpoints. Oracle Entitlements Server enables an enterprise to control access to OSB runtime resources, allowing them to become accessible only after

authorization. In general, OSB runtime resources are those resources passed to the `isAccessAllowed()` authorization API.

> **Note:** Oracle Entitlements Server does not secure resources used during OSB configuration such as the OSB console.

The following sections contain detailed information on the OSB resource object and how to map its values to Oracle Entitlements Server policy objects.

- Section 9.3.1, "Examining the OSB Resource Object"
- Section 9.3.2, "Mapping Secure OSB Resources to Oracle Entitlements Server"
- Section 9.3.3, "Mapping Non-secure OSB Resources to Oracle Entitlements Server"
- Section 9.3.4, "Enabling the WebLogic Server Providers"

9.3.1 Examining the OSB Resource Object

OSB runtime resources are represented as objects. The object representing the resource contains a string array of `KEYS` that define values representing the object's context; for example, the OSB project or task. In order to secure OSB resources, the creation of Oracle Entitlements Server security objects used to define an Authorization Policy must mirror the values that will be passed in the resource object's KEYS. The following list are the `KEYS` that will be defined in an OSB resource object. The type of this OSB resource object is always `<alsb-proxy-service>`.

- **proxy** defines the name of the OSB proxy service associated with the protected resource. The value uniquely identifies one OSB proxy service.
- **path** defines the full path to the OSB proxy service; for example, *`Project-name/Folder-name`* where:
  - *Project-name* is the name of the OSB project with which the proxy service is associated.
  - *Folder-name* is an optionally defined directory structure for the proxy service. Multiple directories may be defined using the `/` string separator as in *`/folder_name/sub_folder_name`*.

  The value uniquely identifies the same OSB proxy service as the one referenced for **proxy**.
- **action** defines whether entry to the OSB proxy service will be secure or not and takes one of the following values:
  - *invoke* represents access control on entry to an OSB proxy service.
  - *wss-invoke* represents secure access control on entry to an operation of an OSB proxy service. With this action, OSB Web Service Security is configured.
- **operation** defines the name of the Web service operation being invoked. If the **action** is *invoke*, this value is null.

> **Note:** When the OSB proxy service uses Web Service Security, OSB performs security checks at the transport layer and the message layer. At the transport layer, OSB checks if the user is allowed to access the proxy service; at the message layer, it checks if the user is allowed to do the specified proxy service operation. Thus, if no user information is passed into the transport layer, an additional policy will be needed to grant access privileges to the Anonymous role.

Mapping OSB resources to Oracle Entitlements Server policy objects is dependent on the chosen secure or non-secure **action**. See Section 9.3.2, "Mapping Secure OSB Resources to Oracle Entitlements Server" and Section 9.3.3, "Mapping Non-secure OSB Resources to Oracle Entitlements Server" for details.

9.3.2 Mapping Secure OSB Resources to Oracle Entitlements Server

When the OSB resource object defines a *wss-invoke* action, the applicable OSB proxy service uses OSB Web Service Security. Let's assume an OSB proxy service named `SampleProxyService` is associated with the OSB project named `SampleProject` and is configured to use OSB Web Service Security. This service resource is in the `Mortgage/ProxyService` folder. Thus, the `KEYS` values are as follows:

- path: SampleProject/Mortgage/ProxyService
- proxy: SampleProxyService
- action: wss-invoke
- operation: `sayHello` (Suppose the Web Service action is sayHello)

Based on the KEYS values, the Oracle Entitlements Server object values are:

- Application - It is mandatory to name the Application used for securing OSB resources as `alsbProxyServices`.
- Resource Type - This value should always be the OSB object type `alsb-proxy-service`. Also select *yes* as the value of the Supports Resource Hierarchy parameter.

  > **Note:** It is not necessary to add `wss-invoke` as an action for the `alsb-proxy-service` Resource Type; just select the operation for the policy.

- Resource - SampleProject/Mortgage/ProxyService/SampleProxyService/sayHello (takes a value equal to the values of the OSB resource object's *path*/*proxy* `KEYS` values)
- Action - `access` (*access*, the default Oracle Entitlements Server privilege, is always used)

9.3.3 Mapping Non-secure OSB Resources to Oracle Entitlements Server

When the OSB resource object defines an *invoke* action, the applicable OSB proxy service does not use OSB Web Service Security. Let's assume an OSB proxy service resource named `SampleProxyService` that is associated with the OSB project named `SampleProject`. This service resource is in the `Mortgage/ProxyService` folder. Thus, the `KEYS` values are as follows:

- path: SampleProject/Mortgage/ProxyService
- proxy: SampleProxyService
- action: invoke
- operation: null

Based on the KEY values, if not configured with OSB Web Service Security, the Oracle Entitlements Server object values are:

- Application - alsbProxyServices (the OSB resource object does not have a defined value so this default value is used)
- Resource Type - This value should always be the OSB object type `alsb-proxy-service`. Also select *yes* as the value of the Supports Resource Hierarchy parameter.
- Resource - SampleProject/Mortgage/ProxyService/SampleProxyService (takes a value equal to the values of the OSB resource object's *path*/*proxy* `KEYS` values)
- Action - access (if *operation* has a value, this value is used; if not, *access*, the default Oracle Entitlements Server privilege, is used)

9.3.4 Enabling the WebLogic Server Providers

The Oracle Entitlements Server Proxy Provider must be enabled to secure and protect OSB runtime resources as well. Section 9.4.1, "Integrating with WebLogic Server" contains the procedure for accomplishing this.

9.4 Securing WebLogic Server Resources

Besides providing the authorization API to accept authorization requests, the WebLogic Server Security Module allows protection of WebLogic Server-specific resources after configuring the specific Oracle Entitlements Server Authorization and Role Mapping providers. The following high-level procedure documents the tasks to secure WebLogic Server resources.

1. Enable the Authorization and Role Mapping providers.

   See Section 9.4.1, "Integrating with WebLogic Server."

2. Discover the resources to be protected with Discovery Mode.

   See Section 9.4.2, "Discovering WebLogic Server Resources."

3. Define the WebLogic Server-specific resources as Oracle Entitlements Server objects.

   See Section 9.4.3, "Converting WebLogic Server Resources" and Section 9.4.4, "Mapping WebLogic Server Resources to Policy Objects."

4. Configure the appropriate Authorization and Role Mapping policies.

   See Chapter 4, "Managing Policies and Policy Objects."

5. Distribute the policies to the Security Module.

   See Chapter 6, "Managing Policy Distribution."

9.4.1 Integrating with WebLogic Server

As discussed in Section 1.3.2.2, "Security Module as Combination PDP / PEP," WebLogic Server can automatically intercept authorization requests after enabling the

Role Mapping and Authorization providers. The following procedure explains how to do this; it assumes the WebLogic Server is installed in the `$WLS` directory in the `$DOMAIN` domain. Replace the values based on your installation when following the procedure.

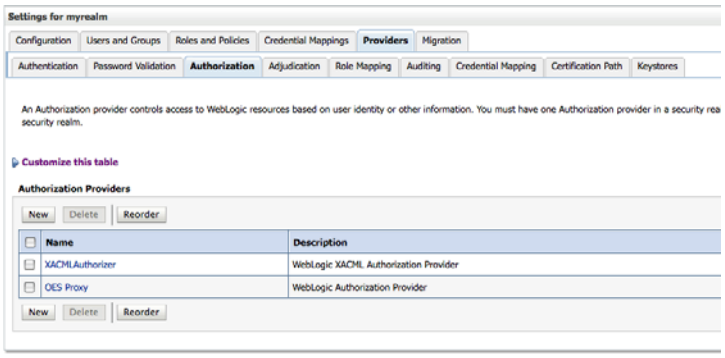
1. Start the `$DOMAIN` domain using the following command.

   ```
   $DOMAIN/startWeblogic.sh
   ```

2. Add the Authorization Proxy and Role Mapping providers to the realm that protects the domain.

   Figure 9–1 is a screenshot of the WebLogic Server console that illustrates this.

Figure 9–1 Adding Providers to the WebLogic Server Domain's Realm



3. Restart the domain.

After enabling the providers, see Section A.2.5, "WebLogic Server Security Module" for the configuration parameters.

9.4.2 Discovering WebLogic Server Resources

When writing policy to secure an application's resources, all resources that must be secured must be *discovered*. By running the WebLogic Server Security Module in Discovery Mode and opening one or more user sessions (to track usage), the application's resources can be defined. (Discovery Mode does not authorize; it discovers objects to be protected.) Based on the activities performed during the user session, Oracle Entitlements Server will generate an initial policy set (that defines all resources to be protected); this policy set can then be imported into the policy store.

Note: The generated files are meant to serve as a starting point for defining a policy set to fully secure the application. In particular:

- The recorded policy data is based only on requests made during the user session; no policy data will be generated for parts of the application that are not used.
- Depending on the Resource hierarchy you use to define the application's resources, the imported policy set may contain more Resources than actually needed.

Resource discovery is enabled when the Authorization and Role Mapping providers run in Discovery Mode. In this mode, these providers always return *true* when

evaluating user requests and generate the initial policy files based on those requests. Discovery Mode may find Applications, Resource Types (and corresponding actions), the Resource Type `matcherClass` name, and Resources. The following sections contain more information.

- Section 9.4.2.1, "Enabling Discovery Mode"
- Section 9.4.2.2, "Loading Discovered Resources"

9.4.2.1 Enabling Discovery Mode

By default, Discovery Mode is off. Setting the `oracle.security.jps.discoveryMode` property to true (in `jps-config.xml`) enables the feature. Adding a directory value for the `oracle.security.jps.discoveredPolicyDir` property defines where the policy set will be written.

> **Note:** Discovery Mode does not generate parent Resources for hierarchical Resource Types. If the administrator knows that all Resource Types to be discovered are hierarchical, add the appropriate values to the optional `oracle.security.jps.discoveredResourceIsHierarchical` and `oracle.security.jps.discoveredResourceNameDelimiter` properties.

See Appendix A, "Installation and Configuration Parameters" for additional details on these configuration parameters.

9.4.2.2 Loading Discovered Resources

The resulting Discovery Mode file follows the `jazn-data.xml` schema and is a standard XML policy store file as illustrated in Example 9–8. Use the Oracle Entitlements Server API to create the discovered objects as Authorization Policy objects in the policy store. See Section 13.5, "Migrating Policies" for information.

Example 9–8 Sample File Of Discovered Resources

```
<?xml version = '1.0' encoding = 'UTF-8' standalone = 'yes'?>
<jazn-data xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation=
  "http://xmlns.oracle.com/oracleas/schema/jazn-data-11_0.xsd">
<policy-store>
  <applications>
    <application>
      <name>addConfRes#V2.0</name>
        <resource-types>
          <resource-type>
            <name>FileResourceType</name>
              <matcher-class>oracle.security.jps.JpsPermission</matcher-class>
                 <actions-delimiter>,</actions-delimiter>
                 <actions>delete,write,read</actions>
          </resource-type>
        <resource-type>
            <name>ResType1</name>=
                 <actions-delimiter>,</actions-delimiter>
                 <actions>write,read</actions>
        </resource-type>
       </resource-types>
       <resources>
```

```
          <resource>
            <name>EmpInfo</name>
            <type-name-ref>FileResourceType</type-name-ref>
          </resource>
          <resource>
            <name>resource1</name>
            <type-name-ref>ResType1</type-name-ref>
          </resource>
        </resources>
      </application>
    </applications>
  </policy-store>
</jazn-data>
```

9.4.3 Converting WebLogic Server Resources

This section describes how Oracle Entitlements Server converts the different resource types supported by WebLogic Server and how they are represented in the Oracle Entitlements Server Administration Console. A WebLogic Server resource is an object that the WebLogic Security Service creates to represent an underlying WebLogic Server entity; it is used to determine who can access the entity.

An Authorization Policy defines, among other objects, a top-level Application, a Resource Type and the actual Resource to be protected. The objects may include those documented in Table 9–4.

Table 9–4 WebLogic Server Authorization Policy Objects

Node	Description
Application	The Application corresponds to the application with which the Resource is associated. Not every resource belongs to a specific Application; for example, a JDBC resource does not. In these cases, `shared` substitutes for the name of the Application. The Application name is defined when the Application is created using the Administration Console; for example, `MyEjbApplication`.
Resource Type	The Resource Type corresponds to the supported WebLogic Server resource types as defined in Section 9.4.4. The Resource Type name is defined when the Resource Type is created using the Administration Console; for example, `ejb`. A Resource Type can be defined as hierarchical or flat.
Resource	The Resource is the instance of the Resource Type that is being protected. The Resource can be hierarchical (as a directory in which protection is afforded to all contents) or a specific file. The Resource Type from which the Resource instance is created must first be defined as hierarchical.

9.4.4 Mapping WebLogic Server Resources to Policy Objects

This section describes how to map Oracle Entitlements Server policy objects to WebLogic Server resources defined for common external resources. WebLogic Server supports the following Resource Type values: `adm`, `app`, `com`, `eis`, `ejb`, `jdbc`, `jms`, `jndi`, `ld`, `svr`, `url`, `web`, `webservices`. It contains information on the following types of resources.

- Section 9.4.4.1, "Enterprise Java Bean Resources"
- Section 9.4.4.2, "Java Naming and Directory Interface Resources"
- Section 9.4.4.3, "URL Resources"
- Section 9.4.4.4, "JDBC Resources"

- Section 9.4.4.5, "JMS Resources"
- Section 9.4.4.6, "Web Services Resources"
- Section 9.4.4.7, "Server Resources"

9.4.4.1 Enterprise Java Bean Resources

When defining objects for a policy that will be used to protect an Enterprise Java Bean (EJB) resource, policy objects should be named based on values defined in the standard EJB deployment descriptor, `ejb-jar.xml`. Example 9–9 illustrates how one EJB named `AccountService` might be defined.

Example 9–9 Defining an EJB Resource in ejb-jar.xml

```
<enterprise-beans>
<!-- Session Beans -->
  <session>
    <display-name>AccountService</display-name>
    <ejb-name>AccountService</ejb-name>
    <home>com.bea.security.examples.ejb.AccountServiceHome</home>
    <remote>com.bea.security.examples.ejb.AccountService</remote>
    <ejb-class>ejb.AccountServiceSession</ejb-class>
    <session-type>Stateless</session-type>
    <transaction-type>Bean</transaction-type>
  </session>
</enterprise-beans>
```

Table 9–5 contains the mappings that should be used when defining policy objects for use with an EJB resource.

Table 9–5 Mapping EJB Definitions to Policy Objects

Policy Object Name	EJB Definition
Application	Same as the EJB name; in this case, `AccountService`
Resource Type	Use the value `ejb`
Resource name	*ejb_name*/*method_name* where: ■ *ejb_name* is the name of the EJB ■ *method_name* is the name of the invoked method The EJB method is part of the resource URL. The Resource action is always `execute`.

The following list documents the attributes supported by JNDI resources that can be used as a part of a Condition in an Authorization Policy. See Section 4.6, "Using the Condition Builder" for details.

- application: name of the application
- module: name of the module
- ejb: name of the EJB
- method: name of the method
- methodinterface: Takes as a value Home, Remote, LocalHome, or Local
- Param*N*: A value of the *N*th parameter in the method; for example, Param1, Param2…

Note: Before using the Condition Builder, the dynamic attributes first have to be created using the Oracle Entitlements Server Administration Console. See Section 4.5.9, "Managing Attributes and Functions as Extensions" for more details.

9.4.4.2 Java Naming and Directory Interface Resources

When defining objects for a policy that will be used to protect Java Naming and Directory Interface (JNDI) based resources, policy objects should be named based on values defined in theWebLogic-specific deployment descriptor, `weblogic-ejb-jar.xml`. Example 9–10 illustrates how an EJB named `AccountService` might be defined with a JNDI name.

Example 9–10 Defining a JNDI Resource in weblogic-ejb-jar.xml

```
<weblogic-ejb-jar>
  <weblogic-enterprise-bean>
    <ejb-name>AccountService</ejb-name>
    <stateless-session-descriptor></stateless-session-descriptor>
    <reference-descriptor></reference-descriptor>
    <jndi-name>AccountService</jndi-name>
  </weblogic-enterprise-bean>
</weblogic-ejb-jar>
```

Table 9–6 contains the mappings that should be used when defining policy objects for use with a JNDI based resource.

Table 9–6 Mapping JNDI Definitions to Policy Objects

Policy Object Name	JNDI Definition
Application	shared
Resource Type	jndi
Resource	Not used

The action for a JNDI call is the JNDI action name. The value can be one of the following.

- `modify` is required whenever an application modifies (add, remove, change) the JNDI tree in any way. This includes the `bind()`, `rebind()`, `createSubContext()`, `destroySubContext()`, and `unbind()` methods.
- `lookup` is required whenever an application looks up an object in the JNDI tree. This includes the `lookup()` and `lookupLink()` methods.
- `list` is required whenever an application lists the contents of a context in JNDI. This includes the `list()` and `listBindings()` methods.

The following list documents the attributes supported by JNDI resources that can be used as a part of a Condition in an Authorization Policy. See Section 4.6, "Using the Condition Builder" for details.

- application - Always *shared*
- path - The JNDI resource path
- action - the JNDI action name (modify | lookup | list)

Note: Before using the Condition Builder, the dynamic attributes first have to be created using the Oracle Entitlements Server Administration Console. See Section 4.5.9, "Managing Attributes and Functions as Extensions" for more details.

9.4.4.3 URL Resources

A URL (Web) resource is a specific WebLogic Server resource related to Web applications. To secure Web applications, create Authorization Policies for a Web Application aRchive (WAR) or for individual components of the Web application (such as servlets and JSPs). Table 9–7 describes how to name the Oracle Entitlements Server objects when securing a URL resource. These values are defined when the objects are created using the Oracle Entitlements Server Adminstration Console.

Table 9–7 URL Resource Values Mapped to Oracle Entitlements Server Objects

OES Object Name	URL Resource Value
Application	Takes as a value the web application name (defined in the WebLogic Server configuration file) that is (or contains) the resource; for example, `bankapp`.
Resource Type	Takes as a value one of the supported resource types; in this case, `url`.
Resource parent	Takes as a value the context path of the web application as defined in the WebLogic Server configuration file. In the following examples, the context path is defined as `/currencyExchange` for the first two policies and `/mybroker` for the last two.
Resource	Takes as a value the resource URI after the context path. In this case, `currentRates.jsp`.

To illustrate how to create an Authorization Policy for a URL resource, let's assume we want to protect Web resources accessible through different banking related JSP. The WebLogic Server configuration file references the web application name as `bankapp` with the context path `/currencyExchange` (for the first two policies) and `/mybroker` (for the last two policies). In the case of a URL resource, the action name is mapped to the HTTP request method name (GET, POST, PUT, HEAD, DELETE, TRACE, CONNECT, and the like).

The first Authorization Policy example grants any unauthorized user (anonymous) permission to view current currency exchange rates (`currentRates.jsp`) if the connection through which the page is being accessed is secure (HTTPS). Create the following objects using the Administration Console:

- Application = bankapp
- Resource Type = url
- Resource = currencyExchange/currentRates.jsp
- Action = GET
- User = anonymous (unauthorized)
- Condition = if issecure=yes

Notice the *GET* action is part of the Authorization Policy; it is not the action of the Authorization Policy. The full range of actions allowed on the Resource Type are always defined as part of the Resource Type profile. The policy action is always equal to GRANT or DENY.

The second Authorization Policy example grants any member of the Manager role permission to post new currency exchange rates (`postNewRates.jsp`) if the user updates the data from the local machine. Create the following objects using the Administration Console:

- Application = bankapp
- Resource Type = url
- Resource = currencyExchange/postNewRates.jsp
- Action = POST
- Role = Manager
- Condition = if remotehost="localhost"

Notice the *POST* action is part of the Authorization Policy; it is not the action of the Authorization Policy. The full range of actions allowed on the Resource Type are always defined as part of the Resource Type profile. The policy action is always equal to GRANT or DENY.

The third Authorization Policy example grants access to `buyStocks.jsp` if the customer has positive purchasing power. The page will not be displayed if a customer's purchasing power is not positive. To decipher purchasing power, when the customer clicks on the `buyStocks.jsp` link, the browser sends an HTTP request mapped to a Java servlet. The servlet sets a request attribute named `purchasingPower` and forwards the request to a second page that is responsible for fetching balances from all of the customer's accounts, calculating the amount of money that can be spent on buying new stocks (purchasing power) and populating the `purchasingPower` attribute with a value. Create the following objects using the Administration Console:

- Application = bankapp
- Resource Type = url
- Resource = mybroker/buyStocks.jsp
- Action = GET
- Role = Client
- Condition = if purchasingPower>0

Notice the *GET* action is part of the Authorization Policy; it is not the action of the Authorization Policy. The full range of actions allowed on the Resource Type are always defined as part of the Resource Type profile. The policy action is always equal to GRANT or DENY.

The fourth Authorization Policy allows a customer to open an account for trading stocks only if the Trading Agreement has been accepted (by ticking the checkbox). After clicking the `openAccount.jsp` link, the first page displayed contains the Trading Agreement and asks the customer to accept it. The checkbox is linked to an HTML form parameter named `customerAgreed`. When the HTML form is posted, this parameter is set to true if the customer has accepted the trading agreement. The policy checks for this value in the `customerAgreed` HTTP request parameter. Create the following objects using the Administration Console:

- Application = bankapp
- Resource Type = url
- Resource = mybroker/openAccount.jsp
- Action = POST

- Role = Client
- Condition = if Not customerAgreed="true"

Notice the *POST* action is part of the Authorization Policy; it is not the action of the Authorization Policy. The full range of actions allowed on the Resource Type are always defined as part of the Resource Type profile. The policy action is always equal to GRANT or DENY.

Table 9–8 documents the dynamic attributes supported by URL resources that can be used as a part of a Condition in an Authorization Policy. See Section 4.6, "Using the Condition Builder" for details.

Table 9–8 Dynamic Attributes Supported by URL Resources

Attribute Name	Value
application	The name of the web application.
contextpath	The context path of the web application.
uri	The URI of the resource.
httpmethod	The HTTP method (same as action).
transporttype	The transport guarantee required to access the URL resource, as it appears in the corresponding <transport-guarantee> element in the deployment descriptor. The value can be one of INTEGRAL or CONFIDENTIAL.
authtype	The name of the authentication scheme used to protect the servlet. The value can be one of: BASIC, FORM, CLIENT_CERT or DIGEST.
pathInfo	Extra path information associated with the URL sent by the client when it made a request.
pathtranslated	Extra path information after the servlet name but before the query string is translated to a real path.
querystring	The query string that is contained in the request URL after the path.
remoteuser	The login of the user making the request, if the user has been authenticated.
requestedsessionid	The session ID specified by the client.
requesturi	The part of this request's URL from the protocol name up to the query string in the first line of the HTTP request.
requesturl	The URL used by the client to make the request. The returned URL contains a protocol, server name, port number, and server path, but it does not include query string parameters.
servletpath	The part of this request's URL that calls the servlet.
characterencoding	The character encoding used in the body of the request.
contenttype	The MIME type of the body of the request.
locale	The preferred Locale of the client.
protocol	The name and version of the protocol, for example, HTTP/1.1.
remoteaddr	The Internet Protocol address of the client or last proxy that sent the request.
remotehost	The fully qualified name of the client or the last proxy that sent the request.
scheme	The name of the scheme used to make this request, for example, http, https, or ftp.

Table 9–8 (Cont.) Dynamic Attributes Supported by URL Resources

Attribute Name	Value
servername	The host name of the server to which the request was sent.
serverport	The port number to which the request was sent.
issecure	A boolean indicating whether this request was made using a secure channel, such as HTTPS.

Note: Before using the Condition Builder, the dynamic attributes first have to be created using the Oracle Entitlements Server Administration Console. See Section 4.5.9, "Managing Attributes and Functions as Extensions" for more details.

HTTP requests may contain elements such as servlet attributes, URL query parameters, HTTP request headers and cookies. These elements, available as name/value pairs, can be mapped to dynamic attributes.

Note: The attributes that correspond to servlet attributes, URL query parameters, HTTP request headers and cookies are case insensitive; however, an assumption that the attribute names are case sensitive will slightly improve the performance.

The order in which the framework searches for a matching attribute is:

1. URL query parameters - are name/value pairs appended to a URL. The attribute names that correspond to the parameters in a URL query string are the same as the parameter names. The names are represented as strings and are case insensitive. The attributes refer to the query string variable encoded within the request. For example, if a URL includes a query such as `?test=endcoded%20char`, the parameter can be accessed in the Condition of an Authorization Policy as: `"if test= "encoded char"`
2. Servlet attributes - are name/value pairs that can be added to a request internally by a servlet container. Usually this is accomplished by calling the `setAttribute` method of the `ServletRequest` interface. The policy attribute names correspond to the names of servlet attributes, and are represented as strings and case insensitive.
3. HTTP request headers - The attribute name of an HTTP request header corresponds to the name of the header. The name is returned as a string and is case insensitive. Examples of the available headers are: date, if-modified-since, referrer, or user-agent. (The date header, usually a date type, is returned as a string.)
4. cookies - The attribute names that correspond to cookies in an HTTP request are the same as the cookie name in the request. The names are returned as strings and case insensitive. The value of the cookie returned is application-specific and may need further decoding.

Note: If the names of a servlet attribute, URL query parameter, HTTP request header or cookie collide, only one attribute will be available in policy constraints.

9.4.4.4 JDBC Resources

A Java DataBase Connectivity (JDBC) resource is a WebLogic Server resource that is related to JDBC. You can secure JDBC resources that are deployed as a service or as an application. To secure JDBC database access, create Authorization Policies for all data sources as a group, individual data sources, and multiple data sources. Example 9–11 shows how a JDBC resource named `MyJDBCConnectionPool` could be defined in the WebLogic Server configuration file, `config.xml`.

Example 9–11 Defining a JDBC Resource in config.xml

```
<JDBCConnectionPool DriverName="oracle.jdbc.driver.OracleDriver"
            Name="MyJDBCConnectionPool"
            PasswordEncrypted="{3DES}B2Bl+tp70Eh3D1pT53/anw=="
            Properties="user=wles" Targets="myserver"
            TestTableName="SQL SELECT 1 FROM DUAL"
            URL="jdbc:oracle:thin:@localhost:1521:ASI"/>
<JDBCTxDataSource JNDIName="MyDataSource"
            Name="MyJDBCDataSourceName"
            PoolName="MyJDBCConnectionPool"
            Targets="myserver"/>
```

Table 9–9 describes how to name the Oracle Entitlements Server objects in the case of securing the JDBC resource, `MyJDBCConnectionPool`. These values are defined when the objects are created using the Oracle Entitlements Server Adminstration Console

Table 9–9 JDBC Values Mapped to Oracle Entitlements Server Objects

OES Object Name	JDBC Value
Application	A JDBC resource does not belong to a specific Application. In these cases, `shared` substitutes for the name of the Application. The Application, shared, is defined when the Application is created using the Administration Console.
Resource Type	Takes as a value one of the supported resource types; in this case, `jdbc`.
Resource parent	Takes as a value the module name (if any) plus the pool type (ConnectionPool or MultiPool); in this case, ConnectionPool.
Resource	Takes as a value the name of the JDBC resource as defined in `config.xml`. In this case, `MyJDBCConnectionPool`.

To illustrate how to create policy objects using Oracle Entitlements Server for a JDBC resource, let's assume we want to grant members of the `ExternalApplication` role permission to `reserve` (open) a JDBC connection from a connection pool called ExternalDataPool. Create the following objects using the Administration Console:

- Application = shared
- Resource Type = jdbc
- Resource = ConnectionPool/ExternalDataPool
- Action = reserve

This second group of objects will be used in a policy to grant members of the Admin role permission to shut down any JDBC resource except the resource named `SystemJdbcPool`. Create the following objects using the Administration Console:

- Application = shared

- Resource Type = jdbc
- Resource = ConnectionPool/ExternalDataPool

Note that the `//app/policy/mybank/shared/jdbc` resource must be a virtual one.

- Action = admin
- Condition = if Not resource="SystemJdbcPool"

Notice the *reserve* and *admin* actions are part of the Authorization Policies; it is not the action of the Authorization Policy. The full range of actions allowed on the Resource Type are always defined as part of the Resource Type profile. The policy action is always equal to GRANT or DENY. Table 9–10 documents the specific actions that can be performed on a JDBC resource and thus must be defined as actions of the Resource Type.

Table 9–10 JDBC Resource Action Options

Action Name	Operation
admin	Action to perform the admin operations such as clearStatementCache, suspend, forceSuspend, resume, shutdown, forceShutdown, start, getProperties, and poolExists.
reserve	Action to reserve a connection in the data source by looking up the data source and then calling getConnection.
shrink	Action to shrink the number of connections in the data source.
reset	Action to reset the data source connections by shutting down and re-establishing all physical database connections.

Example 9–12 is sample code that uses the JDBC resource previously defined. It calls the `getConnection()` method on the data source instance. This initiates an authorization check to verify the reserve action against the `//app/policy/AppParentNode/shared/jdbc/ConnectionPool/MyJDBCConnectionPool` resource.

Example 9–12 Initiating Authorization on JDBC Resource

```
javax.naming.InitialContext initialContext = new javax.naming.InitialContext();
javax.sql.DataSource ds = (javax.sql.DataSource)
initialContext.lookup("MyDataSource");
java.sql.Connection conn = ds.getConnection();
PreparedStatement statement =
conn.prepareStatement("SELECT accountName FROM accounts WHERE balance < 0");
ResultSet result = statement.executeQuery();
if (result.next()) {
    String accountName = result.getString(1);
    System.out.println("The first account with negative balance is " +
accountName);
}
```

Table 9–11 documents the attributes supported by JDBC resources that can be used as a part of a Condition in an Authorization Policy. See Section 4.6, "Using the Condition Builder" for details.

Table 9–11 Dynamic Attributes Supported by JDBC Resources

Attribute Name	Element
application	The name of an application that hosts the resource
module	The name of a module to which the resource belongs
category	The resource type (ConnectionPool | MultiPool)
resource	The name of the resource
action	The JDBC operation name (admin | reserve | shrink | reset)

Note: Before using the Condition Builder, the dynamic attributes first have to be created using the Oracle Entitlements Server Administration Console. See Section 4.5.9, "Managing Attributes and Functions as Extensions" for more details.

9.4.4.5 JMS Resources

A Java Messaging Service (JMS) resource is a WebLogic Server resource related to JMS. You can secure JMS resources that are deployed as a service or as an application. To secure JMS destinations, create Authorization Policies for all destinations (JMS queues and JMS topics) as a group, or individually (one JMS queue or JMS topic on a JMS server). Table 9–12 describes how to name the Oracle Entitlements Server objects in the case of securing a JMS resource. These values are defined when the objects are created using the Oracle Entitlements Server Adminstration Console.

Table 9–12 JMS Values Mapped to Oracle Entitlements Server Objects

OES Object Name	JMS Value
Application	A JMS resource does not belong to a specific Application. In these cases, `shared` substitutes for the name of the Application. The Application, shared, is defined when the Application is created using the Administration Console.
Resource Type	Takes as a value one of the supported resource types; in this case, `jms`.
Resource parent	The destination type (topic or queue)
Resource	The resource name

Example 9–13 configures a JMS queue named `MyJMSQueue` in the WebLogic Server configuration file, `config.xml`.

Example 9–13 Defining a JMS Queue Resource in config.xml

```
<JMSServer Name="WSStoreForwardInternalJMSServermyserver"
           Store="FileStore" Targets="myserver">
  <JMSQueue CreationTime="1150241964468"
           JNDIName="JMSQueue" Name="MyJMSQueue"/>
</JMSServer>

<JMSConnectionFactory JNDIName="JmsConnectionFactory"
          Name="MyJMSConnectionFactory" Targets="myserver"/>
```

Example 9–14 illustrates a JMS client that uses the JMS queue previously declared. The client sends a text message to `MyJMSQueue`.

Example 9–14 JMS Client Example

```
//Instantiate the inital context
javax.naming.InitialContext initialContext = new javax.naming.InitialContext();

//Look up the JMS connection factory and the message queue
Queue messageQueue = (Queue) initialContext.lookup("JMSQueue");
JMSConnectionFactory factory =
    (JMSConnectionFactory) initialContext.lookup("JmsConnectionFactory");

//Create the queue connection and session
QueueConnection queueConnection = factory.createQueueConnection();
QueueSession session =
    queueConnection.createQueueSession(false, Session.AUTO_ACKNOWLEDGE);

//Create a text message
TextMessage textMessage = session.createTextMessage();
textMessage.setText("Hello from the client!");

//Send message to the queue
QueueSender sender = session.createSender(messageQueue);
sender.send(textMessage);
```

To illustrate how to create an Authorization Policy for a JMS resource, let's assume we want to grant members of the Client role permission to send messages to a JMS queue named `FeedbackQueue`. Create the following objects using the Administration Console:

- Application = shared
- Resource Type = jms
- Resource = queue/FeedbackQueue
- Action = send
- Role = Client

This second Authorization Policy grants the `FeedbackProcessor` user permission to receive messages from a JMS queue named `FeedbackQueue`.

- Application = shared
- Resource Type = jms
- Resource = queue/FeedbackQueue
- Action = send
- user = myusers/FeedbackProcessor

Notice the actions (send and receive) are part of the Authorization Policies. The policy outcome is always equal to GRANT or DENY. Table 9–13 documents the specific actions that can be performed on a JMS resource.

Table 9–13 JMS Resource Action Options

Action	Description
send	Required to send a message to a queue or a topic. This includes calls to the `MessageProducer.send()`, `QueueSender.send()`, and `TopicPublisher.publish()` methods.

Table 9–13 (Cont.) JMS Resource Action Options

Action	Description
receive	Required to create a consumer on a queue or a topic. This includes calls to the `Session.createConsumer()`, `Session.createDurableSubscriber()`, `QueueSession.createReceiver()`, `TopicSession.createSubscriber()`, `TopicSession.createDurableSubscriber()`, `Connection.createConnectionConsumer()`, `Connection.createDurableConnectionConsumer()`, `QueueConnection.createConnectionConsumer()`, `TopicConnection.createConnectionConsumer()`, and `TopicConnection.createDurableConnectionConsumer()` methods.
browse	Required to view the messages on a queue using the QueueBrowser interface.

Table 9–14 documents the attributes supported by JMS resources that can be used as a part of a Condition in an Authorization Policy. See Section 4.6, "Using the Condition Builder" for details.

Table 9–14 Dynamic Attributes Supported by JMS Resources

Attribute Name	Description
application	The name of an application that hosts the resource
destinationtype	The JMS destination type (queue | topic)
resource	The name of the resource
action	The JDBC operation name (send | receive | browse).

Note: Before using the Condition Builder, the dynamic attributes first have to be created using the Oracle Entitlements Server Administration Console. See Section 4.5.9, "Managing Attributes and Functions as Extensions" for more details.

9.4.4.6 Web Services Resources

A Web Services resource is a WebLogic Server resource related to a Web service. To secure Web services, create Authorization Policies for the entire Web Service resource, for a subset of the Web Service resource operations, for the stateless session EJB that implements the Web Service resource, or for a subset of the methods within the stateless session EJB. Example 9–15 shows the configuration of a web application named `BasicWS` that contains a Web service implementation named `BasicWS_Component`.

Example 9–15 Web Application Configuration

```
<application Name="BasicWS"
             Path="applications/BasicWS.ear"
             StagedTargets="myserver"
<WebServiceComponent Name="BasicWS_Component"
                     Targets="myserver"
                     URI="BasicWS.war"/>
</application>
```

Example 9–16 shows how the `application.xml` file within the `BasicWS.ear` defines the web application context.

Example 9–16 Web Application Context Configuration

```
<module>
  <web>
    <web-uri>basic_javaclass.war</web-uri>
    <context-root>myservices</context-root>
  </web>
</module>
```

Example 9–17 shows the configuration of a Web Service named `HelloWorld`. It is defined in the `web-services.xml` descriptor file inside the web application WAR file.

Example 9–17 Web Service Configuration

```
<web-services>
  <web-service useSOAP12="false"
               name="HelloWorld"
               style="rpc"
               uri="/HelloWorld">
   <operations>
    <operation name="sayHello"
        method="sayHello(int,java.lang.String)"/>
   </operations>
  </web-service>
</web-services>
```

Table 9–15 describes how to name the Oracle Entitlements Server objects in the case of securing a Web resource. These values are defined when the objects are created using the Oracle Entitlements Server Adminstration Console.

Table 9–15 Web Services Values Mapped to Oracle Entitlements Server Objects

OES Object Name	Web Services Value
Application	The name of the application as defined in the `web-services.xml` file; in this case, `BasicWS`.
Resource Type	Takes as a value one of the supported resource types; in this case, `webservices`.
Resource parent	Takes as a value the context path of the web service as defined in the `<context-root>` element of the `application.xml` configuration file; in this case, `myservices`.
Resource	The Web Service name as defined in `web-services.xml`.

To call the `sayHello()` method in the `HelloWorld` Web service, the client must be granted the action `sayHello`. Some clients may also require access to the Web Services Definition Language (WSDL) file that defines the Web service; the WSDL file is defined as a URL resource. Example 9–18 is code that allows the client to, before calling the `sayHello()` method, access the WSDL file at the defined URL.

Example 9–18 Client Code For Accessing WSDL

```
String wsdlUrl = "http://localhost:7001//HelloWorld?WSDL";
HelloWorld service = new HelloWorld_Impl(wsdlUrl);
HelloWorldPort port = service.getHelloWorldPort();
String result = port.sayHello(34, "Josh");
```

To successfully execute this code, the client must be granted GET permission on the WSDL file (URL resource). (Note that for the URL Resource name is lower case.) Additionally, the client must be granted GET permission on the Web Service resource.

(Note that for the Web Services Resource name has initial capitalization.) Create the following objects for the URL Resource Authorization Policy using the Administration Console:

- Application = BasicWS
- Resource Type = url
- Resource = myservices/helloworld
- Action = GET
- Role = SomeUser

Create the following objects for the Web Services Resource Authorization Policy using the Administration Console:.

- Application = BasicWS
- Resource Type = webservices
- Resource = myservices/HelloWorld
- Action = sayHello
- Role = SomeUser

Notice the actions (GET and sayHello) are part of the Authorization Policies. The policy outcome is always equal to GRANT or DENY.

Table 9–16 documents the attributes supported by Web Services resources that can be used as a part of a Condition in an Authorization Policy. See Section 4.6, "Using the Condition Builder" for details.

Table 9–16 Dynamic Attributes Supported by Web Services Resources

Attribute Name	Value
application	The name of the application
contextpath	The context part of the web application
webservice	The name of the web service
method	The name of the web service operation called
Param*N*	A value of the *N*th parameter in the method, for example, Param1, Param2…

> **Note:** Before using the Condition Builder, the dynamic attributes first have to be created using the Oracle Entitlements Server Administration Console. See Section 4.5.9, "Managing Attributes and Functions as Extensions" for more details.

9.4.4.7 Server Resources

A Server resource determines who can control the state of a WebLogic Server instance. When users start server instances by invoking the `weblogic.Server` class in a Java command, the policy on the Server resource is the only security check that occurs. You can create Authorization Policies that apply to all WebLogic Server instances in a domain or to individual servers. Example 9–19 is an example of how a WebLogic Server instance named `myserver` might be configured.

Example 9–19 Configuration of WebLogic Server Instance

```
<Server ListenAddress=""
        ListenPort="7001"
        Machine="mymachine"
        Name="myserver"
        NativeIOEnabled="true"
        ReliableDeliveryPolicy="RMDefaultPolicy"
        ServerVersion="8.1.5.0">
    <SSL Enabled="false" HostnameVerificationIgnored="false"
        IdentityAndTrustLocations="KeyStores" Name="myserver"/>
</Server>
```

Table 9–17 describes how to name the Oracle Entitlements Server objects in the case of securing the a Server resource. These values are defined when the entities are created using the Oracle Entitlements Server Adminstration Console

Table 9–17 Server Resource Values Mapped to Oracle Entitlements Server Objects

OES Object Name	Server Resource Value
Application	A Server resource does not belong to a specific Application. In these cases, `shared` substitutes for the name of the Application. The Application, shared, is defined when the Application is created using the Administration Console.
Resource Type	Takes as a value one of the supported resource types; in this case, `svr`.
Resource	The server instance name as defined in XXXX

To illustrate how to create an Authorization Policy for a Server resource, let's assume we want to grant members of the Admin role permission to boot all WebLogic Server instances. Create the following objects using the Administration Console:

- Application = shared
- Resource Type = svr
- Resource = /lib/*
- Action = boot
- Role = Admin

This second Authorization Policy grants members of the Admin role permission to shutdown or suspend a WebLogic Server instance named CentralServer. The policy is constrained in that permission is granted only on Sundays or other days between 2 AM and 4 AM.

- Application = shared
- Resource Type = svr
- Resource = CentralServer
- Action = shutdown / suspend
- Role = Admin
- Condition = Only on Sunday or other days between 2 AM and 4 AM

Notice the actions (boot and shutdown/suspend) are part of the Authorization Policies. The policy outcome is always equal to GRANT or DENY. Table 9–18 documents the specific actions that can be performed on a Server resource.

Table 9–18 Server Resource Action Options

Action	Description
boot	Action required to start a WebLogic Server instance, either an Administration Server or Managed Server.
shutdown	Action required to shut down a running WebLogic Server instance, either an Administration Server or Managed Server.
suspend	Action required to prohibit additional logins (logins other than for privileged administrative actions) to a running WebLogic Server instance, either an Administration Server or Managed Server.
resume	Action required to re-enable non-privileged logins to a running WebLogic Server instance, either an Administration Server or Managed Server.

Table 9–19 documents the attributes supported by Server resources that can be used as a part of a Condition in an Authorization Policy. See Section 4.6, "Using the Condition Builder" for details.

Table 9–19 Dynamic Attributes Supported by Server Resource

Attribute Name	Value
server	Name of the server with which the resource is associated.
action	Name of an operation performed on the server instance (boot | shutdown | suspend | resume).

Note: Before using the Condition Builder, the dynamic attributes first have to be created using the Oracle Entitlements Server Administration Console. See Section 4.5.9, "Managing Attributes and Functions as Extensions" for more details.

10

Managing System Configurations

Security Module definitions and administrator configurations are defined within the top-level System Configuration tab in the Authorization Policy Manager Administration Console. This chapter contains the following topics:

- Delegating With Administrators
- Configuring Security Module Definitions
- Configuring Identity Directory Service Profiles

10.1 Delegating With Administrators

Administrator Roles can be created to delegate management operations for policy objects. For example, Application and Policy Domain delegating administrators can be defined by creating an Administrator Role at the appropriate level and assigning the role Administration Privileges as well as a user, group, or another role. See Chapter 11, "Delegating With Administrator Roles" for more information. It includes a section on creating System Administrator Roles which can manage other types of Administrator Roles in any Application or Policy Domain.

10.2 Configuring Security Module Definitions

A Security Module is an Oracle Entitlements Server client that plays a key role in authorization. After an authorization request is generated, the Security Module evaluates policy data to determine if access to the resource will be granted or denied. An Application (the Oracle Entitlements Server object that represents the protected resource) must be *bound* to the Security Module that protects it. Binding Security Modules enables policy data to be transmitted to it for evaluation. The Policy Distribution Component (discussed in Chapter 6, "Managing Policy Distribution") is the mechanism used to transmit policy data to the Security Modules.

> **Note:** For more information about the authorization process, see Section 1.4, "How Oracle Entitlements Server Processes Authorization Policies."

The following sections document how to bind (and unbind) Security Module definitions to (and from) Application objects.

- Section 10.2.1, "Creating a Security Module Definition"
- Section 10.2.2, "Binding an Application to a Security Module"
- Section 10.2.3, "Unbinding an Application From a Security Module"

- Section 10.2.4, "Deleting a Security Module Definition"

> **Note:** Before binding Security Module definitions, you must install and configure the Security Module. See Chapter 8, "Managing Security Module Configurations" for details.

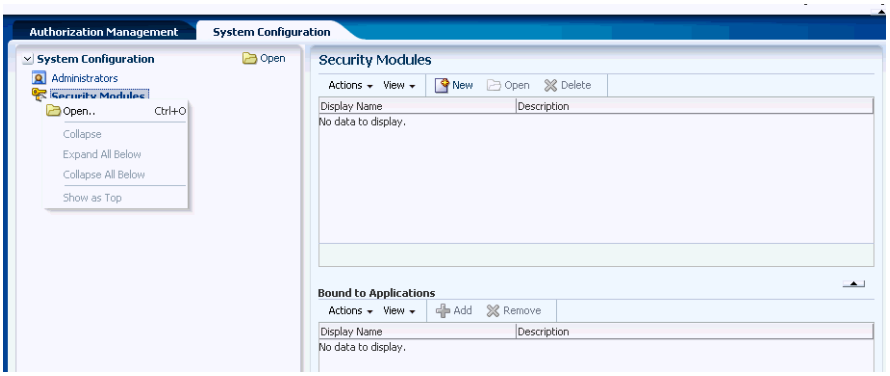
10.2.1 Creating a Security Module Definition

To create a security module, proceed as follows.

1. Select the System Configuration tab from the Home area.
2. Double-click Security Modules in the Navigation Panel.

   Alternately, right-click Security Modules and select Open. The Security Modules page is displayed as in Figure 10–1.

Figure 10–1 Security Modules in Home Area



3. Click New to create a new Security Module definition.

   Alternately, select New from the Actions menu. The Security Module dialog is displayed.
4. Provide the following values for the new Security Module.
   - **Name**: The entry must be a unique.
   - **Display Name**
   - **Description**
5. Click Save.

10.2.2 Binding an Application to a Security Module

To bind an Application to a Security Module, proceed as follows.

1. Select the System Configuration tab from the Home area.
2. Double-click Security Modules in the Navigation Panel.

   Alternately, right-click Security Modules and select Open. The Security Modules page is displayed.
3. Select the name of the Security Module definition from the table.

4. Click Add in the Bound to Applications table to display the Add Applications dialog.

   Alternately, select Add from the Bound to Applications Actions menu to display the Add Applications dialog.

5. Enter a search string in the text box and click the arrow to search.

   Alternately, click the arrow with no search string to return all available Applications.

6. Select one or more applications from the list returned.

7. Click Add.

   The selected applications are bound to the selected Security Module and displayed in the Bound to Applications table.

10.2.3 Unbinding an Application From a Security Module

To unbind an application from a Security Module, proceed as follows.

1. Select the System Configuration tab from the Home area.

2. Double-click Security Modules in the Navigation Panel.

   Alternately, right-click Security Modules and select Open. The Security Modules page is displayed.

3. Select the name of the applicable Security Module definition in the table.

4. Select the name of the applicable Application in the Bound to Applications table.

5. Click Remove or select Remove from the Actions menu.

   A confirmation dialog is displayed.

6. Click Unbind.

10.2.4 Deleting a Security Module Definition

To remove a Security Module definition, proceed as follows.

1. Select the System Configuration tab from the Home area.

2. Double-click Security Modules in the Navigation Panel.

   Alternately, right-click Security Modules and select Open. The Security Modules page is displayed.

3. Select the name of the applicable Security Module definition in the table.

4. Click Delete or select Delete from the Actions menu.

   A confirmation dialog is displayed.

5. Click Remove.

10.3 Configuring Identity Directory Service Profiles

When defining policies using Oracle Entitlements Server, users and groups need to be specified as the subject. This is done by searching an identity data store. The Identity Directory Service is a flexible and configurable service used by Oracle Entitlements Server as the means for accessing multiple identity data stores. The purpose of the

Identity Directory Service is to allow the management of policies that contain users or groups from identity stores not deployed with Oracle Entitlements Server itself.

An Application (the Oracle Entitlements Server object that represents the protected resource) must be *bound* to an Identity Directory Service profile to enable Oracle Entitlements Server to communicate with the identity store which the profile represents. By default, an Application is associated with the LDAP directory used for authentication when logging into the Administration Console. An administrator can change this association so that any identity data store configured with the Identity Directory Service can be used.

Note: Once an Identity Directory Service profile is bound to an Application, an Advanced or Simple Search can be initiated either globally (default option) or within the scope of the Application. The latter option searches in the identity data store configured for that Application. See Chapter 5, "Querying Security Objects" for more information.

The following sections document how to bind (and unbind) Identity Directory Service profiles to (and from) Application objects.

- Section 10.3.1, "Creating an Identity Directory Service Profile"
- Section 10.3.2, "Binding an Application to an Identity Directory Service Profile"
- Section 10.3.3, "Unbinding an Application From an Identity Directory Service Profile"
- Section 10.3.4, "Deleting an Identity Directory Service Profile"

Note: For information on the WebLogic Scripting Tool (WLST) commands for the Identity Directory Service, see *Oracle Fusion Middleware WebLogic Scripting Tool Command Reference*.

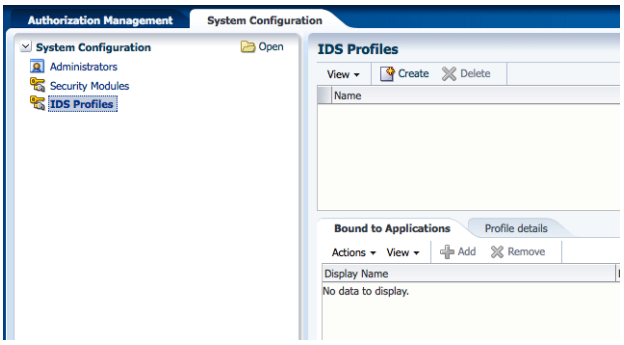
10.3.1 Creating an Identity Directory Service Profile

To create an Identity Directory Service profile, proceed as follows.

1. Select the System Configuration tab from the Home area.
2. Double-click IDS Profiles in the Navigation Panel.

   Alternately, select IDS Profiles and click Open. The IDS Profiles page is displayed as in Figure 10–2.

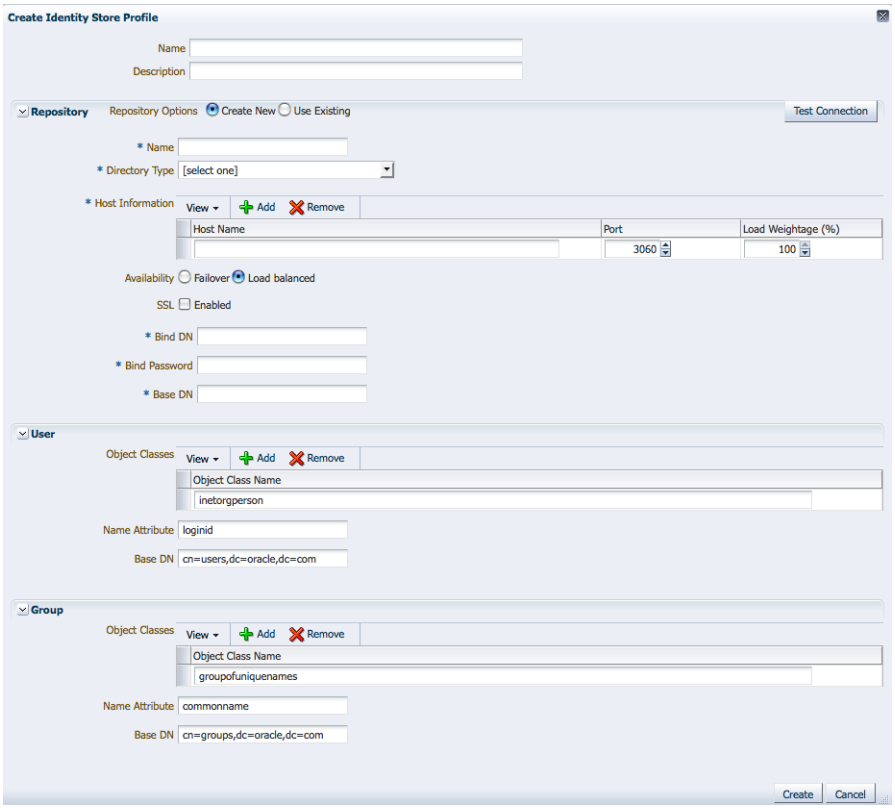
Figure 10–2 IDS Profiles in Home Area



3. Click Create to create a new Identity Directory Service profile.

   The Create Identity Store Profile page is displayed as in Figure 10–3.

Figure 10–3 Create Identity Store Profile Page



4. Provide the following values for the new Identity Directory Service profile.
   - Name: the entry must be a unique.
   - Description
5. Select Create New or Use Existing and provide the applicable values to define the repository.
   - Name: the entry must be a unique.

- Directory Type - select from the drop-down menu
- Host Information - add Host name, Port number and Load Weightage percentage
- Availability - select Failover or Load balanced
- SSL - select to enable
- Bind DN
- Bind Password
- Base DN

Click Test Connection to confirm the values are correct.

6. Provide the applicable values to define how users can be found.
   - Object Classes
   - Name Attribute
   - Base DN
7. Provide the applicable values to define how groups can be found.
   - Object Classes
   - Name Attribute
   - Base DN
8. Click Create.

   The profile is displayed in the IDS Profiles table.

10.3.2 Binding an Application to an Identity Directory Service Profile

To bind an Application to an Identity Directory Service profile, proceed as follows.

1. Select the System Configuration tab from the Home area.
2. Double-click IDS Profiles in the Navigation Panel.

   Alternately, select IDS Profiles and click Open.
3. Select the name of the Identity Directory Service profile from the upper table.
4. Click Add in the Bound to Applications table to display the Add Applications dialog.

   Alternately, select Add from the Bound to Applications Actions menu to display the Add Applications dialog.
5. Enter a search string in the text box and click the arrow to search.

   Alternately, click the arrow with no search string to return all available Applications.
6. Select one or more applications from the list returned.
7. Click Add.

   The selected applications are bound to the selected Identity Directory Service profile(s) and displayed in the Bound to Applications table.

Note: If an Application is already bound to a Identity Directory Service profile and you want to bind it to a different Identity Directory Service profile, the Application will automatically be unbound from the first profile.

10.3.3 Unbinding an Application From an Identity Directory Service Profile

To unbind an application from an Identity Directory Service profile, proceed as follows.

1. Select the System Configuration tab from the Home area.
2. Double-click IDS Profiles in the Navigation Panel.

   Alternately, select IDS Profiles and click Open.
3. Select the name of the applicable Identity Directory Service profile in the table.
4. Select the name of the applicable Application in the Bound to Applications table.
5. Click Remove or select Remove from the Actions menu.

   A confirmation dialog is displayed.
6. Click Unbind.

10.3.4 Deleting an Identity Directory Service Profile

To remove an Identity Directory Service profile, proceed as follows.

1. Select the System Configuration tab from the Home area.
2. Double-click IDS Profiles in the Navigation Panel.

   Alternately, select IDS Profiles and click Open.
3. Select the name of the applicable Identity Directory Service profile in the table.
4. Click Delete.

   A confirmation dialog is displayed.
5. Click OK.

11

Delegating With Administrator Roles

System administrative rights and policy management permissions can be delegated from one administrator to another by creating Administrator Roles with restricted rights, or by granting an existing Administrator Role to a user or existing External Role. This chapter documents information on how to delegate policy and system administrative tasks. It contains the following sections:

11.1 About Delegated Administrators

Administration is when one or more authorized rights are granted to someone to do a certain job. Delegation is the ability for that someone to transfer the authorized right that has been granted them to another. In combination, we can define delegating administration as the transference of authorized rights from one to another. In Oracle Entitlements Server, administrators who are authorized to perform a task on policy objects and entities may transfer this right to others using Administration Roles. Administration Roles consist of a subject (the person to whom the role is granted), the resources (the objects to which the role pertains) and actions (view, manage/modify).

> **Note:** See Section 1.5.1, "Role-based Access Control (RBAC)" for more details on roles.

Oracle Entitlements Server allows you to define delegating Administrator Roles by assigning Administration Privileges, and mapping external roles and users, to it. When a user is logged in as an Administrator, the Navigation Panel displays only the set of Applications the logged in user is authorized to administer. In point of fact, all objects that a delegating Administrator cannot administer are hidden. Any nondefault delegating Administrator Role can perform management operations if it is granted the Admin Role with VIEW and MANAGE privileges.

Note: A nondefault Administrator Role is any Administrator Role created manually. This would not include Administrator Roles automatically created when you create an Application or a Policy Domain.

The following restrictions also apply to Administrator Roles.

- Non-system level (delegating) Administration Roles can only manage other Administration Roles within its scope. For example, an Administration Role created for Application1 can manage Administration Roles in Application 1 Policy Domains but cannot manage peer Administration Roles in Application1, or any roles in Application2 and its Policy Domains. Scope and granularity are discussed further in Chapter 11.2, "Delegating Using Scope and Granularity."
- System level Administration Roles (as discussed in Chapter 10, "Managing System Configurations") can manage delegating Administration Roles in any Application or Policy Domain.
- Nondefault Administration Roles (again, created manually) cannot manage default Administration Roles in any Application or Policy Domain.

11.2 Delegating Using Scope and Granularity

Delegated administration is all about transferring management of resources and policy objects from one person to another. The scope of the delegation (or range of objects covered by the delegation) is defined in levels. The granularity of administration defines the type of objects managed at each scope. A default Administration Role is automatically created when each scope is created; additional Administration Roles can be created later.

Note: The following is applicable to all default Administration Roles.

- Default Administrator Roles cannot be deleted individually.
- If a Policy Domain is deleted, all Administration Roles (including the default) are deleted.
- If the Application is deleted, all Administration Roles are deleted.
- Privileges assigned to default Administrator Roles cannot be modified.

From highest to lowest, the scopes and applicable granularity are as follows:

- The top-level `SystemAdmin` has privileges to manage system-level resources as well as all policy-related objects. System resources include Administrator Roles, system configurations and Security Module bindings. Policy objects include the Application objects.

Note: System Administrators have rights to all policy objects, including all Application objects and child Policy Domains but they are primarily intended to manage configurations, Application objects, and the bindings between the two.

Information on managing system level Administrator Roles is in Chapter 10, "Managing System Configurations."

- Application administrators have privileges to manage all objects in the Application to which they are assigned. One `ApplicationPolicyAdmin` is generated for each Application that is created. They are primarily intended to delegate the management of policy objects within the Application (including the Policy Domains and its children, such as Functions, Attributes, Application Roles and Resource Types). For more information, see Section 11.3, "Delegating Application Administration."
- Policy Domain administrators have privileges to manage all child objects in the Policy Domain to which they are assigned. One `PolicyDomainAdmin` is generated for each Policy Domain that is created. They are primarily created to delegate the management of policies, permissions and resources within a Policy Domain. For an overview of this concept, see Section 11.4, "Using Policy Domains to Delegate." For additional information, see Section 11.5, "Delegating Policy Domain Administration."

11.3 Delegating Application Administration

The following sections explain how to manage administrators for an Application.

- Section 11.3.1, "Adding a Delegated Administrator for An Application"
- Section 11.3.2, "Modifying or Deleting an Application's Delegated Administrator"

11.3.1 Adding a Delegated Administrator for An Application

This procedure documents how to create a new Administrator Role and assign it to the applicable roles or users. To add a delegated administrator to an Application, proceed as follows.

1. Expand the Applications node in the Navigation Panel.
2. Select the Application to modify.
3. Right-click the Application name and select Open from the menu.

   The General, Delegated Administrators, Policy Distribution and Simulation tabs are all active.
4. Click the Delegated Administrators tab.

   The Application name is listed in the displayed table. Click the arrow next to the Application name to see the default `ApplicationPolicyAdmin` created when the Application object was created. Click the Administrator Role name to display its details, in tabs, below the Delegated Administrators table.

   - Role Details
   - External Role Mapping
   - External User Mapping
5. Click New to create a new Administrator Role.

   Be sure to select the name of the Application to activate New. Alternately, select the Application and select New from the Actions menu. A New Administrator Role dialog is displayed.
6. Provide the following values for the new Administrator Role and click OK.

- **Name**: The entry must be a unique.
- **Display Name**
- **Description**

7. Select the new Administrator Role to activate its configuration tabs.

   The Role Details tab is active.

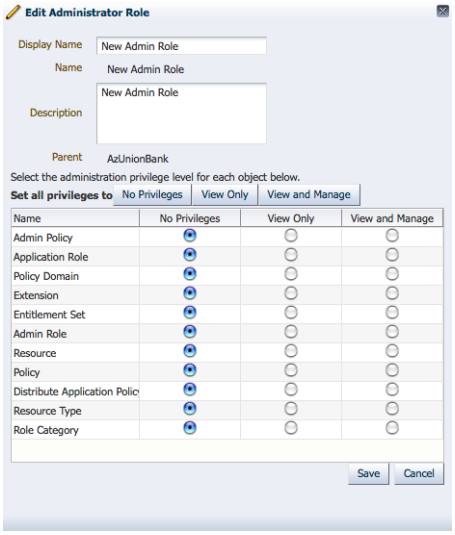
8. Click Edit to define the role details.

   An Edit Administrator Role dialog is displayed.

9. Grant View or Manage privileges for the appropriate policy objects and click Save.

   Figure 11–1 is the Edit Admin Role privileges pop up screen. Select View or Manage for the listed policy objects. For example, Admin Policy allows the administrator to assign new permissions to an Admin Role. Admin Role, however, allows the administrator to assign members to an Admin Role. See Section 2.3, "The Policy Object Glossary" for details on the other listed objects.

Figure 11–1 Edit Admin Role Pop Up Screen



10. Click the External Role Mapping tab to grant the Administrator Role to members of External Roles.
11. Click Add to display the Search Principals dialog.
12. Complete the query fields in the External Roles search box and click Search.

    Empty strings fetch all roles. The results display in the Search Results table.

13. Select the external role to map to by clicking its name in the table.

    Use Ctrl+click to select multiple roles.

14. Click Add Principals.

    The selected roles display in the External Role Mapping tab.

15. Click the External User Mapping tab to grant the Administrator Role to External Users.
16. Click Add to display the Search Principals dialog.

17. Complete the query fields in the Users search box and click Search.

    Empty strings fetch all roles. The results display in the Search Results table.

18. Select the user to map by selecting its name in the table.

    Use Ctrl+click to select multiple roles.

19. Click Add Principals.

    The selected roles display in the External User Mapping tab.

11.3.2 Modifying or Deleting an Application's Delegated Administrator

To modify or delete an Application's configured Administrator Role, proceed as follows.

1. Expand the Applications node in the Navigation Panel.
2. Select the Application to modify.
3. Right-click the Application name and select Open from the menu.

   The General, Delegated Administrators, Policy Distribution and Simulation tabs are all active.

4. Click the Delegated Administrators tab.
5. Navigate to the Administrator Role you want to modify and select it.

   The Role Details, External Role Mapping and External User Mapping tabs are displayed.

6. Select the tab which contains the configuration to modify or delete.
   - To modify the configuration, see Section 11.3.1, "Adding a Delegated Administrator for An Application" for details.
   - To remove a mapping from an Administrator Role, select the applicable Administrator Role and the appropriate Mapping tab. Select the mapping and click Remove.
   - To delete an Administrator Role, select the Administrator Role and click Delete.

11.4 Using Policy Domains to Delegate

Administration of the policies securing one protected application may be delegated using one or more (optional) Policy Domains. A Policy Domain contains the components of completed policy definitions. It is the amalgamation of a target Resource (an instance of the Resource Type), an Entitlement (the actions that can be performed on the Resource), and a Policy (a rule that assembles the controls and the principals they affect).

The use of multiple Policy Domains allows policies to be partitioned according to some defined logic, such as the architecture of the protected application or how administration of the policies are delegated. For example, one Policy Domain can be used to maintain all policies securing a Resource or multiple Policy Domains can be used to reflect a particular characteristic of the Resource. Different administrators can then be placed in charge of different Policy Domains.

Note: Because the creation of a Policy Domain is optional, if there is no need to delegate policy administration, there is no need to create any Policy Domains. In this case, a default Policy Domain is created with each Application that will contain all the Application's policy objects.

The following sections contain the management procedures for Policy Domains.

- Section 11.4.1, "Creating a Policy Domain"
- Section 11.4.2, "Modifying a Policy Domain"
- Section 11.4.3, "Deleting a Policy Domain"

11.4.1 Creating a Policy Domain

To create a Policy Domain, proceed as follows.

1. Right-click the name of the Application in the Navigation Panel under which the Policy Domain will be created and select New from the menu.

   An Untitled page displays in the Home area.

2. Provide the following information for the Policy Domain.
   - **Display Name**
   - **Name**
   - **Description**: Although optional, it is recommended to provide useful information about the entitlement.
3. Select one of the following from the Save menu.
   - Save and Close saves the configuration and renames the tab with the value provided for the Policy Domain's Display Name.
   - Save and Create Another saves the configuration to the information tree in the Navigation Panel but leaves the Untitled area open for you to create another Application.

11.4.2 Modifying a Policy Domain

To modify a Policy Domain, proceed as follows.

1. Navigate to the Application under which the Policy Domain you want to delete was created and expand the information tree.
2. Double click the name of the Policy Domain you want to modify.

   The Policy Domain configuration displays in the Home area.
3. Modify as necessary and click Apply.

11.4.3 Deleting a Policy Domain

To delete a Policy Domain, proceed as follows.

1. Navigate to the Application under which the Policy Domain you want to delete was created and expand the information tree.
2. Double click the name of the Policy Domain you want to delete.

The Policy Domain configuration displays in the Home area.

3. Click Delete.

   A confirmation dialog is displayed.

4. Click OK to delete.

11.5 Delegating Policy Domain Administration

The following sections describe how to manage administrators for Policy Domains.

- Section 11.5.1, "Adding a Delegated Administrator to a Policy Domain"
- Section 11.5.2, "Modifying or Deleting a Policy Domain's Delegated Administrator"

11.5.1 Adding a Delegated Administrator to a Policy Domain

This procedure documents how to create a new Administrator Role and assign it to the applicable roles or users. To add a delegated administrator to a Policy Domain, proceed as follows.

1. Expand the Applications node in the Navigation Panel.
2. Select the Application to modify.
3. Right-click the Application name and select Open from the menu.

   The General, Delegated Administrators, Policy Distribution and Simulation tabs are all active.

4. Click the Delegated Administrators tab.

   The Policy Domain names are listed in the displayed table. Clicking the arrow next to the Policy Domain expands the hierarchy and displays any Administrator Roles already configured; for example, the default `PolicyDomainAdmin`.

5. Select the Policy Domain under which you will create the Administrator Role.
6. Click New to create a new Administrator Role.

   Be sure to select the name of the Policy Domain to activate New. Alternately, select the Policy Domain and select New from the Actions menu. A New Administrator Role dialog is displayed.

7. Provide the following values for the new Administrator Role and click OK.
   - **Name**: The entry must be a unique.
   - **Display Name**
   - **Description**
8. Select the new Administrator Role to activate its configuration tabs.

   The Role Details tab is active.

9. Click Edit to define the role details.

   An Edit Administrator Role dialog is displayed.

10. Grant View or Manage privileges for the appropriate Policy Domain objects and click Save.
11. Click the External Role Mapping tab.

a. Click Add to display the Search Principals dialog.

b. Complete the query fields in the External Roles search box and click Search.

   Empty strings fetch all roles. The results display in the Search Results table.

c. Select the external role to map to by clicking its name in the table.

   Use Ctrl+click to select multiple roles.

d. Click Add Principals.

   The selected roles display in the External Role Mapping tab.

12. Click the External User Mapping tab.

a. Click Add to display the Search Principals dialog.

b. Complete the query fields in the Users search box and click Search.

   Empty strings fetch all roles. The results display in the Search Results table.

c. Select the user to map by selecting its name in the table.

   Use Ctrl+click to select multiple roles.

d. Click Add Principals.

   The selected roles display in the External User Mapping tab.

11.5.2 Modifying or Deleting a Policy Domain's Delegated Administrator

To modify or delete a Policy Domain's configured Administrator Role, proceed as follows.

1. Expand the Applications node in the Navigation Panel.
2. Select the Application to modify.
3. Right-click the Application name and select Open from the menu.

   The General, Delegated Administrators, Policy Distribution and Simulation tabs are all active.

4. Click the Delegated Administrators tab.
5. Navigate to the Administrator Role you want to modify and select it.

   The Role Details, External Role Mapping and External User Mapping tabs are displayed.

6. Select the tab which contains the configuration to modify or delete.

   - To modify the configuration, see Section 11.5.1, "Adding a Delegated Administrator to a Policy Domain" for details.
   - To remove a mapping from an Administrator Role, select the applicable Administrator Role and the appropriate Mapping tab. Select the mapping and click Remove.
   - To delete an Administrator Role, select the Administrator Role and click Delete.

11.6 Managing System Administrators Using Administrator Roles

You can delegate system administration privileges to users by creating and configuring System Administrator Roles. By default, `SystemAdmin` is created during

installation and is displayed in the System Administrators table when you navigate to System Administrators under the main System Configuration tab. `SystemAdmin` manages system-level resources (including other Administrator Roles, and system configurations and bindings) and maps to the WebLogic Server `weblogic` user.

The following sections document the management operations for all Oracle Entitlements Server System Administrator Roles.

- Section 11.6.1, "Creating a New Administrator Role"
- Section 11.6.2, "Assigning Privileges to an Administrator Role"
- Section 11.6.3, "Modifying Administrator Role Membership"
- Section 11.6.4, "Deleting an Administrator Role"

11.6.1 Creating a New Administrator Role

To create a new Administrator Role, proceed as follows.

1. Select the System Configuration tab from the Home area.

   The System Administrators tab is displayed in the Home area.

2. Click New under Administrator Roles to create a new Administrator Role.

   A dialog is displayed.

3. Provide the following values for the new Administrator Role.
   - **Name**: The entry must be a unique.
   - **Display Name**
   - **Description**
4. Click Create.

11.6.2 Assigning Privileges to an Administrator Role

To assign privileges to an Administrator Role, map external roles, external users or both to the role as documented in this procedure.

1. Select the System Configuration tab from the Home area.

   The System Administrators tab and configured Administrator Roles are displayed in the Home area. Alternately, right-click Administrators and select Open.

2. Select the name of the Administrator Role from the table.
3. Select the Modify or View option to define the Administrator Control.

   Modify defines the administrator as having management (and by proxy viewing) privileges on all system administrator resources. View defines the administrator as having only viewing privileges.

4. Click the External Role Mapping tab.
   a. Click Add or select Add from the Actions menu.

      The Add Roles search dialog is displayed.

   b. Enter a search string in the text box and click the arrow to search for External Roles.

      Alternately, click Search with no search string to return all available External Roles.

c. Select one or more roles from the results and click Add Selected.

   Alternately, click Add All to add all returned results.

d. Click Add Principals.

5. Click the External User Mapping tab.

   a. Click Add or select Add from the Actions menu.

      The Add Users search dialog is displayed.

   b. Enter a search string in the text box and click the arrow to search for External Users.

      Alternately, click Search with no search string to return all available External Users.

   c. Select one or more users from the results and click Add Selected.

      Alternately, click Add All to add all returned results.

   d. Click Add Principals.

11.6.3 Modifying Administrator Role Membership

To modify Administrator Role membership, delete the mappings as documented in this procedure.

1. Select the System Configuration tab from the Home area.
2. Double-click System Administrators in the Navigation Panel.

   Alternately, right-click System Administrators and select Open. The System Administrators page is displayed.

3. Select the name of the Administrator Role from the table.
4. Modify the Modify or View Administrator Control as necessary.
5. Click the External Role Mapping tab.

   a. Select the External Role to delete.

   b. Click Remove.

      Alternately, select Remove from the Actions menu.

6. Click the External User Mapping tab.

   a. Select the External User to delete.

   b. Click Remove.

      Alternately, select Remove from the Actions menu.

11.6.4 Deleting an Administrator Role

To delete an Administrator Role, proceed as follows.

1. Select the System Configuration tab from the Home area.
2. Double-click System Administrators in the Navigation Panel.

   Alternately, right-click System Administrators and select Open. The System Administrators page is displayed.

3. Select the name of the Administrator Role from the table.

4. Click Delete.

   A confirmation dialog is displayed.

5. Click Remove.

12

Customizing the Administration Console

This chapter explains several customizations you can make to Oracle Authorization Policy Manager, the Oracle Entitlements Server Administration Console. It contains the following sections:

12.1 Customizing Authorization Policy Manager

All customizations described in this chapter require modifying data in one or both of the following file archives:

```
$ORACLE_IDM_HOME$/apm/modules/oracle.security.apm_11.1.1/oracle.security.apm.ear
$ORACLE_IDM_HOME$/apm/modules/oracle.security.apm_
   11.1.1/oracle.security.apm.core.view.war
```

> **Tip:** Before you begin, it is recommended that you backup these Authorization Policy Manager EAR and WAR files.

Any customizations applied to a version of Authorization Policy Manager must be specified again every time a new version is installed. The following procedure specifies, from a high level, how to customize Authorization Policy Manager.

1. Unzip the EAR, WAR and view WAR files using the following commands:

```
$ unzip -d $tempDir/ear $ORACLE_IDM_HOME$/apm/modules/oracle.security.apm_
   11.1.1/oracle.security.apm.ear
$ unzip -d $tempDir/war $tempDir/ear/oracle.security.apm.war
$ unzip -d $tempDir/viewWar $ORACLE_IDM_HOME$/apm/modules/
   oracle.security.apm_11.1.1/oracle.security.apm.core.view.war
```

2. Modify one or more of the unzipped files as documented in one of the following sections of this chapter.
   - Section 12.2, "Customizing Headers, Footers, and Logo"
   - Section 12.3, "Customizing Color Schemes"
   - Section 12.4, "Customizing the Login Page"
3. Rearchive the modified EAR, WAR and view WAR files using the following commands:

```
$ zip $tempDir/ear/oracle.security.apm.war $tempDir/war/*
$ zip $ORACLE_IDM_HOME$/apm/modules/oracle.security.apm_
   11.1.1/oracle.security.apm.ear $tempDir/ear/*
$ zip $ORACLE_IDM_HOME$/apm/modules/oracle.security.apm_
   11.1.1/oracle.security.apm.core.view.war $temp/viewWar/*
```

4. Redeploy Authorization Policy Manager.

12.2 Customizing Headers, Footers, and Logo

Use the following procedure to customize headers, footers, and the logo.

1. Unzip the view WAR file.

```
$ORACLE_IDM_HOME$/apm/modules/oracle.security.apm_
   11.1.1/oracle.security.apm.core.view.war
```

2. Open `AuthPolicyMgr.jspx` file and apply any or all of the following modifications.

   - Specify a new branding title (header) by modifying the `branding` facet.

```
<f:facet name="branding">
 <af:outputText value="My Custom Application Title" noWrap="true"
id="ot1"/>
</f:facet>
```

   - Specify a new footer by modifying the `appAbout` and `appCopyright` facets.

```
<f:facet name="appAbout">
<af:outputText value="My Custom Footer at Right" noWrap="true" id="ot2"/>
</f:facet>
<f:facet name="appCopyright">
<af:outputText value="My Custom Footer at Left"  noWrap="true" id="ot3"/>
</f:facet>
```

   - Specify a new logo image as follows:

     a. Insert your resource in the `metaContainer` facet.

```
<f:facet name="metaContainer">
....
<af:resource type="css">
.MyCustomBrandingLogo {
background-image:url(/apm/images/world_36x20.png);
 background-position:center;
 background-repeat:no-repeat; display:block;
height:2.5em; width:119px;
          }
 </af:resource>
...
</f:facet>
```

     Be sure to leave all other content inside the `metaContainer` facet as is.

     b. Specify the style class name (defined in the previous step) as the attribute value of the `pageTemplate` tag.

```
<af:pageTemplate viewId="/templates/IdmShell.jspx"
 value="#{bindings.pageTemplateBinding}" id="pt1">
...
<f:attribute name="brandingLogoCls" value="MyCustomBrandingLogo"/>
...
```

Be sure to leave all other content inside the `pageTemplate` tag as is.

3. Rearchive the view WAR file.

```
$ zip $ORACLE_IDM_HOME$/apm/modules/oracle.security.apm_
   11.1.1/oracle.security.apm.core.view.war $temp/viewWar/*
```

4. Redeploy Authorization Policy Manager.

12.3 Customizing Color Schemes

You can develop a new skin to apply to a web application. Use the following procedure to customize the Authorization Policy Manager color scheme. It assumes that you have a new skin available to reference.

Note: Authorization Policy Manager uses the Oracle Application Development Framework (ADF) and supports ADF skinning. See the *Oracle Fusion Middleware Skin Editor User's Guide for Oracle Application Development Framework* for more information on ADF skins.

---

1. Unzip the EAR and WAR files.

```
$ unzip -d $tempDir/ear $ORACLE_IDM_HOME$/apm/modules/oracle.security.apm_
   11.1.1/oracle.security.apm.ear
$ unzip -d $tempDir/war $tempDir/ear/oracle.security.apm.war
```

2. Open the `Trinidad-config.xml` file.

   This file is typically located in the decompressed WAR's `WEB-INF` folder.

3. Specify the value of the new skin location in the `skin-family` tag.

```
<trinidad-config xmlns="http://myfaces.apache.org/trinidad/config">
...
<skin-family>MyCustomSkin</skin-family>
...
</trinidad-config>
```

4. Rearchive the modified EAR and WAR files using the following commands:

```
$ zip $tempDir/ear/oracle.security.apm.war $tempDir/war/*
$ zip $ORACLE_IDM_HOME$/apm/modules/oracle.security.apm_
   11.1.1/oracle.security.apm.ear $tempDir/ear/*
```

5. Redeploy Authorization Policy Manager.

12.4 Customizing the Login Page

Use the following procedure to customize the login and login error pages.

1. Unzip the EAR file.

```
$ unzip -d $tempDir/ear $ORACLE_IDM_HOME$/apm/modules/oracle.security.apm_
   11.1.1/oracle.security.apm.ear
```

2. Open the `web.xml` file.

   This file is typically located in the decompressed EAR's `WEB-INF` folder.

3. Specify the appropriate values for the `form-login-page` and `form-error-page` under the element `form-login-config`.

```
<login-config>
 <form-login-config>
  <form-login-page>/MyCustomLoginPage.html</form-login-page>
  <form-error-page> MyCustomLoginErrorPage.html </form-error-page>
 </form-login-config>
</login-config>
```

4. Rearchive the modified EAR file using the following commands:

```
$ zip $ORACLE_IDM_HOME$/apm/modules/oracle.security.apm_
   11.1.1/oracle.security.apm.ear $tempDir/ear/*
```

5. Redeploy Authorization Policy Manager.

13 Management Tasks

This chapter contains information on several management and configuration tasks including configuring Security Modules and the cache, auditing, and migrating policies from different types of policy stores. It contains the following sections.

13.1 Moving from a Test Environment to Production (T2P)

Test to Production (T2P) is the process of migrating the Oracle Entitlements Server Administration Console from a test environment to a production environment. The T2P process is only applicable to the console component (an application runnng in a WebLogic Server domain) as Security Modules do not maintain their own data. The following procedure documents the T2P process.

1. Move the Database, Middleware Homes, and Domain Configuration to the new production environment as documented in *Oracle Fusion Middleware Administrator's Guide*.
2. Install Oracle Entitlements Server in the new production environment.
3. Move Oracle Platform Security Services to the new production environment as documented in *Oracle Fusion Middleware Administrator's Guide*.

13.2 Using the Policy Simulator

Oracle Entitlements Server provides a feature to simulate a policy from the Administration Console. Policy simulation allows you to troubleshoot, test, analyze the effect of policy changes, and understand how policies on a given application are enforced. The following sections contain more information.

- Section 13.2.3, "Running the Policy Simulator"

13.2.1 Understanding Policy Simulation

In the simplest policy simulation case, a user and resource-action pair are specified and the following information is retrieved, based on the parameters:

- External Roles that would be granted to the principal, and a list of static role grants and Role Mapping Policies.
- List of attributes required to evaluate any relevant role or authorization policy.
- Value of any evaluation functions that were executed as part of a relevant role or authorization policy.
- Value of any dynamic attributes that is part of a relevant role, authorization policy, or obligation. If a dynamic attribute value is unknown, it should be returned as blank so the user can explicitly set it.
- The final authorization decision returned (*grant* or *deny*), obligations, and the Authorization Policies that were involved.

If attribute values are evaluated as part of the authorization decision, you can perform additional simulations using different values. The sequence of events is:

1. Specify a principal, resource and action.

   The Principal can be a user, a set of External Roles or a set of Application Roles. If External Roles and Application Roles are explicitly specified with the user, role resolution will not be performed.

2. Simulate the policy.
3. Specify which attribute to override and provide values for the ones that they want to override.
4. Re-run the Policy Simulator to see any changed results.

13.2.2 Choosing the Policy Simulation Mode

The Policy Simulator has two modes: Simple and Advanced. The modes have different purposes. Simple mode is typically used for debugging purposes to determine the authorization result for a given Subject and Resource combination. It allows security administrators to understand why a particular user has (or does not have) access to the specified Resource. Advanced mode is for policy simulation - to see what happens, for example, if a user is made a member of a particular set of External Roles and Application Roles. Based on the simulation, security administrators can add or revoke a user's role membership based on the runtime authorization decisions for the specified Resource. Defining values in Simple Mode and clicking Check Access will:

- Display the results of the policy evaluation including the authorization decision, obligations (if any), and a list of attributes required to evaluate any relevant role or policy. If an attribute value cannot be retrieved, it will be returned as blank.
- Display the set of External Roles to which the specified user belongs. This list facilitates selecting a user's External Roles for running the Policy Simulator in Advanced mode.
- Display the Application Roles granted to (or denied) the specified user and the Role Mapping Policies in which the user is specified. The roles may include static and dynamic roles granted by Role Mapping Policies directly or indirectly (through inheritance).

- Display the Authorization Policies used for evaluating access.

Defining values in Advanced Mode involves providing 0 or 1 user, a set of External Roles, a set of Application Roles or any combination of the three as Principal. The results displayed are as in Simple Mode except if External Roles and Application Roles are specified; in this case, group resolution will not be computed.

13.2.3 Running the Policy Simulator

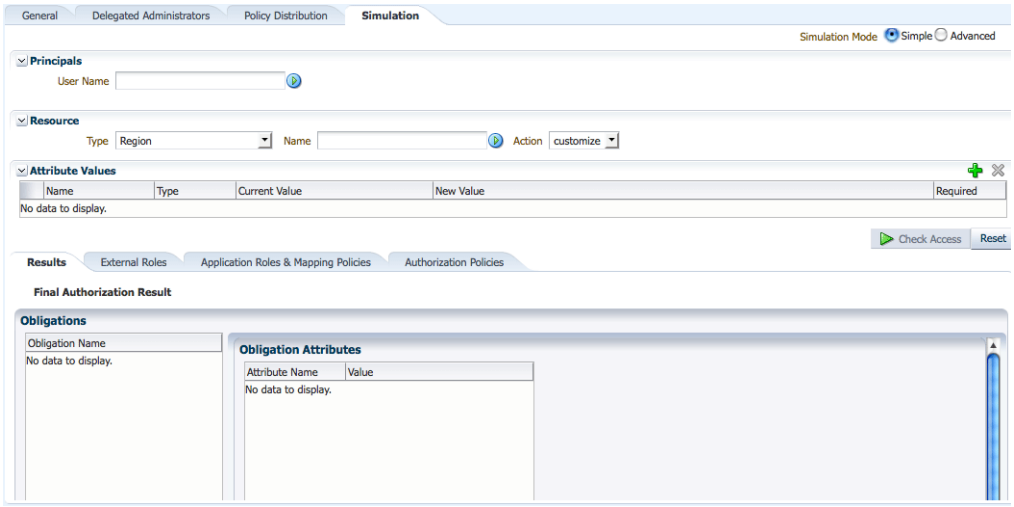
The Policy Simulator is displayed as a tab when a configured Application is open for the Administration Console. The tab is only enabled after an Application object has been created and only for the administrator with modify privileges for the Application. The following sections document how to run the Policy Simulator.

- Section 13.2.3.1, "Running the Policy Simulator in Simple Mode"
- Section 13.2.3.2, "Running the Policy Simulator in Advanced Mode"

13.2.3.1 Running the Policy Simulator in Simple Mode

Figure 13–1 is a screenshot of the Policy Simulator in Simple Mode. Use the procedure in this section to run the Policy Simulator in Simple Mode.

Figure 13–1 Policy Simulator User Interface in Simple Mode



1. Select the Application name that contains the policies and click Open.
2. Click the Simulation tab.
3. Click the Simple radio button.

   A Simple run is used when defining one user as a Principal.
4. Search for and select one Principal.
   a. Enter Search critieria in the User Name box and click the Search arrow.

      The User Search results are displayed.
   b. Select the appropriate row in the User Search and click Add to select the user as a Principal.
5. Search for and select a Resource.

a. Choose the Resource Type (of which the Resource is an instance) from the drop down menu.

b. Choose the appropriate Action Name from the drop down menu.

c. Enter search criteria for the Resource instance name and click the search arrow.

   The Resources are displayed. Optionally, further refine the search results by selecting a Policy Domain, entering additional search criteria and clicking the Search arrow.

6. Add an attribute to the Policy Simulation test.

   Attributes are used in Conditions. To evaluate a policy, values are needed for the attributes. If the attribute has a default value, it will be used. (The default value can be overridden here as well.) If no default value, the Simulator will prompt for one.

   a. Click the Green Plus to add an attribute.

      Use the search pop-up to select a defined attribute and specify the value based on its data type.

   b. Enter a Name.

   c. Enter a New Value.

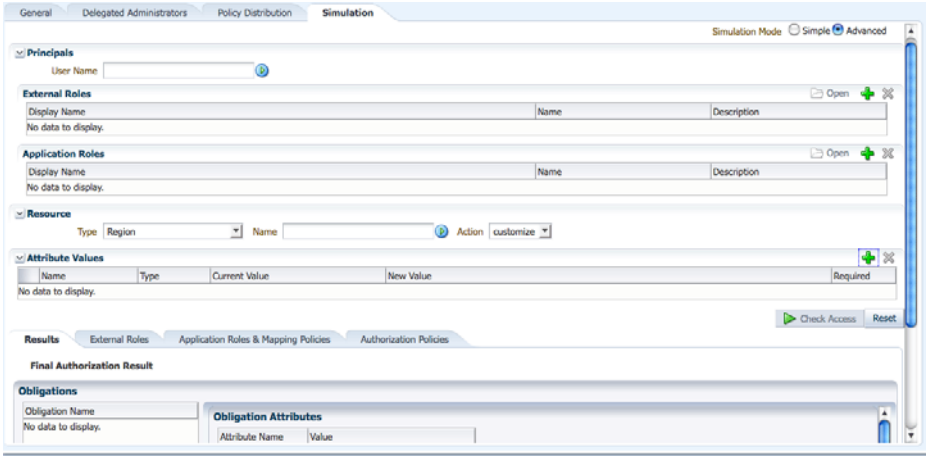
7. Click Check Access.

   The Policy Simulator executes. See Section 13.2.2, "Choosing the Policy Simulation Mode" for details on the results.

13.2.3.2 Running the Policy Simulator in Advanced Mode

Figure 13–2 is a screenshot of the Policy Simulator in Advanced Mode. In Advanced Mode, you can define only one user, as well as Application Roles or External Roles. Use the procedure in this section to run the Policy Simulator in Advanced Mode.

Figure 13–2 Policy Simulator User Interface in Advanced Mode



1. Select the Application name that contains the policies and click Open.
2. Click the Simulation tab.
3. Click the Advanced radio button.

4. Search for and select a User Name, an External Role or an Application Role as the Principal.

   At least one of a User Name, External Role or Application Role must de defined.

   a. Enter Search critieria in the User Name box and click the Search arrow.

      The User Search results are displayed.

   b. Select the appropriate row in the User Search and click Add to select the user as a Principal.

   c. Click the Green Plus to add an External Role.

      The External Roles Search dialog is displayed.

   d. Enter Search critieria and click the Search.

      The Search Results are displayed.

   e. Select the appropriate row(s) in the Search Results and click Add Selected or click Add All to add all results.

      The roles are added. In Advanced Mode, the Policy Simulator will not compute the user's External Roles. If that computation is needed, you must specifically add them.

   f. Click the Green Plus to add an Application Role.

      The Application Roles Search dialog is displayed.

   g. Enter Search critieria and click the Search.

      The Search Results are displayed.

   h. Select the appropriate row(s) in the Search Results and click Add Selected or click Add All to add all results.

      The roles are added. In Advanced Mode, the Policy Simulator will not compute the user's Application Roles. If that computation is needed, you must specifically add them.

5. Search for and select a Resource.

   a. Choose the Resource Type (of which the Resource is an instance) from the drop down menu.

   b. Choose the appropriate Action Name from the drop down menu.

   c. Enter search criteria for the Resource instance name and click the search arrow.

      The results are displayed. Optionally, further refine the search results in the Resource Finder by selecting a Policy Domain, entering additional search criteria and clicking the Search arrow.

6. Add an attribute to the Policy Simulation test.

   Attributes are used in Conditions. To evaluate a policy, values are needed for the attributes. If the attribute has a default value, it will be used. (The default value can be overridded here as well.) If no default value, the Simulator will prompt for one.

   a. Click the Green Plus to add an attribute.

      Use the search pop-up to select a defined attribute and specify the value based on its data type.

b. Enter a Name.

c. Enter a New Value.

7. Click Check Access.

   The Policy Simulator executes. See Section 13.2.2, "Choosing the Policy Simulation Mode" for details on the results.

13.3 Using FIPS-compliant Security Providers

The Federal Information Processing Standard (FIPS) is developed by the United States government to standardize security requirements for cryptography modules. Support for the FIPS allows integration of Oracle Entitlements Server with disparate Java Cryptography Extension (JCE) security providers.

Note: Security services were limited to the container's default security providers in previous releases.

So this document is to figure out how to install one specific JCE provider in OES 11g and configure different JCE provider for related OES 11g subcomponents. Since security services used in enrollment tool and SSL connection setup are SSL related, in most cases they are handled by the container and are out of the scope of this project. The only OES specific subcomponent we need to configure is the local cache file encryption.

13.3.1 Installing the JCE Provider

Before configuring the JCE provider, it must be installed and registered. Detailed information on how to do this can be found in the *Java Cryptography Extension API Specification and Reference* at
`http://docs.oracle.com/javase/1.4.2/docs/guide/security/CryptoSpec.html#ProviderInstalling`

13.3.2 Configuring JCE

The following `jps-config.xml` parameters are used for configuring the JCE security provider. Back up the policy cache files before modifying these JCE provider parameters.

- `oracle.security.jps.runtime.pd.client.localpolicy.JCEProviderName` indicates the JCE provider being used. It is optional and if not specified, the default JDK provider will be used. No default value is defined and any value used is case-sensitive.
- `oracle.security.jps.runtime.pd.client.localpolicy.CipherKeyLength` indicates the key length used for the Cipher class from the specified JCE provider. It is optional and the default value is 128. Only three key lengths (128, 192 and 256) are supported.
- `oracle.security.jps.runtime.pd.client.localpolicy.CipherModePadding` indicates the cipher algorithm name, mode and padding schema used for the Cipher class from the specified JCE provider. The format should be *algorithm-name/mode/padding*. It is optional, not case-sensitive and the default value is `AES/CBC/PKCS5Padding`.

13.4 Managing Audit Tasks

Oracle Entitlements Server audits all administrative activities and authorization requests, optionally recording the information to a file. The auditing framework is based on the framework developed for Oracle Platform Security Services. An overview of the Oracle Platform Security Services auditing framework can be found in *Oracle Fusion Middleware Application Security Guide*. The following information is specific to the auditing functionality in Oracle Entitlements Server.

- Section 13.4.1, "Auditing Oracle Entitlements Server Events"
- Section 13.4.2, "Configuring Oracle Entitlements Server Administration Server for Auditing"
- Section 13.4.3, "Configuring Oracle Entitlements Server Security Modules for Auditing"
- Section 13.4.4, "Additional Auditing Information"

> **Note:** Oracle Entitlements Server will audit decisions resulting from policies configured by itself, Oracle Platform Security Services or any combination thereof.

13.4.1 Auditing Oracle Entitlements Server Events

Table 13–1 lists the events (organized by functional category) that are audited by Oracle Entitlements Server.

Table 13–1 Events Audited in Oracle Entitlements Server

Functional Category	Functional Task
Administration Role Management	■ AdminRoleCreation ■ AdminRoleDeletion ■ AdminRoleGrant ■ AdminRoleRevoke ■ AdminRoleResActionGrant ■ AdminRoleResActionRevoke
Application Management	■ ApplicationDeletion
Grant Management	■ PermissionSetGrant ■ PermissionSetRevocation
PermissionSetManagement	■ PermissionSetCreation ■ PermissionSetModification ■ PermissionSetDeletion
PolicyDomainManagement	■ PolicyDomainCreation ■ PolicyDomainDeletion
PolicyManagement	■ PolicyCreation ■ PolicyModification ■ PolicyDeletion ■ PolicyGrant ■ PolicyRevoke

Table 13–1 (Cont.) Events Audited in Oracle Entitlements Server

Functional Category	Functional Task
ResourceManagement	■ ResourceCreation ■ ResourceModification ■ ResourceDeletion
Role Management	■ RoleCreation ■ RoleModification ■ RoleDeletion ■ RoleMembershipAdd ■ RoleMembershipRemove
RolePolicyManagement	■ RolePolicyCreation ■ RolePolicyModification ■ RolePolicyDeletion
Authorization	■ CheckPermission ■ IsAccessAllowed ■ CheckSubject
ConfigurationBindingManagement	■ SecurityModuleBinding ■ SecurityModuleUnbinding
ConfigurationManagement	■ SecurityModuleCreation ■ SecurityModuleModification ■ SecurityModuleDeletion
PolicyDistributionManagement	■ PolicyDistribution ■ PdpDeregistration ■ purgeDistributionStatus

13.4.2 Configuring Oracle Entitlements Server Administration Server for Auditing

Audit logging is disabled by default. Auditing is configured in `jps-config.xml`, the configuration file used by Java EE containers that is located in the *$DOMAIN_HOME*`/config/fmwconfig` directory. Example 13–1 illustrates how auditing might be configured in jps-config.xml.

Example 13–1 Audit Service Configuration Parameters in jps-config.xml

```
<!-- Audit Service Instance-->
<serviceInstance name="audit" provider="audit.provider"
    location="./audit-store.xml">
  <description>Audit Service</description>
  <property name="audit.filterPreset" value="None"/>
  <property name="audit.maxDirSize" value="0"/>
  <property name="audit.maxFileSize" value="104857600"/>
  <property name="audit.loader.jndi" value="jdbc/AuditDB"/>
  <property name="audit.loader.interval" value="15"/>
  <property name="audit.loader.repositoryType" value="File"/>
  <property name="auditstore.type" value="file"/>
</serviceInstance>
```

Restart the Oracle Entitlements Server Administration Server after making any modifications to the `jps-config.xml` file. Table 13–2 contains details about the configuration parameters.

Table 13–2 Auditing Parameters in jps-config.xml

Parameter	Description
audit.filterPreset	Enable auditing by choosing from the following values: None (default), Low, Medium, All or Custom
audit.maxDirSize	Controls the size of the directory in which the audit files are written. Takes an integer in bytes.
audit.maxFileSize	Controls the size of the bus stop file in which audit events are written. Takes an integer in bytes.
audit.loader.jndi	When a database is in use, takes a path to the JNDI data source to which audit events are uploaded.
audit.loader.interval	When a database is in use, controls the frequency of the audit loader's upload. Takes an integer in seconds.
audit.loader.RepositoryType	Defines the audit repository type. Takes a value of **File** or **Db**. ■ Use `Db` if loading audit records from a file to a data base. If type is database (Db), `audit.loader.jndi` must also be defined. ■ Use `File` if loading audit records from a data base to a file.
auditstore.type	Takes as a value `file` or `db` depending on the audit store type.

13.4.3 Configuring Oracle Entitlements Server Security Modules for Auditing

To configure auditing for Oracle Entitlements Server Security Modules, first apply the Oracle Entitlements Server Bundle Patch 1 (BP1) and then create the audit database.

> **Note:** See the *Oracle Fusion Middleware Release Notes for Microsoft Windows x64 (64-Bit)* for information on completing Oracle Entitlements Server audit schema definitions. The procedure can be done before or after creating the audit database.

After completing these steps, use one of the following procedures depending on the deployed Security Module.

- Section 13.4.3.1, "Configuring the WebLogic Server Security Module"
- Section 13.4.3.2, "Configuring Other Security Modules"

13.4.3.1 Configuring the WebLogic Server Security Module

1. Create and configure the data source to the audit database, and deploy the data source to the server using the WebLogic Server console.

   See *Oracle Fusion Middleware Application Security Guide* for details.

2. Restart the server.

3. Add the following content before the final `</serviceInstances>` tag in the `jps-config.xml` file.

   In a non JRF domain, jps-config.xml is located in the `WLSSM_Client_Domain\config\oeswlssmconfig\AdminServer\` directory. In a JRF domain, jps-config.xml is located in the *`WLSSM_Client_Domain`*`\config\fmwconfig\` directory.

```
<serviceInstance name="audit" provider="audit.provider">
  <property name="audit.filterPreset" value="All"/>
```

```
  <property name="audit.maxDirSize" value ="0"/>
  <property name="audit.maxFileSize" value ="104857600"/>
  <property name="audit.loader.jndi" value="auditdb"/>
  <property name="audit.loader.interval" value="15" />
  <property name="audit.loader.repositoryType" value="Db" />
</serviceInstance>
```

4. Restart the servers.

   `audit.log` is created under the WLSSM_Client_Domain/`servers/AdminServer/logs/auditlogs/JPS/` directory.

5. Run the database `set password` command to set a password for the audit database.

   When prompted, enter the password for your audit schema. See *Oracle Fusion Middleware Application Security Guide* for details.

6. Run the audit loader to load the events and actions from the `audit.log` file to the audit database.

   A message that confirms the number of events transferred is displayed after a successful action. See *Oracle Fusion Middleware Application Security Guide* for details.

13.4.3.2 Configuring Other Security Modules

1. Add the following content before the final `</serviceInstances>` tag in the `jps-config.xml` file (located in the *`OES_Client11115`*`\oes_sm_instances\SM_Name\config\` directory) to set up the audit database loader.

   ```
   <serviceInstance name="audit" provider="audit.provider"
       location="./audit-store.xml">
     <description>Audit Service</description>
   <property name="audit.loader.jndi" value="jdbc/AuditDB"/>
   <property name="audit.loader.interval" value="15"/>
   <property name="audit.loader.repositoryType" value="Db"/>
   ```

   where:

   - The value of `audit.loader.repositoryType` must be `Db`.
   - The database schema for the audit loader should be created using the Repository Creation Utility (RCU).
   - The datasource being connected to the database should be configured as the value of `audit.loader.jndi`.

2. Run the database `set password` command to set a password for the audit database.

   When prompted, enter the password for your audit schema. See *Oracle Fusion Middleware Application Security Guide* for details.

3. Run the audit loader to load the events and actions from the `audit.log` file to the audit database.

   A message that confirms the number of events transferred is displayed after a successful action. Be sure to define the `-Doracle.instance` parameter with the path to the Security Module instance and database details. See *Oracle Fusion Middleware Application Security Guide* for details.

13.4.4 Additional Auditing Information

The following list collects chapter links in other documents with information regarding the auditing framework.

- Introductory material can be found in *Oracle Fusion Middleware Security Guide*.
- You can manage audit policies with the Enterprise Manager user interface or with the WebLogic Scripting Tool (WLST) command-line interface. See *Oracle Fusion Middleware Application Security Guide* for guidance.
- The Oracle Fusion Middleware Audit Framework Reference is in *Oracle Fusion Middleware Security Guide*.
- Additional configuration information is in *Oracle Fusion Middleware Security Guide*.

13.5 Migrating Policies

This section contains information regarding migrating policies from one type of store to another. It contains procedures for the following:

- Section 13.5.1, "Migrating From XML to LDAP"
- Section 13.5.2, "Migrating From LDAP to XML"
- Section 13.5.3, "Migrating From XML to Database"
- Section 13.5.4, "Migrating From Database to XML"

13.5.1 Migrating From XML to LDAP

Following is the procedure to migrate policies from an XML-based policy store to an LDAP-based directory.

1. Modify `jps-config.xml` as described in this sub procedure.
   a. Create a `serviceInstance` for both the source and destination policy stores as illustrated in Example 13–2.

      The location of the source policy store instance must be defined as relative to the location of `jps-config.xml` defined as the value of `configFile` when using the `migrateSecurityStore` command. Example 13–2 assumes that `jps-config.xml` and `jazn-data.xml` (the source policy store) are located in the same directory.

Example 13–2 XML to LDAP serviceInstances for Source and Destination Policy Stores

```
<!-- Source XML-based policy store instance -->
   <serviceInstance name="src.xml" provider="policystore.xml.provider"
      location="./jazn-data.xml">
    <description>File Based Policy Store Service Instance</description>
   </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. 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>
```

b. Create a `serviceInstance` corresponding to the bootstrap credential used to access the destination LDAP directory as illustrated in Example 13–3.

Example 13–3 XML to LDAP serviceInstance for Bootstrap Credential

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

c. Create a `jpsContext` for both source and destination stores as illustrated in Example 13–4.

Example 13–4 XML to LDAP jpsContext for Source and Destination Policy Stores

```
<jpsContext name="sourceContext">
   <serviceInstanceRef ref="src.xml"/>
</jpsContext>

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

<jpsContext name="bootstrap_credstore_context">
   <serviceInstanceRef ref="bootstrap.cred"/>
</jpsContext>
```

2. Start the WebLogic Scripting Tool.

   There is no need to connect the WebLogic Scripting Tool to the WebLogic Server as the migration command is an offline command.

3. Run the WebLogic Scripting Tool `migrateSecurityStore` command to migrate the policy store and application as follows.

   - To migrate the policy store, run:

     ```
     migrateSecurityStore
        (type="policyStore", src="sourceContext",
         dst="destinationContext",
         configFile="myDir/jps-config.xml")
     ```

     where the following applies:

     – The name of the corresponding `jpsContext` (previously created in Step 1) should be passed to the `src` and `dst` parameters.

     – The name of the jps-config.xml file (previously modified in Step 1) should be passed to the `configFile` parameter. The value must be a fully qualified file name with complete path information.

   - To migrate the application, run:

     ```
     migrateSecurityStore
     ```

```
(type="appPolicies", src="sourceContext",
 dst="destinationContext",
 configFile="myDir/jps-config.xml",
 srcApp="sourceApplication", dstApp="destinationApplication",
 overWrite="true")
```

where the following applies:

- The name of the corresponding `jpsContext` (previously created in Step 1) should be passed to the `src` and `dst` parameters.
- The name of the `jps-config.xml` file (previously modified in Step 1) should be passed to the `configFile` parameter. The value must be a fully qualified file name with complete path information.
- The name of the application being migrated is the value of the `srcApp` parameter. `srcApp` is a required parameter for `type="appPolicies"`. Without this value, an error message reading *The source application is undefined but required to migrate application-specific policies.* is displayed and the migration will fail.
- The name that is assigned to the application in the destination policy store is the value of the `dstApp` parameter. If this parameter is not passed, the name of the application in the destination store is the same as the name used in the source store.
- If the `overWrite` parameter is defined as true, policies specific to the destination application are replaced by policies from the source application. The default value of this parameter is false which migrates only the addictive changes.

13.5.2 Migrating From LDAP to XML

Following is the procedure to migrate policies from an LDAP-based directory to an XML-based policy store.

1. Modify `jps-config.xml` as described in this sub procedure.
   a. Create a `serviceInstance` for both the source and destination policy stores as illustrated in Example 13–5.

Example 13–5 LDAP to XML serviceInstances for Source and Destination Policy Stores

```
<!-- Source LDAP-based policy store instance -->
   <serviceInstance provider="ldap.policystore.provider"
      name="policystore.ldap.source">
    <description></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 XML-based policy store instance -->
   <serviceInstance name="dst.xml" provider="policystore.xml.provider"
    location="/scratch/divyasin/WithPSR/jazn-data-fscm.xml">
    <description>File Based Policy Store Service Instance</description>
   </serviceInstance>
```

b. Create a `serviceInstance` corresponding to the bootstrap credential used to access the destination LDAP directory as illustrated in Example 13–6.

Example 13–6 LDAP to XML serviceInstance for Bootstrap Credential

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

c. Create a `jpsContext` for both source and destination stores as illustrated in Example 13–7.

Example 13–7 LDAP to XML jpsContext for Source and Destination Policy Stores

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

<jpsContext name="destinationContext">
   <serviceInstanceRef ref="dst.xml"/>
</jpsContext>

<jpsContext name="bootstrap_credstore_context">
   <serviceInstanceRef ref="bootstrap.cred"/>
</jpsContext>
```

2. Start the WebLogic Scripting Tool.

   There is no need to connect the WebLogic Scripting Tool to the WebLogic Server as the migration command is an offline command.

3. Run the WebLogic Scripting Tool `migrateSecurityStore` command to migrate the policy store and application as follows.

   - To migrate the policy store, run:

     ```
     migrateSecurityStore
        (type="policyStore", src="sourceContext",
         dst="destinationContext",
         configFile="myDir/jps-config.xml")
     ```

     where the following applies:

     – The name of the corresponding `jpsContext` (previously created in Step 1) should be passed to the `src` and `dst` parameters.

     – The name of the `jps-config.xml` file (previously modified in Step 1) should be passed to the `configFile` parameter. The value must be a fully qualified file name with complete path information.

   - To migrate the application, run:

     ```
     migrateSecurityStore
        (type="appPolicies", src="sourceContext",
         dst="destinationContext",
         configFile="myDir/jps-config.xml",
         srcApp="sourceApplication", dstApp="destinationApplication",
         overWrite="true")
     ```

where the following applies:

- The name of the corresponding `jpsContext` (previously created in Step 1) should be passed to the `src` and `dst` parameters.
- The name of the `jps-config.xml` file (previously modified in Step 1) should be passed to the `configFile` parameter. The value must be a fully qualified file name with complete path information.
- The name of the application being migrated is the value of the `srcApp` parameter. `srcApp` is a required parameter for `type="appPolicies"`. Without this value, an error message reading *The source application is undefined but required to migrate application-specific policies.* is displayed and the migration will fail.
- The name that is assigned to the application in the destination policy store is the value of the `dstApp` parameter. If this parameter is not passed, the name of the application in the destination store is the same as the name used in the source store.
- If the `overWrite` parameter is defined as true, policies specific to the destination application are replaced by policies from the source application. The default value of this parameter is false which migrates only the addictive changes.

13.5.3 Migrating From XML to Database

Following is the procedure to migrate policies from an XML-based policy store to a database.

> **Note:** The value of the bootstrap.security.principal.key property needs to be populated with the key generated during reassociation of the policy, credential, and key stores from one repository type to another.

1. Modify `jps-config.xml` as described in this sub procedure.
   a. Create a `serviceInstance` for both the source and destination policy stores as illustrated in Example 13–8.

Example 13–8 XML to Database serviceInstances for Source and Destination Policy Stores

```
<!-- Source XML-based policy store instance -->
   <serviceInstance name="src.xml" provider="policystore.xml.provider"
     location="/scratch/divyasin/WithPSR/jazn-data-fscm.xml">
    <description>File Based Policy Store Service Instance</description>
</serviceInstance>

<!-- Destination DB-based policy store instance -->
   <serviceInstance provider="ldap.policystore.provider"
     name="policystore.db.destination">
    <description>DB Based Policy Store Service Instance</description>
     <property name="policystore.type" value="DB_ORACLE"/>
     <property name="jdbc.url"
       value="jdbc:oracle:thin:@sc.domainexample.com:1722:orcl"/>
     <property name="jdbc.driver" value="oracle.jdbc.driver.OracleDriver"/>
     <property name="bootstrap.security.principal.key"
        value="bootstrap_DWgpEJgXwhDIoLYVZ2OWd4R8wOA=" />
```

```
    <property name="oracle.security.jps.ldap.root.name" value="cn=jpsTestNode"/>
    <property name="oracle.security.jps.farm.name" value="cn=view_steph.atz"/>
  </serviceInstance>
```

b. Create a `serviceInstance` corresponding to the bootstrap credential used to access the destination LDAP directory as illustrated in Example 13–9.

Example 13–9 XML to Database serviceInstance for Bootstrap Credential

```
<!-- Bootstrap credentials to access source DB -->
  <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>
```

c. Create a `jpsContext` for both source and destination stores as illustrated in Example 13–10.

Example 13–10 XML to Database jpsContext for Source and Destination Policy Stores

```
<jpsContext name="sourceContext">
   <serviceInstanceRef ref="src.xml"/>
</jpsContext>

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

<jpsContext name="bootstrap_credstore_context">
   <serviceInstanceRef ref="bootstrap.cred"/>
</jpsContext>
```

2. Start the WebLogic Scripting Tool.

   There is no need to connect the WebLogic Scripting Tool to the WebLogic Server as the migration command is an offline command.

3. Run the WebLogic Scripting Tool `migrateSecurityStore` command to migrate the policy store and application as follows.

   - To migrate the policy store, run:

     ```
     migrateSecurityStore
        (type="policyStore", src="sourceContext",
         dst="destinationContext",
         configFile="/scratch/divyasin/WithPSR/jps-config.xml")
     ```

     where the following applies:

     – The name of the corresponding `jpsContext` (previously created in Step 1) should be passed to the `src` and `dst` parameters.

     – The name of the `jps-config.xml` file (previously modified in Step 1) should be passed to the `configFile` parameter. The value must be a fully qualified file name with complete path information.

   - To migrate the application, run:

     ```
     migrateSecurityStore
        (type="appPolicies", src="sourceContext",
         dst="destinationContext",
     ```

```
configFile="/scratch/divyasin/WithPSR/jps-config.xml",
srcApp="sourceApplication", dstApp="destinationApplication",
overWrite="true")
```

where the following applies:

- The name of the corresponding `jpsContext` (previously created in Step 1) should be passed to the `src` and `dst` parameters.
- The name of the `jps-config.xml` file (previously modified in Step 1) should be passed to the `configFile` parameter. The value must be a fully qualified file name with complete path information.
- The name of the application being migrated is the value of the `srcApp` parameter. `srcApp` is a required parameter for `type="appPolicies"`. Without this value, an error message reading *The source application is undefined but required to migrate application-specific policies.* is displayed and the migration will fail.
- The name that is assigned to the application in the destination policy store is the value of the `dstApp` parameter. If this parameter is not passed, the name of the application in the destination store is the same as the name used in the source store.
- If the `overWrite` parameter is defined as true, policies specific to the destination application are replaced by policies from the source application. The default value of this parameter is false which migrates only the addictive changes.

13.5.4 Migrating From Database to XML

Following is the procedure to migrate policies from a database to an XML-based policy store.

Note: The value of the `bootstrap.security.principal.key` property needs to be populated with the key generated during reassociation of the policy, credential, and key stores from one repository type to another.

---

1. Modify `jps-config.xml` as described in this sub procedure.
   a. Create a `serviceInstance` for both the source and destination policy stores as illustrated in Example 13–11.

Example 13–11 Database to XML serviceInstances for Source and Destination Policy Stores

```
<!-- Source DB-based policy store instance -->
   <serviceInstance provider="policystore.provider"
     name="policystore.db.source">
    <description>DB Based Policy Store Service Instance</description>
    <property name="policystore.type" value="DB_ORACLE"/>
    <property name="jdbc.url"
      value="jdbc:oracle:thin:@sc.domainexample.com:1722:orcl"/>
    <property name="jdbc.driver" value="oracle.jdbc.driver.OracleDriver"/>
    <property name="bootstrap.security.principal.key"
        value="bootstrap_DWgpEJgXwhDIoLYVZ2OWd4R8wOA=" />
    <property name="oracle.security.jps.ldap.root.name" value="cn=jpsTestNode"/>
    <property name="oracle.security.jps.farm.name" value="cn=view_steph.atz"/>
```

```
</serviceInstance>

<!-- Destination XML-based policy store instance -->
   <serviceInstance name="dst.xml" provider="policystore.xml.provider"
      location="/scratch/divyasin/WithPSR/jazn-data-fscm.xml">
    <description>File Based Policy Store Service Instance</description>
   </serviceInstance>
```

b. Create a `serviceInstance` corresponding to the bootstrap credential used to access the destination XML directory as illustrated in Example 13–12.

Example 13–12 Database to XML serviceInstance for Bootstrap Credential

```
<!-- Bootstrap credentials to access source and destination stores -->
   <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>
```

c. Create a `jpsContext` for both source and destination stores as illustrated in Example 13–13.

Example 13–13 Database to XML jpsContext for Source and Destination Policy Stores

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

<jpsContext name="destinationContext">
   <serviceInstanceRef ref="dst.xml"/>
</jpsContext>

<jpsContext name="bootstrap_credstore_context">
   <serviceInstanceRef ref="bootstrap.cred"/>
</jpsContext>
```

2. Start the WebLogic Scripting Tool for the migration.

   There is no need to connect the WebLogic Scripting Tool to the WebLogic Server as the migration command is an offline command.

3. Run the WebLogic Scripting Tool `migrateSecurityStore` command to migrate the policy store and application as follows.

   - To migrate the policy store, run:

     ```
     migrateSecurityStore
        (type="policyStore", src="sourceContext",
         dst="destinationContext",
         configFile="/scratch/divyasin/WithPSR/jps-config.xml")
     ```

     where the following applies:

     – The name of the corresponding `jpsContext` (previously created in Step 1) should be passed to the `src` and `dst` parameters.

     – The name of the `jps-config.xml` file (previously modified in Step 1) should be passed to the `configFile` parameter. The value must be a fully qualified file name with complete path information.

- To migrate the application, run:

```
migrateSecurityStore
   (type="appPolicies", src="sourceContext",
    dst="destinationContext",
    configFile="/scratch/divyasin/WithPSR/jps-config.xml",
    srcApp="sourceApplication", dstApp="destinationApplication",
    overWrite="true")
```

  where the following applies:

  - The name of the corresponding `jpsContext` (previously created in Step 1) should be passed to the `src` and `dst` parameters.
  - The name of the `jps-config.xml` file (previously modified in Step 1) should be passed to the `configFile` parameter. The value must be a fully qualified file name with complete path information.
  - The name of the application being migrated is the value of the `srcApp` parameter. `srcApp` is a required parameter for `type="appPolicies"`. Without this value, an error message reading *The source application is undefined but required to migrate application-specific policies.* is displayed and the migration will fail.
  - The name that is assigned to the application in the destination policy store is the value of the `dstApp` parameter. If this parameter is not passed, the name of the application in the destination store will be the same as the name used in the source store.
  - If the `overWrite` parameter is defined as true, policies specific to the destination application are replaced by policies from the source application. The default value of this parameter is false which migrates only the addictive changes.

13.6 Configuring Cache

Oracle Entitlements Server offers caching capabilites. The cache settings are configured in the `jps-config.xml` file. The following sections contain the appropriate information.

- Section 13.6.1, "Configuring Decision Caching"
- Section 13.6.2, "Configuring Attribute Caching"

13.6.1 Configuring Decision Caching

Authorization decision caching allows Oracle Entitlements Server to cache the result of an authorization call and use that decision in the future, if an identical call is made. The decision cache consists of two hierarchical levels.

- The first level (L1) caches subjects used in the authorization calls.
- The second level (L2) caches authorization and role mapping decisions for the given subject.

Note: The decision cache automatically invalidates itself if there is a change in the policy.

The key of the cache is the incoming Subject, Permission and attributes used during policy evaluation. The value of the cache is the decision and obligations.

All parameter names are prefixed with `oracle.security.jps.pdp`. Example 13–14 illustrates how the decision cache parameters might be set in `jps-config.xml`.

Example 13–14 XML To Configure Decision Caching

```
<serviceInstance name="pdp.service" provider="pdp.service.provider">
          …
 <property name="oracle.security.jps.pdp.AuthorizationDecisionCacheEnabled"
   value="true"/>
 <property name="oracle.security.jps.pdp.
   AuthorizationDecisionCacheEvictionCapacity"
   value="1000"/>
 <property name="oracle.security.jps.pdp.
   AuthorizationDecisionCacheEvictionPercentage"
   value="15"/>
 <property name="oracle.security.jps.pdp.AuthorizationDecisionCacheTTL"
   value="180"/>
          …
</serviceInstance>
```

Table 13–3 documents the decision caching parameters.

Table 13–3 Decision Caching Parameters

Name	Description	Accepted Values
`oracle.security.jps.pdp.AuthorizationDecisionCacheEnabled`	Optional parameter that specifies whether the policy decision cache should be enabled.	true (default) false
`oracle.security.jps.pdp.AuthorizationDecisionCacheEvictionCapacity`	Optional parameter that specifies the maximum capacity of the L1 cache. If the number of entries exceeds the value, some entries are evicted.	Integer representing number of entries 500 (default)
`oracle.security.jps.pdp.AuthorizationDecisionCacheEvictionPercentage`	Optional parameter that specifies the percentage of entries in L1 cache that have to be evicted when the maximum capacity has been reached. For example, if the maximum capacity is 200 and the value of this parameter is 10 then 20 entries are evicted from the cache.	Integer representing percent of entries 10 (default equals 10%)
`oracle.security.jps.pdp.AuthorizationDecisionCacheTTL`	Optional parameter that specifies a time-to-live value (in seconds) for entries in the L2 cache. It defines how long an authorization decision is cached.	Integer representing time in seconds 60 (default equals 1 minute)

13.6.2 Configuring Attribute Caching

Each passed attribute can be cached if the `cached` property is defined for it. A corresponding time-to-live (TTL) value must also be defined if `cached` is enabled. The key of the cache is the attribute URI. The value of the cache is the attribute object. Example 13–15 illustrates how the attribute cache might be set in `jps-config.xml`.

Example 13–15 XML To Configure Attribute Caching

```
<propertySet name="ootb.pip.attribute.age.based.on.myattr.rdbms">
    <property name="ootb.pip.attr.type" value="OOTB_PIP_ATTRIBUTE"/>
    <property name="ootb.pip.ref" value="pip.service.ootb.db"/>
```

```
    <property name="name" value="myattr"/>
    <property name="query" value=
       "select myattr from test where username=%SYS_USER%"/>
    <property name="cached" value="true"/>
    <property name="TTL" value="60"/>
</propertySet>
```

Note: If `cached` is not defined for the attribute, it will not be cached.

13.7 Logging

Oracle Entitlements Server uses the standard Java package `java.util.logging` for logging. The name of the logging setup file is `logging.properties`. It is the standard configuration file for the Java Development Kit (JDK) and it is located (by default) in `$JAVA_HOME/jre/lib/`.

Note: Configure the location of `logging.properties` by running the following on the command line with the actual path based on your install.

`-Djava.util.logging.config.file=`*`/path/filename`*

Java logging defines log levels to control output ranging from FINEST (indicating a highly detailed tracing message) to SEVERE (indicating fatal program errors and the like). Enabling logging at a given level also enables logging at all higher levels. Table 13–4 contains the specific logger properties that can be set to FINE (details for debugging and diagnosing problems) to provide information for purposes of troubleshooting server issues.

Table 13–4 Logging Server Issues

To Troubleshoot...	Set These Properties To FINE
Policy management issues	▪ `oracle.jps.policymgmt`
Basic authorization issues	▪ `oracle.jps.authorization`
Policy distribution issues	▪ `oracle.oes.pd` ▪ `oracle.oes.common`
Security Module issues	▪ `oracle.oes.sm`
Specific tool issues (for example, `enroll.sh`)	▪ `oracle.oes.tool`

After modifying `logging.properties`, ensure the `java.util.logging.config.file` system property is set by running the following command:

`-Djava.util.logging.config.file=/<`*`directory`*`>/logging.properties`

Example 13–16 is the default Oracle Entitlements Server `logging.properties` file.

Example 13–16 Default logging.properties Configuration File

```
############################################################
#  Default Logging Configuration File
```

```
# You can use a different file by specifying a filename
# with the java.util.logging.config.file system property.
# For example java -Djava.util.logging.config.file=myfile
############################################################
#  Global properties
############################################################
# "handlers" specifies a comma separated list of log Handler
# classes. These handlers will be installed during VM startup.
# Note that these classes must be on the system classpath.
# By default we only configure a ConsoleHandler, which will only
# show messages at the INFO and above levels.
handlers= java.util.logging.ConsoleHandler, java.util.logging.FileHandler
# To also add the FileHandler, use the following line instead.
#handlers= java.util.logging.FileHandler, java.util.logging.ConsoleHandler
# Default global logging level.
# This specifies which kinds of events are logged across
# all loggers. For any given facility this global level
# can be overriden by a facility specific level
# Note that the ConsoleHandler also has a separate level
# setting to limit messages printed to the console.
.level= WARNING
############################################################
# Handler specific properties.
# Describes specific configuration info for Handlers.
############################################################
# default file output is in user's home directory.
java.util.logging.FileHandler.pattern = /logs/java%u.log
java.util.logging.FileHandler.limit = 50000000
java.util.logging.FileHandler.count = 1
java.util.logging.FileHandler.formatter = java.util.logging.XMLFormatter
# Limit the message that are printed on the console to INFO and above.
java.util.logging.ConsoleHandler.level = FINE
java.util.logging.ConsoleHandler.formatter = java.util.logging.SimpleFormatter
############################################################
# Facility specific properties.
############################################################
# Provides extra control for each logger.
############################################################
# For example, set the com.xyz.foo logger to only log SEVERE
# messages:
oracle.oes.sm=FINE
oracle.oes.common=FINE
oracle.oes.tool=FINE
oracle.oes.pd=FINE
oracle.jps.policymgmt=FINE
oracle.jps.authorization=FINE
oracle.jps.common=FINE
```

13.8 Debugging

Policy debugging allows for analysis of the policy evaluation process for a given authorization call. The debugging capabilities of the Oracle Entitlements Server runtime are collected in the DebugStore which provides (among other information):

- Authorization and Role Mapping Policies (and the appropriate policy objects) involved during the evaluation process
- Attributes that were evaluated and their values
- Results of function invocations

- Roles granted or denied the requesting subject
- Policy evaluation result (GRANT, DENY, NOT APPLICABLE)
- Obligations and Conditions (if applicable)
- Failed authentications, missing group memberships and other incorrect identity values

A client can access debugging information in either of the following ways. The first option entails configuring debugging parameters to see the information in a standard debug log. The second option entails overloading the `isAccessAllowed_debug()` and `getRoles_debug()` methods. These two methods return an additional `DebugInfo` parameter which is populated with debug information during policy evaluation. This second option is used by end users as well as the policy simulation tool. The following sections have more information on these options.

- Section 13.8.1, "Enabling Debugging By Defining Parameters"
- Section 13.8.2, "Enabling Debugging Using Methods"

Section 13.8.3, "Debugging Policy Distribution" contains information on debugging the policy distribution process.

13.8.1 Enabling Debugging By Defining Parameters

The following sections contain information on how to configure debugging parameters and read the debugging logs.

- Configuring Logging for Debugging
- Searching Logs to Debug Authorization Policies

13.8.1.1 Configuring Logging for Debugging

When policy outcomes are other than expected, it may be useful to enable debugging so that the Security Module's logs will capture all events related to policy decisions. The following sections contain information on how to debug Authorization Policies created using Oracle Entitlements Server and how to debug the Policy Distribution process.

Oracle Entitlements Server uses the standard Java logging framework for capturing debugging details. Logging is the process of notifying an entity of a particular event. In the case of Oracle Entitlements Server, the entity can be a file or the Administration Console, and the event can be debugging information, runtime exceptions, or a record of actions taken by a user. The logging framework is configured based on the Oracle Entitlements Server deployment. More information is in the following sections.

- Section 13.8.1.1.1, "Configuring Logging for a Java Security Module"
- Section 13.8.1.1.2, "Configuring Logging for a WebLogic Server Security Module"

> **Note:** The `java.util.logging` package provides the classes and interfaces of the platform's core logging facilities.

13.8.1.1.1 Configuring Logging for a Java Security Module The following configurations must be made to enable logging when the Java Security Module is deployed.

- Run the following command when you start the Security Module to specify the logging configuration file:

```
-Djava.util.logging.config.file=logging.properties
```

- Set the logging level by adding the following lines to the configuration file:

  ```
  oracle.jps.authorization.level=FINEST
  oracle.jps.openaz.level=FINEST
  oracle.jps.authorization.debugstore=FINEST
  ```

  Keep the default level of the first property and set the level of the last two to FINEST; this configuration will log the result of a policy evaluation. Logging levels define the complexity of the logging record and include (from the least complex to the most) FINEST (simple information), FINER, FINE, CONFIG, INFO, WARNING and SEVERE (complex information).

If you don't specify a configuration file, the `logging.properties` file in *$JAVA_HOME*`/jre/lib/` is used. Example 13–17 illustrates how to configure `logging.properties` to log information to the Administration Console.

Example 13–17 Configuration for Administration Console Logging

```
#The messages will we printed to the standard output
handlers=java.util.logging.ConsoleHandler

#The default level for all loggers is INFO
.level=INFO

#Override the default level for OES authorization to FINEST
oracle.jps.authorization.level=FINEST
oracle.jps.openaz.level=FINEST

#Use default formatter to print the messages
java.util.logging.ConsoleHandler.formatter = java.util.logging.SimpleFormatter
```

Example 13–18 illustrates how to configure `logging.properties` to log information to a file.

Example 13–18 Configuration for File Logging

```
#The messages will be written to a file
handlers=java.util.logging.FileHandler

#The default level for all loggers is INFO
.level=INFO

#Override the default level for OES authorization to FINEST
oracle.jps.authorization.level=FINEST
oracle.jps.openaz.level=FINEST

#Configure file information. %h - is the user home directory
java.util.logging.FileHandler.pattern = %h/java%u.log
java.util.logging.FileHandler.limit = 50000
java.util.logging.FileHandler.count = 1
java.util.logging.FileHandler.formatter = java.util.logging.SimpleFormatter
```

13.8.1.1.2 Configuring Logging for a WebLogic Server Security Module To enable logging when the WebLogic Server Security Module is deployed, run the following command to specify the logging configuration file when you start the WebLogic Server domain.

```
startWeblogic.sh -Djava.util.logging.config.file=logging.properties
```

Tip: If you specify a relative path, the base directory is the domain home - not the directory where `startWeblogic.sh` is located

Other configurations relevant to the WebLogic Server Security Module are similar to those defined in Section 13.8.1.1.1, "Configuring Logging for a Java Security Module."

13.8.1.2 Searching Logs to Debug Authorization Policies

The following sections explain how to search for information recorded to the logging file. They include the commands to be run and, in many sections, sample output.

- Section 13.8.1.2.1, "Searching for PEP Request Information"
- Section 13.8.1.2.2, "Searching Against DebugStore Output"
- Section 13.8.1.2.3, "Searching for Security Module Cache Configuration Parameters"
- Section 13.8.1.2.4, "Searching for Principals"
- Section 13.8.1.2.5, "Searching for Resources and Actions"
- Section 13.8.1.2.6, "Searching for the Value of an Attribute"
- Section 13.8.1.2.7, "Searching for an Authorization Decision"
- Section 13.8.1.2.8, "Searching for the Value of an Obligation"
- Section 13.8.1.2.9, "Searching for Static Application Roles"

13.8.1.2.1 Searching for PEP Request Information Run the following command against the logging file to output PEP Request related information (including the Authentic Identity, the Runtime Resource, the Runtime Action and the Application Context).

```
grep "PepRequestImpl"
```

13.8.1.2.2 Searching Against DebugStore Output Example 13–19 is sample output illustrating what you might find in a log file when the DebugStore logger is enabled. The output includes the name of the Application, the requested Resource, Action, Subject and policies that were evaluated.

Example 13–19 DebugStore Sample Logging Output

```
========== Start Of Policy Evaluation Info ==========
Application: Library
 Requested Resource Type: LibraryResource
 Requested Resource: books/CrimeAndPunishment
 Requested Resource Present: false
 Requested Action: borrow
 Request Subject Principals:
     class weblogic.security.principal.WLSUserImpl:John
 Effective Roles Granted: [authenticated-role, Member, PrivilegedMember]
 Role-Mapping Policies:
     1.Policy Name: GrantPrivilegedMemberPolicy
     Matched Policy Principals:
         class weblogic.security.principal.WLSUserImpl:John
     Policy Principals Semantics: OR
     Policy Roles: [PrivilegedMember]
     Matched Policy Resources: NONE
     Policy Evaluation Result: NOT_APPLICABLE
     Policy Rules:
```

```
        Rule Name: GrantPrivilegedMemberRule
        Rule Effect: GRANT
        Rule Condition: (MembershipLength >= 5)
        Evaluated Rule Attributes and Functions:
          MembershipLength(Dynamic, Integer) = 4
          Rule Evaluation Result: NOT_APPLICABLE
    2.Policy Name: DenyMemberPolicy
    Matched Policy Principals:
        class weblogic.security.principal.WLSUserImpl:John
    Policy Principals Semantics: OR
    Policy Roles: [Member]
    Matched Policy Resources: [.*]
    Policy Evaluation Result: NOT_APPLICABLE
    Policy Rules:
        Rule Name: DenyMemberRule
        Rule Effect: DENY
        Rule Condition: (MemberState = MA)
        Evaluated Rule Attributes and Functions:
          MemberState(Dynamic, String) = CA
          Rule Evaluation Result: NOT_APPLICABLE

Static Role Grants:
    Principal= class oracle.security.jps.service.policystore.ApplicationRole:
        Member, Roles= [PrivilegedMember]
    Principal= class weblogic.security.principal.WLSUserImpl:John,
        Roles= [Member]

Denied Static Role Grants: NONE

Authorization Policies:
    1.Policy Name: DenyBorrowLibResourcesPolicy
    Matched Policy Principals:
        class oracle.security.jps.service.policystore.ApplicationRole:Member
        class weblogic.security.principal.WLSUserImpl:John
    Policy Principals Semantics: AND
    Matched Policy Resource-Actions:
        Resource = books/.*, Action = borrow
    Policy Obligations: NONE
    Policy Evaluation Result: NOT_APPLICABLE
    Policy Rules:
        Rule Name: DenyBorrowLibResourcesRule
        Rule Effect: DENY
        Rule Condition: (NumBorrowedBooks >= 6)
        Evaluated Rule Attributes and Functions:
            NumBorrowedBooks(Dynamic, Integer) = 2
        Rule Evaluation Result: NOT_APPLICABLE

    2.Policy Name: GrantBorrowLibResourcesPolicy
    Matched Policy Principals:
        class oracle.security.jps.service.policystore.ApplicationRole:Member
    Policy Principals Semantics: OR
    Matched Policy Resource-Actions:
        Resource = .*books.*, Action = ANY
        Resource = .*, Action = borrow
    Policy Obligations: NONE
    Policy Evaluation Result: GRANT
    Policy Rules:
        Rule Name: GrantBorrowLibResourcesRule
        Rule Effect: GRANT
        Rule Condition: ((NumBorrowedBooks < 4) && (age() >= 12))
```

```
        Evaluated Rule Attributes and Functions:
            NumBorrowedBooks(Dynamic, Integer) = 2
            age(Function, Integer) = 24
        Rule Evaluation Result: GRANT
========== End Of Policy Evaluation Info ==========
```

13.8.1.2.3 Searching for Security Module Cache Configuration Parameters

Run the following command against the logging file to output the cache configuration parameters for a particular Security Module.

```
grep "AuthorizationDecisionCacheTTL"
```

The following properties may be returned for this search. If a property does not appear in the log, it is not specified in `jps-config.xml`. In cases like this, the default value of the property is used.

- `AuthorizationDecisionCacheTTL` defines the time-to-live (in seconds) for the Authorization Decision cache. The default value is 60.
- `AuthorizationDecisionCacheEvictionPercentage` defines the percentage of authorization decisions to drop when the Authorization Decision cache has reached maximum capacity. The default value is 10.
- `AuthorizationDecisionCacheEvictionCapacity` defines the number used to evict the Authorization Decision cache if the decision cache size reaches this size. The default value is 500.
- `AuthorizationDecisionCacheEnabled` specifies whether the Authorization Decision cache is enabled. The default value is *true*.

Example 13–20 illustrates output for this search.

Example 13–20 Sample Output for Cache Configuration Parameters Search

```
oracle.security.jps.az.internal.runtime.service.AbstractPDPService

FINE: properties : {
oracle.security.jps.pdp.AuthorizationDecisionCacheTTL=60,
oracle.security.jps.pdp.AuthorizationDecisionCacheEvictionPercentage=10,
oracle.secuirty.jps.pdp.AuthorizationDecisionCacheEvictionCapacity=1000,
oracle.security.jps.pdp.AuthorizationDecisionCacheEnabled=true}
```

13.8.1.2.4 Searching for Principals

Run the following command against the logging file to output the names of Principals that have been received by Oracle Entitlements Server in the form of an authorization request.

```
grep "Principal:"
```

Example 13–21 illustrates the output for a Principal search.

Example 13–21 Sample Output for Principal Search

```
com.bea.security.providers.authorization.asi.AuthorizationProviderImpl
isAccessAllowed

FINE:subject: Subject:
     Principal: John
     Principal: Employee
     Principal: Administrator
     Principal: Principal Developer
```

13.8.1.2.5 Searching for Resources and Actions Run the following command against the logging file to output the Resources and Actions that have been received by Oracle Entitlements Server in the form of an authorization request.

```
grep "Resource:"
```

Example 13–22 illustrates how the information is returned. The defined values are the name of the policy object.

- Application = Lib
- Resource Type = libraryresourcetype
- Resource = Book
- Action = borrow

Example 13–22 Sample Output for Resource and Action Search

```
com.bea.security.providers.authorization.asi.AuthorizationProviderImpl
isAccessAllowed

FINE: Resource: resource=Lib/libraryresourcetype/Book, action=borrow
```

13.8.1.2.6 Searching for the Value of an Attribute Run the following command against the logging file to output the value of an attribute that has been received by Oracle Entitlements Server in the form of an authorization request.

```
grep "<name-of-the-attribute>:"
EXAMPLE: grep "NumberOfBorrowedBooksAttribute:"
```

Example 13–23 illustrates the returned information where the name of the attribute is `NumberOfBorrowedBooksAttribute` and the value is 2.

Example 13–23 Sample Output for the Value of an Attribute Search

```
com.bea.security.providers.authorization.asi.ARME.evaluator.EvalSession logDebug
FINE: getAttributeInternal: name: NumberOfBorrowedBooksAttribute; value: 2;
type: 3
```

13.8.1.2.7 Searching for an Authorization Decision Run the following command against the logging file to retrieve an authorization decision that has been stored.

```
grep "AccessResultLogger"
```

Example 13–24 illustrates the returned information and confirm that the authorization decision was affirmative.

Example 13–24 Sample Output for Authorization Decision Search

```
com.bea.security.providers.authorization.asi.AccessResultLogger log
FINE: Subject Subject:
Principal: John
Principal: Employee
Principal: Administrator
Principal: Principal Developer
 privilege borrow resource //app/policy/Lib/Book result PERMIT
```

13.8.1.2.8 Searching for the Value of an Obligation Run the following command against the logging file to output the value of a specific obligation.

```
grep "adding response attribute:" | grep  "obligations"
```

Example 13–25 illustrates the returned information indicating that the obligation (named `DenyObligation`) denies the request when the amount of library books the Principal currently has checked out is more than three; in this case, the Principal has five books checked out.

Example 13–25 Sample Output for Obligation Value Search

```
com.bea.security.providers.authorization.asi.AuthorizationProviderImpl
ARMEisAccessAllowed
FINE: adding response attribute: namespace=oracle.security.oes.authorization.
 name=obligations value={DenyObligation=
   { reason_part1=Too many borrowed books (max=3), reason_part2=5, }}
```

13.8.1.2.9 Searching for Static Application Roles Run the following command against the logging file to output the names of Application Roles granted statically.

```
grep "AbstractRoleManager" | grep "getGrantedStaticAppRoles"
```

Example 13–26 illustrates how two static roles are added to the list of principals: an authenticated-role – build-in role and Reader, an Application Role defined in the Application named Library.

Example 13–26 Sample Output for Static Role Search

```
oracle.security.jps.az.internal.runtime.entitymanager.AbstractRoleManager
getGrantedStaticAppRoles(Set)
FINER: RETURN [authenticated-role,
  ApplicationRoleLibrary/Readeruname:
  cn=Writer,cn=Roles,cn=Lib,cn=akapisni_dwps1_
  view1.atzsrg,cn=JPSContext,cn=jpsTestNode,guid:
  411EBF807CD411E0BF887FB1A0F3878F]
```

13.8.2 Enabling Debugging Using Methods

`isAccessAllowed_Debug` and `getRoles_Debug` are the methods that can be used in your code to return debugging information. `DebugInfo` is the object returned by the isAccessAllowed_Debug and `getRoles_Debug` calls. The following information has to be provided to the consumers of the policy debug feature (including the Policy Simulation tool) when overloading these methods:

- Roles granted to the requesting subject
- Roles denied the requesting subject
- Effective roles granted to the requesting subject
- Role Mapping and Authorization Policies that match the requesting subject(s), Resource Type, Resource and action (when applicable). Each policy must contain a name, a principal (that matches the user or role subject), a Resource or Resource Expression (that matches the input resource), action or role (depending on the policy type), policy rule (including a rule name, a GRANT or DENY effect, a string representation of the Condition, evaluated attributes and their values (including dynamic, function, and Resource Types) and a rule evaluation result (Grant, Deny, Not Applicable).

- Policy evaluation result (Grant, Deny, Not Applicable)
- Obligations (for Authorization Policies only)

See the *Oracle Fusion Middleware Management Java API Reference for Oracle Entitlements Server* for more details.

13.8.3 Debugging Policy Distribution

The Policy Distribution Component uses the policy management `Logger` interface. To enable debugging for the Policy Distribution Component, change the logging level of the `oracle.jps.policymgmt.level` property in the logging configuration file to FINEST. The procedure is documented in Section 13.8.1.1, "Configuring Logging for Debugging." The following should also be verified when debugging policy distribution.

- Check that the Security Module distribution mode parameter, `oracle.security.jps.runtime.pd.client.policyDistributionMode`, is correctly defined. See Appendix A.1, "Policy Distribution Configuration."
- When distributing policies in controlled-push or controlled-pull mode, check for the presence of a local cache file. The local cache directory is defined as the value of the `oracle.security.jps.runtime.pd.client.localpolicy.work_folder` parameter. Within this directory, see sub directories for each policy's `XML_CACHE_FILE` such as *`policyA/`* and *`policyB/`*. Ensure that the time stamp in the cache file matches the time when the last distribution was initiated. See Appendix A.1, "Policy Distribution Configuration."
- When distributing policies in controlled-push mode, ensure that the instance of the Security Module to which the policies are being distributed is displayed on the Application's Policy Distribution page, and that it is synchronized.
- Verify that the correct policies are being evaluated by debugging the Security Module side.

A

Installation and Configuration Parameters

This Appendix lists the parameters and accepted values that may be defined for Oracle Entitlements Server services using `jps-config.xml`, the configuration file used by Java EE containers. It is located in the *$DOMAIN_HOME*`/config/fmwconfig` directory. This Appendix is comprised of the following sections:

- Policy Distribution Configuration
- Security Module Configuration
- PDP Proxy Client Configuration
- Policy Store Service Configuration

A.1 Policy Distribution Configuration

The Policy Distribution Component is responsible for distributing policy objects and policies from the policy store to one or more Security Modules. It can distribute in a controlled-push mode, a controlled-pull mode, a non-controlled mode, or a mixed mode. Each mode entails different configurations.

- Section A.1.1, "Policy Distribution Component Server Configuration"
- Section A.1.2, "Policy Distribution Component Client Configuration"

A.1.1 Policy Distribution Component Server Configuration

Typically, configuration for the Policy Distribution Component to fetch policies and policy objects (in a scenario when it runs within Oracle Entitlements Server) is associated with the Policy Store configuration in the `jps-config.xml` file. Only in cases when data is pulled in a controlled manner (*controlled-pull mode*) is the Policy Distribution Component associated with the PDP Service configuration on the Security Module side. Table A–1 contains the configuration parameters.

Table A–1 Policy Distribution Server Configuration

Parameter Name	Information	Console Name
oracle.security.jps.pd.server.transactionalScope	Defines the scope of the policy distribution as either to one Security Module or to all Security Modules. If distribution fails when it involves only one Security Module, it does not affect distributions to other Security Modules. **Optional** **Accepted Values:** All (default), One	none
oracle.security.jps.register.waiting.interval	Defines the amount of time to delay policy distribution after a request for registration is received. **Optional** **Accepted Values:** time in seconds (default value is 0)	none

A.1.2 Policy Distribution Component Client Configuration

The Policy Distribution Component client is responsible for making policies available to the Security Module. Thus, the Policy Distribution Client configuration is always associated with the PDP Service configuration portion of the `jps-config.xml` file on the Security Module side. Configuration is different depending on the mode of distribution and the environment in which the Security Module is running. The following sections contain descriptions of the applicable configuration parameters.

- Section A.1.2.1, "Policy Distribution Component Client Java Standard Edition Configuration (Controlled Push Mode)"
- Section A.1.2.2, "Policy Distribution Component Client Java Enterprise Edition Container Configuration (Controlled Push Mode)"
- Section A.1.2.3, "Policy Distribution Client Configuration (Controlled Pull Mode)"
- Section A.1.2.4, "Policy Distribution Client Configuration (Non-controlled Mode)"
- Section A.1.2.5, "Policy Distribution Client Configuration (Mixed Mode)"

A.1.2.1 Policy Distribution Component Client Java Standard Edition Configuration (Controlled Push Mode)

Table A–2 compiles the parameters for the Policy Distribution Component client configuration when the Oracle Entitlements Server is running in a Java Standard Edition (JSE) environment and is configured to distribute data in the controlled-push mode.

Table A–2 Policy Distribution Client Configuration, JSE, Controlled Push Mode

Parameter Name	Information	Console Name
oracle.security.jps.runtime.pd.client.policyDistributionMode	Specifies the mode of policy distribution. *Controlled distribution* is initiated by the Policy Distribution Component, ensuring that the Security Module receives policy data that has been created or modified since the last distribution. **Mandatory** **Accepted Value:** controlled-push	Policy Distribution Mode
oracle.security.jps.runtime.pd.client.sm_name	Defines the name of the Security Module. **Mandatory** **Accepted Value:** Name of the Security Module	SM Name
oracle.security.jps.runtime.pd.client.localpolicy.work_folder	Defines the name of any directory in which local cache files are stored. If a value is not defined, a work directory will be created in the directory where `jps-config.xml` is kept. If the applicable Security Module is created in a JRF domain, the server name will be used to create a sub directory under the specified or default work folder which will be used as the actual work folder. **Optional** **Accepted Value:** The name of any directory in which local cache files will be stored. This directory must have read and write privileges.	Local Policy Work Folder
oracle.security.jps.runtime.pd.client.incrementalDistribution	Defines whether the distribution is incremental or flush. *Incremental distribution* is when only new and modified data is distributed. *Flush distribution* is when the Policy Distribution Component notifies the Security Module to cleanup locally stored policies in preparation for a complete re-distribution of all policy objects in the policy store. **Optional** **Accepted Values:** ■ false (policy distribution is flush for this Security Module) ■ true (default value; policy distribution is incremental for this Security Module if the required change logs are kept in the policy store)	Incremental Distribution

Table A–2 (Cont.) Policy Distribution Client Configuration, JSE, Controlled Push Mode

Parameter Name	Information	Console Name
oracle.security.jps.runtime.pd.client.registrationRetryInterval	When a Security Module starts, it registers itself with the Policy Distribution Component to ensure the local policy cache is up to date. If registration fails, it will retry each time this interval of time passes until successful. **Optional** **Accepted Value:** time in seconds (default value is 5)	Registration Retry Interval
oracle.security.jps.runtime.policyDistributionWaitTime	If this value is defined and not equal to zero, it specifies the amount of time that a Security Module will wait for initial policy distribution to happen. During this wait period, authorization requests are blocked until either the initial policy distribution completes or the configured period expires. **Optional** **Accepted Value:** time in seconds (default value is 60)	Wait Distribution Time (seconds)
oracle.security.jps.runtime.pd.client.RegistrationServerURL	Defines the URL of the Oracle Entitlements Server Administration Server. Used by the Security Module to register itself with Oracle Entitlements Server when it starts. **Mandatory** **Accepted Value:** URL	Registration Server URL
oracle.security.jps.runtime.pd.client.backupRegistrationServerURL	Defines a backup URL for the Oracle Entitlements Server Administration Server. Used by the Security Module to register itself with Oracle Entitlements Server when it starts if the primary URL (parameter above) is unavailable. **Optional** (although if not configured Oracle Entitlements Server failover will not work) **Accepted Value:** URL	Backup Registration Server URL
oracle.security.jps.runtime.pd.client.DistributionServicePort	Defines the port to which a remote Policy Distributor will push policy updates. **Mandatory** **Accepted Value:** port number	Distribution Service Port

Table A–2 (Cont.) Policy Distribution Client Configuration, JSE, Controlled Push Mode

Parameter Name	Information	Console Name
oracle.security.jps.pd.client.sslMode	Defines whether communication between the Policy Distribution Component server and client will use the Secure Sockets Layer (SSL) protocol or not. **Mandatory** **Accepted Values:** none, two-way (default value)	SSL Mode
oracle.security.jps.pd.client.ssl.identityKeyStoreFileName	Defines the name of the Identity Key Store file in which client certificates are stored. Used for SSL communication between the Security Module and the Policy Distribution Component. **Mandatory** **Accepted Value:** the name of the keystore file	SSL Identity Key Store File Name
oracle.security.jps.pd.client.ssl.trustKeyStoreFileName	Defines the name of the Trust Key Store file where Certificate Authority (CA) certificates are stored. Used for SSL communication between the Security Module and the Policy Distribution Component. **Mandatory** **Accepted Value:** the name of the identity key store file	SSL Trust Key Store File Name
oracle.security.jps.pd.client.ssl.identityKeyStoreKeyAlias	Defines an Identity Key alias to identify the client certificate used for SSL communication between the Security Module and the Policy Distribution Component. **Optional** (if only one alias exists in the identity keystore there is no need to specify this value) **Accepted Value:** the identity key alias	SSL Identity Key Store Key Alias
oracle.security.jps.runtime.pd.client.SMinstanceType	Defines the type of Security Module to which the Policy Distribution Component client is connecting. **Mandatory** **Accepted Value:** java (Other accepted values include wls, RMI and ws. Because this table covers the Java Security Module only, the value must be java.)	Configured during OES Client installation only.

Table A–2 (Cont.) Policy Distribution Client Configuration, JSE, Controlled Push Mode

Parameter Name	Information	Console Name
oracle.security.jps.runtime.pd.client.localpolicy.JCEProviderName	Defines which JCE provider will be used. **Optional** **Accepted Values:** SunJCE, JsafeJCE; no default value is defined. The value is case-sensitive. If no value is provided, the default JDK provider is used.	
oracle.security.jps.runtime.pd.client.localpolicy.CipherKeyLength	Defines the key length used for the Cipher class available from the specified JCE provider. **Optional** **Accepted Values:** 128, 192, 256; default value is 128.	
oracle.security.jps.runtime.pd.client.localpolicy.CipherModePadding	Defines a cipher algorithm name, mode and padding schema used for the Cipher class available from the specified JCE provider. It is not case-sensitive. The format should be: *algorithm name/mode/padding* **Optional** **Accepted Values:** default value is `AES/CBC/PKCS5Padding`; others include `AES/CBC/PKCS5Padding` or `AES/GCM/NoPadding`.	

A.1.2.2 Policy Distribution Component Client Java Enterprise Edition Container Configuration (Controlled Push Mode)

Table A–3 compiles the parameters for the Policy Distribution Component client configuration when the Oracle Entitlements Server is running in a Java Enterprise Edition (JEE) environment and is configured to distribute data in the controlled-push mode.

Table A–3 Policy Distribution Client Configuration, JEE, Controlled Push Mode

Parameter Name	Information	Console Name
oracle.security.jps.runtime.pd.client.policyDistributionMode	Specifies the mode of policy distribution. *Controlled distribution* is initiated by the Policy Distribution Component, ensuring that the Security Module receives policy data that has been created or modified since the last distribution. **Mandatory** **Accepted Value:** controlled-push	Policy Distribution Mode
oracle.security.jps.runtime.pd.client.sm_name	Defines the name of the Security Module. **Mandatory** **Accepted Value:** Name of the Security Module	SM Name

Table A–3 (Cont.) Policy Distribution Client Configuration, JEE, Controlled Push Mode

Parameter Name	Information	Console Name
oracle.security.jps.runtime.pd.client.localpolicy.work_folder	Defines the name of any directory in which local cache files are stored. If a value is not defined, a work directory will be created in the directory where `jps-config.xml` is kept. If the applicable Security Module is created in a JRF domain, the server name will be used to create a sub directory under the specified or default work folder which will be used as the actual work folder. **Optional** **Accepted Value:** The name of any directory in which local cache files will be stored. This directory must have read and write privileges.	Local Policy Work Folder
oracle.security.jps.runtime.pd.client.incrementalDistribution	Defines whether the distribution is incremental or flush. *Incremental distribution* is when new and modified data is distributed. *Flush distribution* is when the Policy Distribution Component notifies the Security Module to cleanup locally stored policies in preparation for a complete re-distribution of all policy objects in the policy store. **Optional** **Accepted Values:** ■ false (policy distribution is flush for this Security Module) ■ true (default value; policy distribution is incremental for this Security Module if the required change logs are kept in the policy store)	Incremental Distribution
oracle.security.jps.runtime.pd.client.registrationRetryInterval	When a Security Module starts, it registers itself with the Policy Distribution Component to ensure the local policy cache is up to date. If registration fails, it will retry each time this interval of time passes until successful. **Optional** **Accepted Value:** time in seconds (default value is 5)	Registration Retry Interval (seconds)

Table A–3 (Cont.) Policy Distribution Client Configuration, JEE, Controlled Push Mode

Parameter Name	Information	Console Name
oracle.security.jps.runtime.policyDistributionWaitTime	If this value is defined and not equal to zero, it specifies the amount of time that a Security Module will wait for initial policy distribution to happen. During this wait period, authorization requests are blocked until either the initial policy distribution completes or the configured period expires. **Optional** **Accepted Value:** time in seconds (default value is 60)	Wait Distribution Time (seconds)
oracle.security.jps.runtime.pd.client.RegistrationServerURL	Defines the URL of the Oracle Entitlements Server Administration Server. Used by the Security Module to register itself with Oracle Entitlements Server when it starts. **Mandatory** **Accepted Value:** URL	Registration Server URL
oracle.security.jps.runtime.pd.client.backupRegistrationServerURL	Defines a backup URL for the Oracle Entitlements Server Administration Server. Used by the Security Module to register itself with Oracle Entitlements Server when it starts if the primary URL (parameter above) is unavailable. **Optional** (although if not configured Oracle Entitlements Server failover will not work) **Accepted Value:** URL	Backup Registration Server URL
oracle.security.jps.runtime.pd.client.SMinstanceType	Defines the type of Security Module to which the Policy Distribution Component client is connecting. **Mandatory** **Accepted Values:** ▪ was ▪ wls	Configured during OES Client installation only.
oracle.security.jps.runtime.pd.client.DistributionServiceURL	Defines the URL to which the remote Policy Distributor will push policy updates. **Mandatory** **Accepted Values:** URL	

Table A–3 (Cont.) Policy Distribution Client Configuration, JEE, Controlled Push Mode

Parameter Name	Information	Console Name
oracle.security.jps.runtime.pd.client.localpolicy.JCEProviderName	Defines which JCE provider will be used. It is optional and case sensitive. **Optional** **Accepted Values:** SunJCE, JsafeJCE; no default value is defined. If no value is provided, the default JDK provider is used.	
oracle.security.jps.runtime.pd.client.localpolicy.CipherKeyLength	Defines the key length used for the Cipher class available from the specified JCE provider. **Optional** **Accepted Values:** 128, 192, 256; default value is 128.	
oracle.security.jps.runtime.pd.client.localpolicy.CipherModePadding	Defines a cipher algorithm name, mode and padding schema used for the Cipher class available from the specified JCE provider. It is not case-sensitive. The format should be: *algorithm name/mode/padding* **Optional** **Accepted Values:** default value is `AES/CBC/PKCS5Padding`; others include `AES/CBC/PKCS5Padding` or `AES/GCM/NoPadding`.	

A.1.2.3 Policy Distribution Client Configuration (Controlled Pull Mode)

Table A–4 compiles the parameters for the Policy Distribution Component client configuration when the Oracle Entitlements Server is running in either a JEE or a JSE environment and is configured to distribute data in the controlled-pull mode.

Table A–4 Policy Distribution Client Configuration, Controlled Pull Mode

Parameter Name	Information	Console Name
oracle.security.jps.runtime.pd.client.policyDistributionMode	Specifies the mode of policy distribution. *Controlled distribution* is initiated by the Policy Distribution Component, ensuring that the Security Module receives policy data that has been created or modified since the last distribution. **Mandatory** **Accepted Value:** controlled-pull	Policy Distribution Mode
oracle.security.jps.runtime.pd.client.sm_name	Defines the name of the Security Module. **Mandatory** **Accepted Value:** the name of the Security Module	SM Name

Table A–4 (Cont.) Policy Distribution Client Configuration, Controlled Pull Mode

Parameter Name	Information	Console Name
oracle.security.jps.runtime.pd.client.localpolicy.work_folder	Defines the name of any directory in which local cache files are stored. If a value is not defined, a work directory will be created in the directory where `jps-config.xml` is kept. If the applicable Security Module is created in a JRF domain, the server name will be used to create a sub directory under the specified or default work folder which will be used as the actual work folder. **Optional** **Accepted Value:** The name of any directory in which local cache files will be stored. This directory must have read and write privileges.	Local Policy Work Folder
oracle.security.jps.runtime.pd.client.incrementalDistribution	Defines whether the distribution is incremental or flush. *Incremental distribution* is when new and modified data is distributed. *Flush distribution* is when the Policy Distribution Component notifies the Security Module to cleanup locally stored policies in preparation for a complete re-distribution of all policy objects in the policy store. **Optional** **Accepted Values:** ■ false (policy distribution is flush for the Security Module) ■ true (default value; policy distribution is incremental for this Security Module if the required change logs are kept in the policy store)	Incremental Distribution
oracle.security.jps.runtime.policyDistributionWaitTime	If this value is defined and not equal to zero, it specifies the amount of time that a Security Module will wait for initial policy distribution to happen. During this wait period, authorization requests are blocked until either the initial policy distribution completes or the configured period expires. **Optional** **Accepted Value:** time in seconds (default value is 60)	Wait Distribution Time (seconds)

Table A–4 (Cont.) Policy Distribution Client Configuration, Controlled Pull Mode

Parameter Name	Information	Console Name
oracle.security.jps.pd.client .PollingTimerEnabled	Enables a periodic check for policy updates in the Policy Store. Can be set to false to disable polling for environment when policies are not expected to be modified. **Optional** **Accepted Values:** ■ false ■ true (default value)	
oracle.security.jps.pd.client .PollingTimerInterval	Defines the interval of time in which the Policy Distribution Component will check for policy data changes. **Optional** **Accepted Value:** time in seconds (default value of 600)	
oracle.security.jps.ldap.root.name	Defines the top (root) entry of the LDAP policy store directory information tree (DIT). **Mandatory** **Accepted Value:** the top (root) entry of the LDAP policy store directory information tree (DIT)	LDAP Root Name
oracle.security.jps.farm.name	Defines the RDN format of the domain node in the LDAP policy store. **Mandatory** **Accepted Value:** name of the domain	Farm Name
jdbc.url	Takes a URL that points to the database. **Mandatory** (if using Java Database Connectivity API to connect to policy store) **Accepted Value:** URL	JDBC URL
jdbc.driver	Location of the driver if using Java Database Connectivity API to connect to an Apache Derby database. **Mandatory** **Accepted Value:** driver	JDBC Driver

Table A–4 (Cont.) Policy Distribution Client Configuration, Controlled Pull Mode

Parameter Name	Information	Console Name
datasource.jndi.name	The JNDI name of the JDBC data source instance. The instance may correspond to a single source or multi-source datasource. Valid in only JEE applications. Applies only to database stores. **Mandatory** **Accepted Value:** name of JNDI data source; for example, `jdbc/APMDBDS`.	Datasource JNDI Name
security.principal	The name of the user with access rights to the database. **Mandatory** **Accepted Value:** Database user name	
security.credential	The password of the user with access rights to the database. **Optional** **Accepted Value:** Password associated with the database user in clear text; instead of storing the password in clear text, use `bootstrap.security.principal.map`.	
bootstrap.security.principal.key	The key for the password credentials to access the policy store. Credentials are stored in the Credential Store Framework (CSF) store. **Mandatory** **Accepted Value:** CSF credential key	Bootstrap Security Principal Key
bootstrap.security.principal.map	The map for the password credentials to access the policy store. Credentials are stored in the CSF store. **Mandatory** **Accepted Value:** name of the CSF credential map	Bootstrap Security Principal Map
oracle.security.jps.runtime.pd.client.localpolicy.JCEProviderName	Defines which JCE provider will be used. It is optional and case sensitive. **Optional** **Accepted Values:** SunJCE, JsafeJCE; no default value is defined. If no value is provided, the default JDK provider is used.	

Table A–4 (Cont.) Policy Distribution Client Configuration, Controlled Pull Mode

Parameter Name	Information	Console Name
oracle.security.jps.runtime.pd.client.localpolicy.CipherKeyLength	Defines the key length used for the Cipher class available from the specified JCE provider. **Optional** **Accepted Values:** 128, 192, 256; default value is 128.	
oracle.security.jps.runtime.pd.client.localpolicy.CipherModePadding	Defines a cipher algorithm name, mode and padding schema used for the Cipher class available from the specified JCE provider. It is not case-sensitive. The format should be: *algorithm name/mode/padding* **Optional** **Accepted Values:** default value is `AES/CBC/PKCS5Padding`; others include `AES/CBC/PKCS5Padding` or `AES/GCM/NoPadding`.	

A.1.2.4 Policy Distribution Client Configuration (Non-controlled Mode)

Table A–5 compiles the parameters for Policy Distribution Component client configuration when the Oracle Entitlements Server is running in either a JEE or a JSE environment and is configured to distribute data in the non-controlled mode.

Table A–5 Policy Distribution Client Configuration, Non-controlled Mode

Parameter Name	Information	Console Name
oracle.security.jps.runtime.pd.client.policyDistributionMode	Specifies the mode of policy distribution. *Non-controlled distribution* is when the Security Module periodically retrieves policy data from a policy store (or from a component that serves as an intermediary between the two). **Optional** **Accepted Value:** non-controlled (default value)	Policy Distribution Mode

A.1.2.5 Policy Distribution Client Configuration (Mixed Mode)

Table A–4 compiles the parameters for the Policy Distribution Component client configuration when the PDP is running in either a JEE or a JSE environment and is configured to distribute data in mixed mode. Mixed mode is a distribution combination of controlled-pull and uncontrolled mode.

Table A–6 Policy Distribution Client Configuration, Mixed Mode

Parameter Name	Information	Console Name
oracle.security.jps.runtime.pd.client.policyDistributionMode	Specifies the mode of policy distribution. *Controlled distribution* is initiated by the Policy Distribution Component, ensuring that the Security Module receives policy data that has been created or modified since the last distribution. **Mandatory** **Accepted Value:** mixed	Policy Distribution Mode
oracle.security.jps.runtime.pd.client.sm_name	Defines the name of the Security Module. **Mandatory** **Accepted Value:** the name of the Security Module	SM Name
oracle.security.jps.runtime.pd.client.localpolicy.work_folder	Defines the name of any directory in which local cache files are stored. If a value is not defined, a work directory will be created in the directory where `jps-config.xml` is kept. If the applicable Security Module is created in a JRF domain, the server name will be used to create a sub directory under the specified or default work folder which will be used as the actual work folder. **Optional** **Accepted Value:** The name of any directory in which local cache files will be stored. This directory must have read and write privileges.	Local Policy Work Folder
oracle.security.jps.runtime.pd.client.incrementalDistribution	Defines whether the distribution is incremental or flush. *Incremental distribution* is when new and modified data is distributed. *Flush distribution* is when the Policy Distribution Component notifies the Security Module to cleanup locally stored policies in preparation for a complete re-distribution of all policy objects in the policy store. **Optional** **Accepted Values:** ■ false (policy distribution is flush for the Security Module) ■ true (default value; policy distribution is incremental for this Security Module if the required change logs are kept in the policy store)	Incremental Distribution

Table A–6 (Cont.) Policy Distribution Client Configuration, Mixed Mode

Parameter Name	Information	Console Name
oracle.security.jps.runtime.policyDistributionWaitTime	If this value is defined and not equal to zero, it specifies the amount of time that a Security Module will wait for initial policy distribution to happen. During this wait period, authorization requests are blocked until either the initial policy distribution completes or the configured period expires. **Optional** **Accepted Value:** time in seconds (default value is 60)	Wait Distribution Time (seconds)
oracle.security.jps.pd.client.PollingTimerEnabled	Enables a periodic check for policy updates in the Policy Store. Can be set to false to disable polling for environment when policies are not expected to be modified. **Optional** **Accepted Values:** ■ false ■ true (default value)	Polling Timer
oracle.security.jps.pd.client.PollingTimerInterval	Defines the interval of time in which the Policy Distribution Component will check for policy data changes. **Optional** **Accepted Value:** time in seconds (default value of 600)	Polling Timer Interval
oracle.security.jps.runtime.pd.client.localpolicy.JCEProviderName	Defines which JCE provider will be used. It is optional and case sensitive. **Optional** **Accepted Values:** SunJCE, JsafeJCE; no default value is defined. If no value is provided, the default JDK provider is used.	N/A
oracle.security.jps.runtime.pd.client.localpolicy.CipherKeyLength	Defines the key length used for the Cipher class available from the specified JCE provider. **Optional** **Accepted Values:** 128, 192, 256; default value is 128.	N/A

Table A–6 (Cont.) Policy Distribution Client Configuration, Mixed Mode

Parameter Name	Information	Console Name
oracle.security.jps.runtime.pd.client.localpolicy.CipherModePadding	Defines a cipher algorithm name, mode and padding schema used for the Cipher class available from the specified JCE provider. It is not case-sensitive. The format should be: *algorithm name/mode/padding* **Optional** **Accepted Values:** default value is `AES/CBC/PKCS5Padding`; others include `AES/CBC/PKCS5Padding` or `AES/GCM/NoPadding`.	N/A
In Mixed Mode, the following nine properties should be configured for the Policy Store and not the Security Module. See Section A.4, "Policy Store Service Configuration."		
oracle.security.jps.ldap.root.name	Defines the top (root) entry of the LDAP policy store directory information tree (DIT). **Mandatory** **Accepted Value:** the top (root) entry of the LDAP policy store directory information tree (DIT)	LDAP Root Name
oracle.security.jps.farm.name	Defines the RDN format of the domain node in the LDAP policy store. **Mandatory** **Accepted Value:** name of the domain	Farm Name
jdbc.url	Takes a URL that points to the database. **Mandatory** (if using Java Database Connectivity API to connect to policy store) **Accepted Value:** URL	JDBC URL
jdbc.driver	Location of the driver if using Java Database Connectivity API to connect to an Apache Derby database. **Mandatory** **Accepted Value:** driver	JDBC Driver
datasource.jndi.name	The JNDI name of the JDBC data source instance. The instance may correspond to a single source or multi-source datasource. Valid in only JEE applications. Applies only to database stores. **Mandatory** **Accepted Value:** name of JNDI data source; for example, `jdbc/APMDBDS`.	Datasource JNDI Name

Table A–6 (Cont.) Policy Distribution Client Configuration, Mixed Mode

Parameter Name	Information	Console Name
security.principal	The name of the user with access rights to the database. **Mandatory** **Accepted Value:** Database user name	Username
security.credential	The password of the user with access rights to the database. **Mandatory** **Accepted Value:** Password associated with the database user	Password
bootstrap.security.principal.key	The key for the password credentials to access the policy store. Credentials are stored in the Credential Store Framework (CSF) store. **Mandatory** **Accepted Value:** CSF credential key	Bootstrap Security Principal Key
bootstrap.security.principal.map	The map for the password credentials to access the policy store. Credentials are stored in the CSF store. **Mandatory** **Accepted Value:** name of the CSF credential map	Bootstrap Security Principal Map

A.2 Security Module Configuration

This section covers the configurations for the various types of Security Modules and their proxy clients.

- Section A.2.1, "Java Security Module"
- Section A.2.2, "Web Services Security Module"
- Section A.2.3, "Web Services Security Module on WebLogic Server"
- Section A.2.4, "RMI Security Module"
- Section A.2.5, "WebLogic Server Security Module"
- Section A.2.6, "WebLogic Server Security Module Discovery Mode"

A.2.1 Java Security Module

Table A–7 compiles the parameters to configure the Java Security Module embedded in either a JSE or a JEE container.

Table A–7 Java Security Module Configuration Parameters

Parameter Name	Information	Console Name
oracle.security.jps.policystore.rolemember.cache.type	Defines the role member cache type. Valid in J2EE and J2SE applications. Applies to LDAP and database stores. **Optional** **Accepted Values** ■ SOFT (cleaning of a cache of this type relies on the garbage collector when there is a memory crunch) ■ WEAK (behavior of a cache of this type is similar to a cache of type SOFT but the garbage collector cleans it more frequently) ■ STATIC (default value; cache objects are statically cached and can be cleaned explicitly only according to the applied cache strategy, such as FIFO; the garbage collector does not clean a cache of this type)	Rolemember Cache Type
oracle.security.jps.policystore.rolemember.cache.strategy	Defines the type of strategy used in the role member cache. Valid in J2EE and J2SE applications. Applies to LDAP and database stores. **Optional** **Accepted Values** ■ 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) ■ FIFO (default value; the cache implements the first-in-first-out strategy)	Rolemember Cache Strategy
oracle.security.jps.policystore.rolemember.cache.size	Defines the number of roles kept in the role member cache. Valid in J2EE and J2SE application. Applies to LDAP and database stores. **Optional** **Accepted Value:** number (default value is 1000)	Rolemember Cache Size

Table A–7 (Cont.) Java Security Module Configuration Parameters

Parameter Name	Information	Console Name
oracle.security.jps.policystore.rolemember.cache.warmup.enable	Controls the way the Application Role membership cache is created. Valid in J2EE and J2SE applications. Applies to LDAP and database stores. **Optional** **Accepted Values** ■ true (the cache is created at server startup; use when the number of users and groups is significantly higher than the number of Application Roles) ■ false (default value; the cache is created on demand - lazy loading; use when the number of Application Roles is very high)	Rolemember Cache Warmup Enable
oracle.security.jps.policystore.policy.lazy.load.enable	Enables or disables the policy lazy load. Valid in J2EE and J2SE applications. Applies to LDAP and database stores. **Optional** **Accepted Values** ■ false ■ true (default value)	Policy Lazy Load Enable
oracle.security.jps.policystore.policy.cache.strategy	Defines the type of strategy used in the permission cache. Valid in J2EE and J2SE applications. Applies to LDAP and database stores. **Optional** **Accepted Values** ■ 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.) ■ PERMISSION_FIFO (default value; the cache implements the first-in-first-out strategy)	Policy Cache Strategy
oracle.security.jps.policystore.policy.cache.size	Defines the number of permissions kept in the permission cache. Valid in J2EE and J2SE applications. Applies to LDAP and database stores. **Optional** **Accepted Value:** number (default value is 1000)	Policy Cache Size
oracle.security.jps.policystore.cache.updateable	Defines whether the policy cache is incrementally updated for management operations on policy data. **Optional** **Accepted Values** ■ false ■ true (default value)	Policy Cache Updatable

Table A–7 (Cont.) Java Security Module Configuration Parameters

Parameter Name	Information	Console Name
oracle.security.jps.policystore.refresh.enable	Enables or disables the policy store refresh. If this property is set, `oracle.security.jps.ldap.cache.enable` cannot be set. Valid in J2EE and J2SE applications. Applies to LDAP and database stores. **Optional** **Accepted Values:** ■ false ■ true (default value)	Refresh Enable
oracle.security.jps.policystore.refresh.purge.timeout	Defines the time in milliseconds after which the policy store cache is purged. Valid in J2EE and J2SE applications. Applies to LDAP and database stores. **Optional** **Accepted Value:** time in milliseconds; default value is 43200000 which equals 12 hours	Refresh Purge Timeout (milliseconds)
oracle.security.jps.ldap.policystore.refresh.interval	Defines the interval of time in which the policy store is polled for changes. Valid in J2EE and J2SE applications. Applies to LDAP and database stores. **Optional** **Accepted Value:** time in milliseconds; default value is 600000 which equals 10 minutes	Refresh Purge Interval (milliseconds)
oracle.security.jps.pdp.missingAppPolicyQueryTTL	Defines the interval of time to avoid frequently querying a non-exist Application (`ApplicationPolicy`) object. **Optional** **Accepted Value:** time to live in milliseconds (default value is 60000)	Missing App Policy Query TTL
oracle.security.jps.pdp.AuthorizationDecisionCacheEnabled	Specifies whether the authorization cache should be enabled. Valid in J2EE and J2SE applications. Applies to XML, LDAP, and database stores. **Optional** **Accepted Values** ■ false ■ true (default value)	Decision Cache Enabled
oracle.security.jps.pdp.AuthorizationDecisionCacheEvictionCapacity	Defines the maximum number of authorization and role mapping sessions to maintain. When the maximum is reached, old sessions are dropped and reestablished when needed. Valid in J2EE and J2SE applications. Applies to XML, LDAP, and database stores. **Optional** **Accepted Value:** number (default value is 500)	Decision Cache Eviction Capacity

Table A–7 (Cont.) Java Security Module Configuration Parameters

Parameter Name	Information	Console Name
oracle.security.jps.pdp.AuthorizationDecisionCacheEvictionPercentage	Defines the percentage of sessions to drop when the eviction capacity is reached. Valid in J2EE and J2SE applications. Applies to XML, LDAP, and database stores. **Optional** **Accepted Value:** number (default value is 10)	Decision Cache Eviction Percentage
oracle.security.jps.pdp.AuthorizationDecisionCacheTTL	Defines the number of seconds during which session data is cached. Valid in J2EE and J2SE applications. Applies to XML, LDAP, and database stores. **Optional** **Accepted Value:** time in seconds (default value is 60)	Decision Cache TTL (seconds)
oracle.security.jps.pdp.anonymousrole.enable	Specifies whether anonymous role has to be added to anonymous subject for policy matching. **Optional** **Accepted Values** ■ false ■ true (default value)	Anonymous Role Enable
oracle.security.jps.pdp.authenticatedrole.enable	Specifies whether authenticated role has to be added to authenticated subject for policy matching. **Optional** **Accepted Values** ■ false ■ true (default value)	Authenticated Role Enable
oracle.security.jps.pdp.ComputeAppRolesOnceOnBulkAtz	Specifies whether Application Roles should be computed only once within a single bulk authorization call. For example, if a client calls the `checkBulkAuthorization()` method and passes ten resources to it, the roles will be calculated once if the value is *true* or once for every individual resource (ten times) if the parameter is *false*. **Optional** **Accepted Values** ■ false ■ true (default value)	
oracle.security.jps.pdp.AuthorizationPerUserDecisionCacheSize	Specifies the maximum number of authorization decisions cached for each Subject; if the second level decision cache size reaches this size, decisions are evicted from the cache. **Optional** **Accepted Value:** number of decisions (default value is 1000)	

A.2.2 Web Services Security Module

Table A–8 compiles the parameters to configure the Web Services Security Module embedded in either a JSE or a JEE container.

Table A–8 Web Services Security Module Configuration Parameters

Parameter Name	Information	Console Name
oracle.security.jps.pdp.wssm.WSServiceRegistryPortNumber	Defines the port on which the Web Services Security Module listens. **Mandatory** **Accepted Value:** port number	
oracle.security.jps.pdp.wssm.WSServiceRegistryHost	Defines the name of the server on which the Web Services Security Module is running. **Optional** **Accepted Value:** server name (default value is localhost)	
oracle.security.jps.pdp.wssm.Protocol	Defines the transport protocol used between the Policy Distribution Component client and server. **Optional** **Accepted Values** ▪ https ▪ http (default value)	
oracle.security.jps.pdp.sm.IdentityCacheEnabled	Specifies whether the identity cache is being used. If not set, no identity cache is used by default. **Optional** **Accepted Value:** true/false	
oracle.security.jps.pdp.sm.IdentityMaxCacheSize	Specifies the maximum number of users for which information is cached. When the maximum is reached, old records are dropped and reestablished when needed. **Optional** **Accepted Value:** number	
oracle.security.jps.pdp.sm.IdentityCacheEvictionPercentage	Specifies percentage of identities that must be evicted when cache has reached the maximum size. **Optional** **Accepted Value:** number indicating percentage	
oracle.security.jps.pdp.sm.IdentityCachedEntryTTL	Specifies time-to-live of an identity cache record. **Optional** **Accepted Value:** time in seconds	

Table A–8 (Cont.) Web Services Security Module Configuration Parameters

Parameter Name	Information	Console Name
oracle.security.jps.pdp.wssm.responseContext	Specifies whether to merge data from many AppContext responses into a single AppContext response. **Optional** **Accepted Values** ■ Merged ■ Unmerged (default value)	
oracle.security.jps.pdp.wssm.ssl.identityKeyStoreFileName	Defines the name of the Identity Key Store file where client certificates are stored for the Web Services Security Module. Used for SSL communications between the remote client and the Web Services Security Module. **Optional** **Accepted Value:** name of the Identity Key Store file	
oracle.security.jps.pdp.wssm.ssl.trustKeyStoreFileName	Defines the name of the Trust Key Store file in which CA certificates are stored. Used for SSL communications between the remote client and the Web Services Security Module. **Optional** **Accepted Value:** name of the Trust Key Store file	
oracle.security.jps.pdp.wssm.ssl.identityKeyStoreKeyAlias	Specifies the Identity Key alias used to identify the Web Services Security Module client certificate used for SSL communication between the Web Services Security Module and the remote client. Acepted value: Idenity key alias **Optional** **Accepted Value:** Identity Key alias	
oracle.security.jps.pdp.wssm.WSLoggingSoapHandlerEnabled	Enables the Web Services Security Module's EnvelopLoggingSOAPHandler, the web service SOAP message handler for logging. **Optional** **Accepted Values** ■ true ■ false (default value)	

A.2.3 Web Services Security Module on WebLogic Server

Table A–9 compiles the parameters to configure the Web Services Security Module on a WebLogic Server.

Table A–9 Web Services Security Module on WebLogic Configuration Parameters

Parameter Name	Information	Console Name
oracle.security.jps.pdp.wssm.WSServiceRegistryPortNumber	Defines the port on which the Web Services Security Module listens. **Mandatory** **Accepted Value:** port number	
oracle.security.jps.pdp.wssm.WSServiceRegistryHost	Defines the name of the server on which the Web Services Security Module is running. **Optional** **Accepted Value:** server name (default value is localhost)	
oracle.security.jps.pdp.wssm.Protocol	Defines the transport protocol used between the Policy Distribution Component client and server. **Optional** **Accepted Values** ■ https ■ http (default value)	
oracle.security.jps.pdp.wssm.WSServiceRegistryContextName	Specifies the context name for the Web service deployed on the WebLogic Server cache is being used. If not set, no identity cache is used by default. **Mandatory** **Accepted Value:** Ssmws	
oracle.security.jps.pdp.sm.IdentityCacheEnabled	Specifies whether the identity cache is enabled. Enabled by default. **Optional** **Accepted Value:** true (default)/false	
oracle.security.jps.pdp.sm.IdentityMaxCacheSize	Specifies the maximum size of the identity cache. **Optional** **Accepted Value:** number indicating size; default value is 20000	
oracle.security.jps.pdp.sm.IdentityCacheEvictionPercentage	Specifies the percentage of identities that will be removed when the identity cache has reached its maximum size. **Optional** **Accepted Value:** 20 percent	
oracle.security.jps.pdp.sm.IdentityCachedEntryTTL	Specifies the time-to-live (TTL) in seconds for an identity record in the identity cache. **Optional** **Accepted Value:** 3600 seconds (default)	

Table A–9 (Cont.) Web Services Security Module on WebLogic Configuration

Parameter Name	Information	Console Name
oracle.security.jps.pdp.wssm.responseContext	Specifies whether the AppContext is returned as a single response or a merged set of data from all the AppContext responses. **Optional** **Accepted Value:** Merged/Unmerged (default)	
oracle.security.jps.pdp.wssm.WSLoggingSoapHandlerEnabled	Enables the Web Services Security Module's EnvelopLoggingSOAPHandler, the web service SOAP message handler for logging. **Optional** **Accepted Values** ■ true ■ false (default value)	

A.2.4 RMI Security Module

Table A–10 compiles the parameters to configure the RMI Security Module embedded in either a JSE or a JEE container.

Note: Currently this configuration is for a standalone deployment. We need to add the Container based configuration later.

Table A–10 RMI Security Module Configuration Parameters

Parameter Name	Information	Console Name
oracle.security.jps.pdp.rmism.RMIRegistryPortNumber	Defines the port on which the RMI Security Module listens to the RMI server. **Mandatory** **Accepted Value:** port number.	
oracle.security.jps.pdp.rmism.UseSSL	Defines whether the SSL protocol is used for secure communication between the RMI Security Module and RMI server. **Optional** **Accepted Values** ■ true ■ false (default)	
oracle.security.jps.pdp.sm.IdentityCacheEnabled	Specifies whether the identity cache is being used. If not set, no identity cache is used by default. **Optional** **Accepted Value:** true/false	

Table A–10 (Cont.) RMI Security Module Configuration Parameters

Parameter Name	Information	Console Name
oracle.security.jps.pdp.sm.IdentityMaxCacheSize	Specifies the maximum number of users for which information is cached. When the maximum is reached, old records are dropped and reestablished when needed. **Optional** **Accepted Value:** number	
oracle.security.jps.pdp.sm.IdentityCacheEvictionPercentage	Specifies percentage of identities that must be evicted when cache has reached the maximum size. **Optional** **Accepted Value:** number representing percentage	
oracle.security.jps.pdp.sm.IdentityCachedEntryTTL	Specifies the time-to-live of an identity cache record. **Optional** **Accepted Value:** time in seconds	

A.2.5 WebLogic Server Security Module

Table A–11 compiles the parameters to configure the WebLogic Server (WLS) Security Module embedded in a JEE container. These parameters are used only when the WLS Security Module is configured to be used as a PEP.

- See Section 1.3.2, "The Policy Decision Point and the Policy Enforcement Point" for contextual information.
- See Section 9.4, "Securing WebLogic Server Resources" to enable the WebLogic Server Security Module.

Table A–11 WebLogic Server Security Module Configuration Parameters

Parameter Name	Information	Console Name
UndefinedApplicationEffect	Specifies the effect (GRANT, DENY) that the provider must return if an application is not defined in the policy store. **Optional** **Accepted Values** ■ permit (default) ■ abstain ■ deny	Set in the WebLogic Server Administration Console; values are saved to config.xml in the WebLogic domain

Table A–11 (Cont.) WebLogic Server Security Module Configuration Parameters

Parameter Name	Information	Console Name
NoApplicablePolicyEffect	Specifies the effect that the provider has to return if no applicable policies have been found. **Optional** **Accepted Values** ■ deny (default value represents a closed system) ■ abstain ■ permit (represents an open system)	Set in the WebLogic Server Administration Console; values are saved to config.xml in the WebLogic domain

A.2.6 WebLogic Server Security Module Discovery Mode

Table A–12 compiles the parameters to enable Discovery Mode. See Section 9.4.2, "Discovering WebLogic Server Resources" for more information.

Table A–12 WebLogic Server Discovery Mode Parameters

Parameter Name	Information	Console Name
oracle.security.jps.discoveryMode	By default, Discovery Mode is off. **Optional** **Accepted Values** ■ true ■ false	Only in `jps-config.xml`
oracle.security.jps.discoveredPolicyDir	Specifies the absolute path to the directory in which discovery results are defined. **Optional** (Mandatory when Discovery Mode is enabled) **Accepted Value:** absolute path to directory	Only in `jps-config.xml`
oracle.security.jps.discoveredResourceIsHierarchical	Specifies whether the resource is hierarchical. **Optional** **Accepted Values** ■ true ■ false	Only in `jps-config.xml`
oracle.security.jps.discoveredResourceNameDelimiter	Specifies the delimiter to separate the resource name. **Optional (Mandatory when resource is defined as Hierarchical)** **Accepted Value:** any valid resource name delimiter; when used with WLS SM and OSB SM, the value should be "/"	Only in `jps-config.xml`

A.3 PDP Proxy Client Configuration

This section contains information regarding configuration for the PDP Proxy Client available for the RMI and Web Services Security Module.

- Section A.3.1, "Web Services Security Module PDP Proxy Client"
- Section A.3.2, "RMI Security Module PDP Proxy Client"

A.3.1 Web Services Security Module PDP Proxy Client

Table A–13 compiles the parameters to configure the Web Services Security Module PDP Proxy Client.

Table A–13 Web Services Proxy Client Configuration Parameters

Parameter Name	Information	Console Name
oracle.security.jps.pdp.PDPTransport	Specifies the underlying protocol to be used by Multi-protocol Security Module to communicate with Oracle Entitlements Server. **Mandatory** **Accepted Values**: no default value; XACML is always available in the Web Services Security Module. ■ WS ■ RMI	
oracle.security.jps.pdp.proxy.PDPAddress	Specifies the host and port number of either the Web Services Security Module. For example, `http://dadvml0134:9015` **Optional** **Accepted Value:** a comma separated list of URIs (if more then one address is specified the first is conidered the primary, and the rest as backups)	
oracle.security.jps.pdp.proxy.RequestTimeoutMilliSecs	Defines the interval of time in which an authorization request times out when the remote PDP (RMI or Web Services Security Module) is not responding. **Optional** **Accepted Value:** time in milliseconds (default value is 10000)	
oracle.security.jps.pdp.proxy.FailureRetryCount	Specifies the number of attempts to make before attempting the alternate failover server. **Optional** **Accepted Value:** number (default value is 3)	
oracle.security.jps.pdp.proxy.FailbackTimeoutMilliSecs	Specifies the interval of time after which a failed primary server is tried again for failover. **Optional** **Accepted Value:** time in milliseconds (default value is 180000)	

Table A–13 (Cont.) Web Services Proxy Client Configuration Parameters

Parameter Name	Information	Console Name
oracle.security.jps.pdp.proxy.SynchronizationIntervalMilliSecs	Defines how often the PDP Proxy polls the PDP server in order to synchronize its state. For example, the interval is used to periodically check whether the authorization cache has to be flushed. **Optional** **Accepted Value:** time in milliseconds (default value is 60)	
oracle.security.jps.pdp.proxy.wssm.ssl.identityKeyStoreFileName	Defines the name of the Identity Key Store file where client certificates for the Web Services Security Module are stored. Used for SSL communication between a client and the Web Services Security Module. **Optional** **Accepted Value:** name of the Identity Key Store file	
oracle.security.jps.pdp.proxy.wssm.ssl.trustKeyStoreFileName	Defines the name of the Trust Key Store file where CA certificates for Web Services Security Module are stored. Used for SSL communication between a client and the Web Services Security Module. **Optional** **Accepted Value:** the name of the Trust Key Store file.	
oracle.security.jps.pdp.proxy.wssm.ssl.identityKeyStoreKeyAlias	Specifies the alias name of the Web Services client certificate. Used for SSL communication between a client and the Web Services Security Module. **Optional** **Accepted Value:** alias of the identity key store (if only one alias exists in the identity key store, no need to specify this value)	
oracle.security.jps.pdp.proxy.wssm.protocol	Defines the transport protocol used between the Policy Distribution Component client and server. **Optional** **Accepted Values** ■ https ■ http (default value)	

A.3.2 RMI Security Module PDP Proxy Client

Table A–14 compiles the parameters to configure the RMI Security Module PDP Proxy Client.

Table A–14 PDP RMI Proxy Client Configuration Parameters

Parameter Name	Information	Console Name
oracle.security.jps.pdp .PDPTransport	Specifies the underlying protocol to be used by Multi-protocol Security Module to communicate with Oracle Entitlements Server. **Mandatory** **Accepted Values:** no default value; XACML is always available in the RMI Security Module. ■ WS ■ RMI	
oracle.security.jps.pdp .proxy.PDPAddress	Specifies the host and port number of the RMI Security Module. For example, `rmi://localhost:9400` **Mandatory** **Accepted Value:** a comma separated list of URIs (if more then one address is specified the first is conidered the primary, and the rest as backups)	
oracle.security.jps.pdp .proxy.RequestTimeoutMilliSecs	Defines the interval of time in which an authorization request times out when the remote PDP (RMI or Web Services Security Module) is not responding. **Optional** **Accepted Value:** time in milliseconds (default value is 10000)	
oracle.security.jps.pdp .proxy.FailureRetryCount	Specifies the number of attempts to make before attempting the alternate failover server. **Optional** **Accepted Value:** number (default value is 3)	
oracle.security.jps.pdp .proxy.FailbackTimeoutMilliSecs	Specifies the interval of time after which a failed primary server is tried again for failover. **Optional** **Accepted Value:** time in milliseconds (default value is 180000)	
oracle.security.jps.pdp .proxy.SynchronizationIntervalMilliSecs	Defines how often the PDP Proxy polls the PDP server in order to synchronize its state. For example, the interval is used to periodically check whether the authorization cache has to be flushed. **Optional** **Accepted Value:** time in milliseconds (default value is 60)	

A.4 Policy Store Service Configuration

Table A–15 compiles the configuration parameters for the Policy Store Service.

Table A–15 Policy Store Service Configuration Parameters

Parameter Name	Information	Console Name
ldap.url	Defines the URL of the LDAP policy store. Valid in JEE and JSE applications and only applies to LDAP stores. **Mandatory** **Accepted Value:** URI of the LDAP policy store in the format `ldap://host:port`.	
max.search.filter.length	Defines the maximum length of a search filter. **Mandatory** **Accepted Value:** integer defining the maximum length of a search filter; for example, 1024	
oracle.security.jps.ldap.root.name	Defines the RDN format of the root node in the LDAP policy store. Valid in JEE and JSE applications. Applies to LDAP and database stores. **Mandatory** **Accepted Value:** root name of jps context; for example, `cn=jpsroot`.	
oracle.security.jps.farm.name	Defines the RDN format of the root node in the LDAP policy store. Valid in JEE and JSE applications. Applies to LDAP and database stores. **Mandatory** **Accepted Value:** farm name of the domain; for example, `cn=base_domain`.	
oracle.security.jps.policystore.resourcetypeenforcementmode	Controls the throwing of exceptions if any of the following checks fail: ■ Verify that if two resource types share the same permission class, that permission must be either ResourcePermission or extend AbstractTypedPermission, and this last resource type cannot be created. ■ Verify that all permissions have resource types defined, and that the resource matcher permission class and the permission being granted match. Valid in JEE and JSE applications. Applies to LDAP and database stores. **Optional** **Accepted Values** ■ strict (when any of the above checks fail, the system throws an exception and the operation is aborted) ■ lenient (default value; when any of the above checks fail, the system does not throw any exceptions, the operation continues without disruption, and any discrepancies encountered are logged)	

Table A–15 (Cont.) Policy Store Service Configuration Parameters

Parameter Name	Information	Console Name
bootstrap.security.principal.key	Defines the key for the password credentials to access the LDAP policy store, stored in the CSF store. Valid in JEE and JSE applications. Applies to LDAP and database stores. **Mandatory** **Accepted Value:** the key name of the credential; for example, `oes_sm_key`. The out-of-the-box value is `bootstrap`.	
bootstrap.security.principal.map	Defines the map for the password credentials to access the LDAP policy store, stored in the CSF store. Valid in JEE and JSE applications. Applies to LDAP and database stores. **Mandatory** **Accepted Value:** map name of the credential; for example, `oes_sm_map`. The default value is `BOOTSTRAP_JPS`.	
jdbc.driver	Defines the name of the JDBC driver. **Mandatory** **Accepted Value:** name of the JDBC driver.	
jdbc.url	Defines the JDBC driver connection URL. **Mandatory** **Accepted Value:** the JDBC driver connection URL.	

B

Configuring Attribute Retrievers Manually

As discussed in Section 1.3, "Overview of the Oracle Entitlements Server Architecture," the Policy Information Point (PIP) is a system entity that acts as a source for attribute values. Oracle Entitlements Server relies on an Attribute Retriever plug-in to get attribute values from one or more of these information stores. Predefined Attribute Retrievers are shipped with Oracle Entitlements Server. This chapter documents these predefined Attribute Retrievers and related configuration requirements. It contains the following sections.

- Understanding Predefined Attribute Retrievers
- Configuring the Predefined Attribute Retrievers
- Modifying jps-config.xml
- Setting Up PIP Connection Credentials
- Updating the Database Password

> **Note:** See the *Oracle Fusion Middleware Developer's Guide for Oracle Entitlements Server* for information on custom Attribute Retrievers.

B.1 Understanding Predefined Attribute Retrievers

Oracle Entitlements Server contains predefined Attribute Retrievers that are used to connect to, and retrieve attribute values from, Lightweight Directory Access Protocol (LDAP) data stores and relational database management systems (RDBMS). These predefined Attribute Retrievers can handle one or more attributes defined in the system without additional programming. They also contain a caching feature and failover.

- An in-memory cache mechanism is used to improve performance by reducing communications between Oracle Entitlements Server and the external repository. The cache holds up to 1000 entries and can be enabled for each individual attribute. The cache size is not configurable. If the limit is reached, cache items are removed randomly. Example B–2 illustrates the definition of an individual attribute with the `cached` and `ttl` properties.
- Repository failover can also be configured. When a call for an attribute is received, Oracle Entitlements Server checks whether the primary repository is active. If it is active, the value is retrieved. If the primary repository is not active, it has failed previously and the backup repository is active. In the latter case, Oracle Entitlements Server checks to see if it is time to switch back to the active repository (based on configuration). If it is time to switch back, the switch is made and the

value is retrieved from the primary repository. If the configured time has not yet passed, the value is retrieved from the active backup repository.

Note: If errors occur when retrieving values from the primary repository, Oracle Entitlements Server searches the backup repositories, trying them one by one until an active one is found.

See Section B.2.3, "Configuring Individual Attributes for Predefined Attribute Retrievers" for configuration information.

B.2 Configuring the Predefined Attribute Retrievers

Configuration information for these Attribute Retrievers is defined in the `jps-config.xml` configuration file. You must configure two types of information: attribute query information and repository connection information

- Repository connection information is used to connect to the data store and may include its location, JDBC driver and URL or LDAP URL (whichever is applicable) and the user/credential information. This connection information is related to a particular retriever instance. Repository connection information is defined in the `<serviceInstances>` section of `jps-config.xml` as illustrated in Example B–1.

Example B–1 Repository Connection Information Defined for Attribute Retriever

```
<serviceInstance name="policystore.rdbms" provider="policy.rdbms">
 <property name="jdbc.url"
  value="jdbc:oracle:thin:@scl58116.domainexample.com:1521:orcl"/>
 <property name="jdbc.driver" value="oracle.jdbc.driver.OracleDriver"/>
 <property name="bootstrap.security.principal.key" value="keyname"/>
 <property name="bootstrap.security.principal.map" value="mapname"/>
 <property name="oracle.security.jps.ldap.root.name" value="cn=jpsTestNode"/>
 <property name="oracle.security.jps.farm.name"
  value="cn=wcai_view_jing.atzsrg"/>
</serviceInstance>
```

Section B.2.1, "Configuring the LDAP Respository Attribute Retriever Parameters," Section B.2.2, "Configuring the Database Repository Attribute Retriever Parameters," and Section B.3, "Modifying jps-config.xml" contain information regarding a respository connection configuration.

Note: The instance must also be defined in the default `<jpsContexts>` section. See Example B–8, "Declaring the Predefined Attribute Retriever in jpsContext".

- Attribute query information is related to a particular attribute and includes its name, the name of the predefined Attribute Retriever used, the search query for retrieval (for example, a SQL query if the store is a relational database or an LDAP query if it's a directory), and any attribute caching information. Attribute query information is defined in the `<propertySets>` section of `jps-config.xml` as illustrated in Example B–2.

Example B–2 Attribute Query Information Defined for Attribute Retriever

```
<propertySet name="ootb.pip.attribute.age.based.on.myattr.ldap">
    <property name="ootb.pip.attr.type" value="OOTB_PIP_ATTRIBUTE"/>
```

```
    <property name="ootb.pip.ref" value="pip.service.ootb.ldap"/>
    <property name="name" value="oespipage_myattr"/>
    <property name="query" value="(cn=%MyAttr%)"/>
    <property name="cached" value="true"/>
    <property name="ttl" value="60"/>
</propertySet>
```

Section B.2.3, "Configuring Individual Attributes for Predefined Attribute Retrievers" and Section B.3, "Modifying jps-config.xml" contain information regarding an attribute query configuration.

Note: These predefined Attribute Retrievers can be configured with Oracle Database 11gR1, Oracle Internet Directory 11gR1, and Oracle Virtual Directory 11gR1.

The following sections contain information on the configuration parameters for each type of Attribute Retriever. As previously mentioned, these parameters are in the `jps-config.xml`, the configuration file (used by Java EE containers) located in the *$DOMAIN_HOME*`/config/fmwconfig` directory.

- Section B.2.1, "Configuring the LDAP Respository Attribute Retriever Parameters"
- Section B.2.2, "Configuring the Database Repository Attribute Retriever Parameters"
- Section B.2.3, "Configuring Individual Attributes for Predefined Attribute Retrievers"

B.2.1 Configuring the LDAP Respository Attribute Retriever Parameters

Table B–1 documents the parameters that must be defined when using the LDAP Attribute Retriever. See Example B–5, "Using the Predefined LDAP Attribute Retriever" and Example B–10, "Configuring LDAP Failover" for sample configuration code.

Table B–1 LDAP Attribute Retriever Parameters

Name	Usage
name	**Description:** The predefined Attribute Retriever's name **Mandatory** **Accepted Value:** String defining the Attribute Retriever service instance.
description	**Description:** A description of the predefined Attribute Retriever **Optional** **Accepted Value:** string
type	**Description:** The predefined Attribute Retriever's type **Manadatory** **Accepted Value:** LDAP_PIP

Table B–1 (Cont.) LDAP Attribute Retriever Parameters

Name	Usage
failed.server.retry.interval	**Description:** After communication with a primary repository has failed, this attribute defines the interval of time during which the backup repository is used before switching back to the primary repository. **Optional** **Accepted Value:** Takes a value equal to the number of seconds. Default value is 15.
bootstrap.security.principal.key	**Description:** Defines the key for the password credentials to access the LDAP policy store, stored in the CSF store. Valid in JEE and JSE applications. Applies to LDAP and database stores. See Section B.4, "Setting Up PIP Connection Credentials." **Mandatory** **Accepted Value:** key name of the credential; for example, `oes_sm_key`.
bootstrap.security.principal.map	**Description:** Defines the map for the password credentials to access the LDAP policy store, stored in the CSF store. Valid in JEE and JSE applications. Applies to LDAP and database stores. See Section B.4, "Setting Up PIP Connection Credentials." **Mandatory** **Accepted Value:** map name of the credential; for example, `oes_sm_map`.
ldap.url	**Description:** Defines the URL of the LDAP policy store. Valid in JEE and JSE applications and only applies to LDAP stores. **Mandatory** **Accepted Value:** URI of the LDAP policy store in the format `ldap://host:port`.

B.2.2 Configuring the Database Repository Attribute Retriever Parameters

Table B–2 documents the parameters that must be defined when using the RDBMS Attribute Retriever. See Example B–6, "Using the Predefined RDBMS Attribute Retriever with JDBC" and Example B–7, "Using the Predefined RDBMS Attribute Retriever with SQL" for sample configuration code.

Table B–2 RDBMS Attribute Retriever Parameters

Name	Usage
name	**Description:** The predefined Attribute Retriever's name **Mandatory** **Accepted Value:** String defining the Attribute Retriever service instance.
description	**Description:** A description of the predefined Attribute Retriever **Optional** **Accepted Value:** string

Table B–2 (Cont.) RDBMS Attribute Retriever Parameters

Name	Usage
type	**Description:** The predefined Attribute Retriever's type **Manadatory** **Accepted Value:** RDBMS_PIP
failed.server.retry.interval	**Description:** After the primary repository has failed, this attribute identifies the interval of time during which the backup repository is used before switching back to the primary repository. **Optional** **Accepted Value:** Takes a value equal to the number of seconds. Default value is 15.
bootstrap.security.principal.key	**Description:** Defines the key for the password credentials to access the database, stored in the CSF store. Valid in JEE and JSE applications. See Section B.4, "Setting Up PIP Connection Credentials." **Mandatory** **Accepted Value:** key name of the credential; for example, `oes_sm_key`.
bootstrap.security.principal.map	**Description:** Defines the map for the password credentials to access the database, stored in the CSF store. Valid in JEE and JSE applications. See Section B.4, "Setting Up PIP Connection Credentials." **Manadatory** **Accepted Value:** map name of the credential; for example, `oes_sm_map`.
jdbc.driver	**Description:** Location of the driver when using Java Database Connectivity (JDBC) API to connect to a database. **Mandatory:** When using JDBC API to connect to database. **Accepted Value:** `oracle.jdbc.driver.OracleDriver`, for example
jdbc.url	**Description:** Takes a URL that points to the database. **Mandatory:** When using JDBC API to connect to database. **Accepted Value:** A list of comma-delimited URLs. The first is treated as primary and so on. For example, `jdbc:oracle:thin:@scl58116.domainexample.com:1521:orcl`
datasource.jndi.name	**Description:** Data source JNDI name if you want the PIP instance working through data source rather than directly through JDBC. The data source scenario is supported on WebLogic Server and WebSphere Application Server only. **Mandatory:** If you want the PIP instance working through data source rather than directly through JDBC. **Accepted Value:** JNDI name of pre-defined data source object

B.2.3 Configuring Individual Attributes for Predefined Attribute Retrievers

Table B–3 documents the parameters to be defined for each attribute retrieved by the configured Attribute Retriever. See Example B–9, "Enabling an Attribute's Cache" for a sample configuration.

Table B–3 Configure Attributes to be Retrieved

Name	Usage
name	**Description:** The name of the attribute as defined in the policy store. When using the LDAP predefined Attribute Retriever, the attribute name defined for Oracle Entitlements Server must be the same as the attribute name defined in the LDAP store. Currently, there is no name mapping functionality. **Mandatory** **Accepted Value:** Attribute name
query	**Description:** The SQL command or LDAP filter used for the query. Users can use a built-in and custom attributes in the query string. For example, the built-in attribute `sys_user` can be used to define a query such as *select age from customers where name=%sys_user%;*. The token is automatically replaced by its value before sending the query to the data store. Bi-directional dependency (where, for example, AttributeA's query string contains AttributeB and AttributeB's query string contains AttributeA) can also be detected and, in such cases, an exception is thrown. **Mandatory** **Accepted Value:** SQL command or LDAP filter.
search.base	**Description:** The LDAP search base. **Mandatory:** For LDAP only. **Accepted Value:** The DN of the search base object.
ttl	**Description:** The time-to-live in seconds of any cached attribute values when cached is enabled. **Optional** **Accepted Value:** Any integer; default value is 60 seconds if cache is enabled.
cached	**Description:** Enables the caching of attribute values. **Optional** **Accepted Value:** Default value is false.
ootb.pip.attr.type	**Description:** Should be set to OOTB_PIP_ATTRIBUTE. **Mandatory** **Accepted Value:** OOTB_PIP_ATTRIBUTE.
ootb.pip.ref	**Description:** Should be set to an OOTB PIP instance. **Mandatory** **Accepted Value:** The PIP service instance name defined in the `<serviceInstance>` section of `jps-config.xml`

B.3 Modifying jps-config.xml

To configure the predefined Attribute Retriever in `jps-config.xml`, modify the elements as described in each example in this section. Example B–3 is a sample `jps-config.xml` file. The examples following it illustrate the modifications that can be made.

Example B–3 Sample jps-config.xml File

```
<?xml version="1.0"?>

<jpsConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="
```

```
http://xmlns.oracle.com/oracleas/schema/jps-config-11_0.xsd">

   <property name="oracle.security.jps.jaas.mode" value="off"/>
   <property name="oracle.security.jps.enterprise.user.class"
    value="weblogic.security.principal.WLSUserImpl"/>
   <property name="oracle.security.jps.enterprise.role.class"
    value="weblogic.security.principal.WLSGroupImpl"/>

<propertySets>
<!-- These are the global authenticated role properties -->
  <propertySet name="authenticated.role.properties">
     <property name="authenticated.role.name" value="authenticated-role"/>
     <property name="authenticated.role.uniquename" value="authenticated-role"/>
     <property name="authenticated.role.description"
        value="This is the authenticated role used by identity store
        service instance."/>
  </propertySet>

<!-- attribute defined for ldap retriever -->
  <propertySet name="ootb.pip.attribute.age.ldap">
     <property name="ootb.pip.attr.type" value="OOTB_PIP_ATTRIBUTE"/>
     <property name="ootb.pip.ref" value="pip.service.ootb.ldap"/>
     <property name="name" value="oespipage"/>
     <property name="query" value="(cn=%SYS_USER%)"/>
     <property name="cached" value="true"/>
     <property name="ttl" value="60"/>
  </propertySet>

  <propertySet name="ootb.pip.attribute.age.based.on.myattr.ldap">
     <property name="ootb.pip.attr.type" value="OOTB_PIP_ATTRIBUTE"/>
     <property name="ootb.pip.ref" value="pip.service.ootb.ldap"/>
     <property name="name" value="oespipage_myattr"/>
     <property name="query" value="(cn=%MyAttr%)"/>
     <property name="cached" value="true"/>
     <property name="ttl" value="60"/>
  </propertySet>

  <propertySet name="ootb.pip.attribute.gender.ldap">
     <property name="ootb.pip.attr.type" value="OOTB_PIP_ATTRIBUTE"/>
     <property name="ootb.pip.ref" value="pip.service.ootb.ldap"/>
     <property name="name" value="oespipgender"/>
     <property name="query" value="(oespipage=%oespipage%)"/>
     <property name="cached" value="true"/>
     <property name="ttl" value="60"/>
  </propertySet>

<!-- attribute defined for rdbms retriever -->
  <propertySet name="ootb.pip.attribute.age.rdbms">
     <property name="ootb.pip.attr.type" value="OOTB_PIP_ATTRIBUTE"/>
     <property name="ootb.pip.ref" value="pip.service.ootb.db"/>
     <property name="name" value="oespipage"/>
     <property name="query" value="select oespipage
         from pip_info_store where username=%SYS_USER%"/>
     <property name="cached" value="true"/>
     <property name="ttl" value="60"/>
  </propertySet>

  <propertySet name="ootb.pip.attribute.age.based.on.myattr.rdbms">
     <property name="ootb.pip.attr.type" value="OOTB_PIP_ATTRIBUTE"/>
     <property name="ootb.pip.ref" value="pip.service.ootb.db"/>
```

```
      <property name="name" value="oespipage_myattr"/>
      <property name="query" value="select oespipage
          as oespipage_myattr from pip_info_store where username=%MyAttr%"/>
      <property name="cached" value="true"/>
      <property name="ttl" value="60"/>
    </propertySet>

    <propertySet name="ootb.pip.attribute.gender.rdbms">
      <property name="ootb.pip.attr.type" value="OOTB_PIP_ATTRIBUTE"/>
      <property name="ootb.pip.ref" value="pip.service.ootb.db"/>
      <property name="name" value="oespipgender"/>
      <property name="query" value="select oespipgender
           from pip_info_store where oespipage=%oespipage%"/>
       <property name="cached" value="true"/>
      <property name="ttl" value="60"/>
    </propertySet>
  </propertySets>

  <serviceProviders>

    <serviceProvider type="CREDENTIAL_STORE" name="credstoressp"
         class="oracle.security.jps.internal.credstore.ssp.
                SspCredentialStoreProvider">
      <description>SecretStore-based CSF Provider</description>
    </serviceProvider>

    <serviceProvider class="oracle.security.jps.az.
          internal.runtime.provider.PIPServiceProvider"
          name="pip.service.provider" type="PIP"/>

    <serviceProvider type="POLICY_STORE" name="policy.rdbms"
          class="oracle.security.jps.internal.policystore.
          OPSSPolicyStoreProvider">
      <property name="policystore.type" value="DB_ORACLE"/>
      <description>DBMS based PolicyStore</description>
    </serviceProvider>

    <serviceProvider name="pdp.service.provider" type="PDP"
           class="oracle.security.jps.az.internal.
           runtime.provider.PDPServiceProvider">
        <description>OPSS Runtime PDP Service Provider</description>
    </serviceProvider>

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

    <serviceProvider name="jaas.login.provider" type="LOGIN"
           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>

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

```
</serviceProvider>

<serviceProvider type="POLICY_STORE" name="policy.oid"
        class="oracle.security.jps.internal.
        policystore.ldap.LdapPolicyStoreProvider">
    <description>LDAP-based PolicyStore</description>
    <property name="policystore.type" value="OID"/>
    <property name="connection.pool.maxsize" value="30"/>
    <property name="connection.pool.provider.type" value="idmpool"/>
  </serviceProvider>

  <serviceProvider type="AUDIT" name="audit.provider"
       class="oracle.security.jps.internal.audit.AuditProvider">
      <description>Audit Service</description>
  </serviceProvider>
</serviceProviders>

<serviceInstances>

  <serviceInstance name="credstore" provider="credstoressp" location="./">
        <description>File Based Credential Store Service Instance</description>
  </serviceInstance>

  <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="./user-data.xml"/>
<!-- This property set defines the authenticated role -->
    <propertySetRef ref="authenticated.role.properties"/>
  </serviceInstance>
  <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"/>
    <property name="debug" value="true"/>
    <property name="addAllRoles" value="true"/>
  </serviceInstance>

  <serviceInstance name="policystore.rdbms" provider="policy.rdbms">
      <property name="jdbc.url"
         value="jdbc:oracle:thin:@scl58116.domainexample.com:1521:orcl"/>
      <property name="jdbc.driver" value="oracle.jdbc.driver.OracleDriver"/>
      <property name="bootstrap.security.principal.key" value="keyname"/>
      <property name="bootstrap.security.principal.map" value="mapname"/>
      <property name="oracle.security.jps.ldap.root.name"
         value="cn=jpsTestNode"/>
      <property name="oracle.security.jps.farm.name"
         value="cn=wcai_view_jing.atzsrg"/>
  </serviceInstance>

  <serviceInstance name="policystore.rdbms.ds" provider="policy.rdbms">
    <property name="oracle.security.jps.ldap.root.name"
        value="cn=jpsTestNode"/>
    <property name="oracle.security.jps.farm.name"
        value="cn=wcai_view_jing.atzsrg"/>
    <property value="atzsrgds" name="datasource.jndi.name"/>
  </serviceInstance>
```

```
    <serviceInstance name="pdp.service" provider="pdp.service.provider">
       <property name="oracle.security.jps.runtime.pd.client.sm_name"
             value="${atzsrg.pdp.configuration_id}"/>
       <property name="oracle.security.jps.pdp.
             AuthorizationDecisionCacheEnabled" value="true"/>
       <property name="oracle.security.jps.pdp.
             AuthorizationDecisionCacheEvictionCapacity" value="500"/>
       <property name="oracle.security.jps.pdp.
             AuthorizationDecisionCacheEvictionPercentage" value="10"/>
       <property name="oracle.security.jps.pdp.
             AuthorizationDecisionCacheTTL" value="60"/>
       <property name="oracle.security.jps.ldap.
             policystore.refresh.interval" value="30000"/>
       <property name="oracle.security.jps.policystore.
             refresh.purge.timeout" value="600000"/> <!-- 10 minutes -->
       <property name="loading_attribute_backward_compatible" value="false"/>
  <!-- Properties for controlled mode PD -->
       <property name="oracle.security.jps.runtime.
             pd.client.policyDistributionMode" value="non-controlled"/>
       <property name="oracle.security.jps.runtime.
             instance.name" value="${atzsrg.pdp.instance_name}"/>
    </serviceInstance>

    <serviceInstance name="policystore.oid" provider="policy.oid">
      <property name="max.search.filter.length" value="4096"/>
      <property name="bootstrap.security.principal.key" value="keyname"/>
      <property name="bootstrap.security.principal.map" value="mapname"/>
      <property name="ldap.url" value="ldap://scl58126.domainexample.com:3060"/>
      <property name="oracle.security.jps.ldap.root.name"
         value="cn=jpsTestNode"/>
      <property name="oracle.security.jps.farm.name"
         value="cn=wcai_view_jing.atzsrg"/>
      <property name="oracle.security.jps.policystore.resourcetypeenforcementmode"
         value="Lenient"/>
    </serviceInstance>

    <serviceInstance name="policystore.xml" provider="policy.xml"
                location="./system-jazn-data.xml"/>

    <serviceInstance name="user.authentication.loginmodule"
                provider="jaas.login.provider">
           <description>User Authentication Login Module</description>
           <property name="loginModuleClassName"
                value="oracle.security.jps.internal.
                jaas.module.authentication.JpsUserAuthenticationLoginModule"/>
           <property name="jaas.login.controlFlag" value="REQUIRED"/>
    </serviceInstance>

    <serviceInstance name="user.assertion.loginmodule"
                provider="jaas.login.provider">
           <description>User Assertion Login Module</description>
           <property name="loginModuleClassName"
                value="oracle.security.jps.internal.
                jaas.module.assertion.JpsUserAssertionLoginModule"/>
           <property name="jaas.login.controlFlag" value="REQUIRED"/>
    </serviceInstance>

    <serviceInstance name="pip.service.ootb.ldap" provider="pip.service.provider">
           <property name="type" value="LDAP_PIP"/>
```

```
        <property name="ldap.url"
             value="ldap://scl58126.domainexample.com:3060"/>
        <property name="bootstrap.security.principal.key" value="keyname"/>
        <property name="bootstrap.security.principal.map" value="mapname"/>
        <property name="search.base" value="cn=pip_info_store,
             cn=wcai_view_jing.atzsrg,cn=JPSContext,cn=jpsTestNode"/>
        <property name="failed.server.retry.interval" value="10"/>
 </serviceInstance>
<!-- JPS Audit Service Instance-->
 <serviceInstance name="audit" provider="audit.provider">
        <property name="audit.filterPreset" value="None"/>
        <property name="audit.maxDirSize" value ="0"/>
        <property name="audit.maxFileSize" value ="104857600"/>
        <property name="audit.loader.jndi" value="jdbc/AuditDB"/>
        <property name="audit.loader.interval" value="15" />
        <property name="audit.loader.repositoryType" value="File" />
 </serviceInstance>

 <serviceInstance name="pip.service.ootb.db" provider="pip.service.provider">
        <property name="type" value="RDBMS_PIP"/>
        <property name="jdbc.url"
             value="jdbc:oracle:thin:@scl58116.domainexample.com:1521:orcl"/>
        <property name="jdbc.driver" value="oracle.jdbc.driver.OracleDriver"/>
        <property name="bootstrap.security.principal.key" value="keyname"/>
        <property name="bootstrap.security.principal.map" value="mapname"/>
        <property name="failed.server.retry.interval" value="10"/>
 </serviceInstance>

 <serviceInstance name="pip.service.ootb.db.ds" provider="pip.service.provider">
        <property name="type" value="RDBMS_PIP"/>
        <property value="atzsrgds" name="datasource.jndi.name"/>
        <property name="failed.server.retry.interval" value="10"/>
 </serviceInstance>

</serviceInstances>

    <jpsContexts default="default">
        <jpsContext name="default">
            <serviceInstanceRef ref="policystore.oid"/>
            <serviceInstanceRef ref="pdp.service"/>
            <serviceInstanceRef ref="audit"/>
            <serviceInstanceRef ref="idstore.xml"/>
            <serviceInstanceRef ref="idstore.loginmodule"/>
            <serviceInstanceRef ref="pip.service.ootb.ldap"/>
            <serviceInstanceRef ref="pip.service.ootb.db"/>
        </jpsContext>
        <jpsContext name="smsec">
            <serviceInstanceRef ref="credstore"/>
        </jpsContext>

    </jpsContexts>

</jpsConfig>
```

Example B–4 illustrates how the `serviceProvider` element defines the use of a predefined Attribute Retriever by defining the internal Oracle Entitlements Server class.

Example B–4 Declaring the Predefined Attribute Retriever

```
<serviceProvider
  class="oracle.security.jps.az.internal.runtime.provider.PIPServiceProvider"
  name="pip.service.provider" type="PIP"/>
```

The following examples illustrate how to modify the `serviceInstance` element for the predefined Attribute Retriever being used.

- Example B–5, "Using the Predefined LDAP Attribute Retriever"
- Example B–6, "Using the Predefined RDBMS Attribute Retriever with JDBC"
- Example B–7, "Using the Predefined RDBMS Attribute Retriever with SQL"
- Example B–8, "Declaring the Predefined Attribute Retriever in jpsContext"
- Example B–9, "Enabling an Attribute's Cache"
- Example B–10, "Configuring LDAP Failover"

Example B–5 illustrates how to modify the `serviceInstance` element when using the predefined LDAP Attribute Retriever.

Example B–5 Using the Predefined LDAP Attribute Retriever

```
<serviceInstance name="pip.service.ootb.ldap" provider="pip.service.provider">
   <property name="type" value="RDBMS_PIP"/>
   <property name="ldap.url" value="ldap://dadvmg0065.domainexample.com:3080"/>
   <property name="bootstrap.security.principal.key" value="keyname"/>
   <property name="bootstrap.security.principal.map" value="mapname"/>
   <property name="failed.server.retry.interval" value="10"/>
</serviceInstance>
```

The following two examples illustrate how to modify the `serviceInstance` element when using the predefined RDBMS Attribute Retriever. Example B–6 is when using Java Database Connectivity (JDBC) API.

Example B–6 Using the Predefined RDBMS Attribute Retriever with JDBC

```
<serviceInstance name="pip.service.ootb.db" provider="pip.service.provider">
    <property name="type" value="RDBMS_PIP"/>
    <property name="jdbc.url"
     value="jdbc:oracle:thin:@scl58116.domainexample.com:1521:orcl"/>
    <property name="jdbc.driver" value="oracle.jdbc.driver.OracleDriver"/>
    <property name="bootstrap.security.principal.map" value="mapname"/>
    <property name="bootstrap.security.principal.key" value="keyname"/>
    <property name="failed.server.retry.interval" value="10"/>
</serviceInstance>
```

Example B–7 is when using a SQL database.

Example B–7 Using the Predefined RDBMS Attribute Retriever with SQL

```
<serviceInstance name="pip.service.ootb.db" provider="pip.service.provider">
    <property name="type" value="RDBMS_PIP"/>
    <property name="datasource.jndi.name" value="DB_RAC"/>
    <property name="failed.server.retry.interval" value="10"/>
</serviceInstance>
```

Example B–8 illustrates how to declare the predefined Attribute Retriever reference in the `jpsContext` element. This sample defines a predefined RDBMS Attribute Retriever.

Example B–8 Declaring the Predefined Attribute Retriever in jpsContext

```
  <jpsContext name="default">
       <serviceInstanceRef ref="policystore.db"/>
       <serviceInstanceRef ref="pdp.service"/>
       <serviceInstanceRef ref="audit"/>
       <serviceInstanceRef ref="idstore.xml"/>
       <serviceInstanceRef ref="idstore.loginmodule"/>
       <serviceInstanceRef ref="pip.service.ootb.db"/>
</jpsContext>
```

Example B–9 illustrates how to configure the caching of a specific attribute value. Caching is enabled per attribute. In this example, the cache record is deleted after 60 seconds.

Example B–9 Enabling an Attribute's Cache

```
<propertySet name="ootb.pip.attribute.gender.ldap">
        <property name="ootb.pip.attr.type" value="OOTB_PIP_ATTRIBUTE"/>
        <property name="ootb.pip.ref" value="pip.service.ootb.ldap"/>
        <property name="name" value="oespipgender"/>
        <property name="query" value="(oespipage=%oespipage%)"/>
        <property name="cached" value="true"/>
        <property name="ttl" value="60"/>
</propertySet>
```

Example B–10 illustrates how to configure the failover behavior. In this example, the primary connection is `ldap://dadvmg0065:3080` and the backup connection is `ldap://scl58123:3060`. The failed server retry interval is 10 seconds.

Example B–10 Configuring LDAP Failover

```
<serviceInstance name="pip.service.ootb.ldap" provider="pip.service.provider">
      <property name="type" value="LDAP_PIP"/>
      <property name="ldap.url"
         value="ldap://dadvmg0065:3080,ldap://scl58123:3060"/>
      <property name="bootstrap.security.principal.key" value="keyname"/>
      <property name="bootstrap.security.principal.map" value="mapname"/>
      <property name="failed.server.retry.interval" value="10"/>
</serviceInstance>
```

B.4 Setting Up PIP Connection Credentials

As documented in Table B–1, " LDAP Attribute Retriever Parameters" and Table B–2, " RDBMS Attribute Retriever Parameters", the `bootstrap.security.principal.key` and `bootstrap.security.principal.map` parameters define the key and the map (respectively) to access the data store. Oracle Entitlements Server ships with `oesPassword.sh` which sets these LDAP and database connection credentials in the bootstrap credential store. The tool is located in the *$OES_SM_INSTANCE_DIRECTORY*`/bin/` directory. Use the following command to run it.

```
./oesPassword.sh -setpass
```

It prompts for the security principal key name, the security principal map name, the username and associated password.

B.5 Updating the Database Password

For security reasons, the passwords configured for databases are periodically changed; thus, the Oracle Entitlements Server components (Administration Server and Security Modules) will need to be updated with the new password. The following information documents how this can be done.

- The Administration Server uses a WebLogic Server data source to provide database access and database connection management. To change the password, update the data source configuration with the new password using the WebLogic Server console. No additional steps are required.
- Security Modules may connect to a particular database in either of the following ways:
  - The Security Module will connect to the database used as a Policy Store when distributing policies in either the controlled-pull, non-controlled or mixed distribution modes.
  - The Security Module can connect to any relational database management system (RDBMS) when retrieving attributes from an attribute repository using attribute retrievers. (This connection is not limited to the default Policy Store.)

  Both cases can use either the WebLogic Server data source configuration or JDBC properties directly. In cases when connecting via the WLS data source configuration, update the data source configuration with the new password using the WebLogic Server console. No additional steps are required. In cases when connecting via the JDBC properties directly, use the oesPassword utility as discussed in Section B.4, "Setting Up PIP Connection Credentials." The map and key for the credential is defined in `jps-config.xml`.

C

Managing Advanced Policies with WLST

The WebLogic Scripting Tool (WLST) is a command-line scripting interface that system administrators and operators use to monitor and manage WebLogic Server instances and domains. This Appendix documents the commands to manage advanced policies with Oracle Entitlements Server using the WSLT. It contains the following sections.

- Using the WebLogic Scripting Tool with Oracle Entitlements Server
- Using the WLST Commands
- Creating Policy with a Script

C.1 Using the WebLogic Scripting Tool with Oracle Entitlements Server

The WLST scripting environment is based on Jython, the Java scripting interpreter. By following the Jython language syntax, Oracle Entitlements Server extends the scripting language to allow for management of advanced policies. The new scripts are supported on the WebLogic Application Server, the WebSphere Application Server and the JBossApplication Server in both offline and online modes. The following links offer more information on the WLST and related subjects.

- `http://www.jython.org/`
- *Oracle Fusion Middleware Oracle WebLogic Scripting Tool*

The following sections in this book document the general process for creating a simple policy and adding advanced elements to it.

- Section 4.2, "Defining an Authorization Policy And Its Components"
- Section 4.3, "Adding Fine-Grained Elements to an Authorization Policy"

C.2 Using the WLST Commands

This section describes the WLST commands in both online mode. Offline mode is not supported.

- createApplicationPolicy
- updateResourceType
- updateResource
- createPolicy
- updatePolicy
- deletePolicy

- listPolicies
- createAttribute
- updateAttribute
- deleteAttribute
- listAttributes
- createFunction
- updateFunction
- deleteFunction
- listFunctions
- getFunction

C.2.1 createApplicationPolicy

Create an Application (also referred to as an Application Policy) using the specified name.

Interactive Mode

```
createApplicationPolicy(appStripe="<appStripeName>")
```

Script Mode

```
./wlst.sh createAdvancedPolicy.py -appStripe <appStripeName>
```

Arguments

- appStripeName – name of the Application Policy to be created.

C.2.2 updateResourceType

Update the specified Resource Type with the new properties.

Interactive Mode

```
updateResourceType(appStripe="<appStripeName>",
  resourceTypeName="<resourceTypeName>",
  displayName="<displayName>", description="<description>',
  allowedActions="<actions>', delimiter="<delimiter>',
  attributes="<attributes>', provider="<resTypeProvider>',
  matcher="<matcherClass>', hierarchicalResource="<hierarchicalResource>",
  resourceNameDelimiter="<resourceNameDelimiter>")
```

Script Mode

```
./wlst.sh updateResourceType.py -appStripe <appStripeName>
 -resourceTypeName <resourceTypeName> [-displayName <displayName>]
 [-description <description>] [-allowedActions <actions>]
 [-delimiter <delimiter>]
 [-attributes <attributes>] [-provider <resTypeProvider>]
 [-matcher <matcherClass>]
 [-hierarchicalResource <hierarchicalResource>
 [-resourceNameDelimiter <resourceNameDelimiter>]
```

Arguments

- appStripeName – The Application (also referred to as an Application Policy) name. Required.
- resourceTypeName – Name of the Resource Type to be updated. Required.
- displayName – Display name for the Resource Type. Optional.
- description – Short description of the Resource Type. Optional.
- actions - A comma-separated list of the allowed action(s) to be added or removed from the Resource Type. Actions prefixed with a dash (-) will be removed from the list; those without the prefix will be added to the list. Optional.
- delimiter – The delimiter used in the actions list. If unspecified, a comma is the default (`,`). Optional.
- attributes – A comma-separated list of attribute name(s) to be added or removed from the attribute set of the Resource Type. Attribute names prefixed with a dash (-) will be removed from the list; those without the prefix will be added to the list. Optional. Can be null.
- resTypeProvider – Provider class name for this Resource Type. Optional. Can be null.
- matcherClass – Matcher class name for this Resource Type. If unspecified, the default matcher class used is `oracle.security.jps.ResourcePermission`. Optional. Can be null.
- hierarchicalResource - Flag showing whether the Resource Type supports hierarchical resources. It is an optional parameter with a default value of false. Can be null.
- resourceNameDelimiter - Character to be used as a delimiter in names of hierarchical resources of this Resource Type. It is an optional parameter with a default value of a forward slash (`/`). A value should not be provided if the `hierarchicalResource` parameter is not given a value of true.

C.2.3 updateResource

Update the specified Resource with the new properties.

Interactive Mode

```
updateResource (appStripe="<appStripeName>", resourceName="<resourceName>",
 type="<resourceType>", displayName="<displayName>", description="<description>",
 attributes="<attributes>")
```

Script Mode

```
./wlst.sh updateResource.py -appStripe <appStripeName>
 -resourceName <resourceName> -type <resourceType> [-displayName <displayName>]
 [-description <description>] [-attributes <attributes>]
```

Arguments

- appStripeName – The Application (also referred to as an Application Policy) name. Required.
- resourceName – Name of Resource to be updated. Required.
- type – Name of the Resource Type associated with the Resource. Required.
- displayName – Display name for the Resource Type. Optional.

- description – Short description of the Resource Type. Optional.
- attributes – A semi-colon separated list of attribute(s) to be added or removed from the Resurce's attribute set. Optional. The following rules govern how to write an attribute value.
  1. To add a single-valued attribute, the format is <attribute name>:<value>
  2. To add a multi-valued attribute, the format is <attribute name>:<value 1>,<value 2>,...,<value n>
  3. To remove an attribute, the format is -<attribute name>

C.2.4 createPolicy

Create a new Policy within the specified Application.

Interactive Mode

```
createPolicy (appStripe="<appStripeName>", policyName="<policyName>",
 [displayName="<displayName>"], [description="<description>"],
 ruleExpression="<ruleExpression> {entitlements="<entitlements>"|
 resourceActions="<resActions>"},
 {principals="<principals>"|-codeSource="<codeSource>"}",
 [obligations="<obligations>"], [semantic="<semantic>"]
```

Script Mode

```
./wlst.sh createPolicy.py -appStripe <appStripeName>
 -policyName <policyName> [-displayName <displayName>]
 [-description <description>] -ruleExpression <ruleExpression>
 {-entitlements <entitlements>|-resourceActions <resActions>}
 {-principals <principals>|-codeSource <codeSource>}
 [-obligations <obligations>] [-semantic <semantic>]
```

Arguments

- appStripeName – The Application (also referred to as an Application Policy) name. Required.
- policyName – Name of policy to be created. Required.
- displayName – Display name for the Policy. Optional.
- description – Short description of the Policy. Optional.
- ruleExpression – A rule expression is evaluated for the policy decision. Required. The following rules govern how to write an expression.
  1. The ruleExpression begins with a name. The name is followed by a colon (`:`) which is then followed by an effect (GRANT/DENY). The effect is followed by a second colon (`:`) and an expression. The expression is comprised of functions and attributes. If there is no Condition to be evaluated for the ruleExpression, the second colon (`:`) and the expression following it are not required.
  2. All functions, including built-in ones are specified as a valid function name followed by a list of parameters enclosed within parentheses [`()`].
  3. Attributes and function names must be valid Java identifiers.
  4. Integer literal parameters begin with a digit between 1-9 followed by digits between 0-9. They can optionally be prefixed by a dash (`-`) for negative integers.
  5. String literals are enclosed within double quotes (`""`).

6. Date and Time literals are specified as GMT strings followed by the letter `d` and `t`, respectively. Case-insensitive.
7. Standard DataTypes supported are `string`, `int`, `boolean`, `date` and `time`.
8. Supported Operators: `&&` and `||` are used to combine two boolean expressions. ! is used on a boolean expression. `==` is used to check equality between two strings, integers, dates or times. `>`, `<`, `>=` and `<=` is used to compare two integers, dates or times.
9. All operators and functions besides `&&`, `||` and ! take only values and attributes as parameters. The others can take boolean expressions as well as parameters.

- entitlements – A comma-separated list of Permission Set entry name(s). Should be present only if resActions (Resource actions) is absent.
- resActions – A comma-separated list of Resource action(s). Should be present only if entitlements is absent. Resource actions are specified as an existing Resource Type followed by a colon (`:`). This is then followed by an existing Resource name, followed by a comma-separated list (within parentheses) of valid actions for the Resource. For example:

  ```
  resType1:res1(act1, act2),resType2:res2(act1),resType2:res3(act2)
  ```

- principals – A comma-separated list of Principals. Should be present only if codeSource is absent. Principals are specified as a name followed by a colon (`:`) and a fully-qualified class-name. For example:

  ```
  admin:com.example.myPrincipal, manager:com.example.myPrincipal
  ```

- codeSource – Code-source as a string. It should be present only if Principals are absent. Optional.
- obligations – A comma-separated list of Obligations. Optional. Obligations are specified as an Obligation name followed by a comma-separated list of Obligation attribute assignments within parentheses. If the assignment is a literal, it must be prefixed by a name followed by a colon (`:`). When assignment is an attribute, the name is optional. For example:

  ```
  ob1(attr1,str1:"a String"),ob2(a2:attr2)
  ```

- semantic – Either `and` or `or` can be chosen as the policy semantic. Optional.

C.2.5 updatePolicy

Update an existing Policy in the specified application.

Interactive Mode

```
updatePolicy (appStripe="<appStripeName>", policyName="<policyName>",
 [displayName="<displayName>"], [description="<description>"],
 [ruleExpression="<ruleExpression>"], [obligations="<obligations>"],
 [entitlements="<entitlements>"|resourceActions="<resActions>"],
 [principals="<principals>"|codeSource="<codeSource>"])
```

Script Mode

```
./wlst.sh updatePolicy.py -appStripe <appStripeName>
 -policyName <policyName> [-displayName <displayName>]
 [-description <description>] [-ruleExpression <ruleExpression>]
 [-obligations <obligations>]
```

```
[-entitlements <entitlements>|-resourceActions <resActions>]
[-principals <principals>|-codeSource <codeSource>]
```

Arguments

- appStripeName – The Application (also referred to as an Application Policy) name. Required.
- policyName – Name of policy to be updated. Required.
- displayName – Display name for the Policy. Optional.
- description – Short description of the Poicy. Optional.
- ruleExpression – A rule expression is evaluated for the policy decision. Required. The following rules govern how to write an expression.
  1. The ruleExpression begins with a name. The name is followed by a colon (`:`) which is then followed by an effect (GRANT/DENY). The effect is followed by a second colon (`:`) and an expression. The expression is comprised of functions and attributes. If there is no Condition to be evaluated for the ruleExpression, the second colon (`:`) and the expression following it are not required.
  2. All functions, including built-in ones are specified as a valid function name followed by a list of parameters enclosed within parentheses [`()`].
  3. Attributes and function names must be valid Java identifiers.
  4. Integer literal parameters begin with a digit between 1-9 followed by digits between 0-9. They can optionally be prefixed by a dash (`-`) for negative integers.
  5. String literals are enclosed within double quotes (`""`).
  6. Date and Time literals are specified as GMT strings followed by the letter `d` and `t`, respectively. Case-insensitive.
  7. Standard DataTypes supported are `string`, `int`, `boolean`, `date` and `time`.
  8. Supported Operators: `&&` and `||` are used to combine two boolean expressions. ! is used on a boolean expression. `==` is used to check equality between two strings, integers, dates or times. `>`, `<`, `>=` and `<=` is used to compare two integers, dates or times.
  9. All operators and functions besides `&&`, `||` and ! take only values and attributes as parameters. The others can take boolean expressions as well as parameters.
- entitlements – A comma-separated list of Permission Set entry name(s). Should be present only if resActions (Resource actions) is absent.
- resActions – A comma-separated list of Resource action(s). Should be present only if entitlements is absent. Resource actions are specified as an existing Resource Type followed by a colon (`:`). This is then followed by an existing Resource name, followed by a comma-separated list (within parentheses) of valid actions for the Resource. For example:

  ```
  resType1:res1(act1, act2),resType2:res2(act1),resType2:res3(act2)
  ```

- principals – A comma-separated list of Principals. Should be present only if codeSource is absent. Principals are specified as a name followed by a colon (`:`) and a fully-qualified class-name. For example:

  ```
  admin:com.example.myPrincipal, manager:com.example.myPrincipal
  ```

- codeSource – Code-source as a string. It should be present only if Principals are absent. Optional.
- obligations – A comma-separated list of Obligations. Optional. Obligations are specified as an Obligation name followed by a comma-separated list of Obligation attribute assignments within parentheses. If the assignment is a literal, it must be prefixed by a name followed by a colon (:). When assignment is an attribute, the name is optional. For example:

  ```
  ob1(attr1,str1:"a String"),ob2(a2:attr2)
  ```

C.2.6 deletePolicy

Remove an existing Policy from the specified application.

Interactive Mode

```
deletePolicy (appStripe="<appStripeName>", policyName="<policyName>")
```

Script Mode

```
./wlst.sh deletePolicy.py -appStripe <appStripeName> -policyName <policyName>
```

Arguments

- appStripeName – The Application (also referred to as an Application Policy) name. Required.
- policyName – Name of policy to be updated. Required.

C.2.7 listPolicies

List the Policies in the specified Application.

Interactive Mode

```
listPolicies (appStripe="<appStripeName>")
```

Script Mode

```
./wlst.sh listPolicies.py -appStripe <appStripeName>
```

Arguments

- appStripeName – The Application (also referred to as an Application Policy) name. Required.

C.2.8 createAttribute

Create a new Attribute in an application. Default attributes are not supported.

Interactive Mode

```
createAttribute (appStripe="<appStripeName>", attributeName="<attributeName>",
 [displayName="<displayName>"], [description="<description>"], type="<type>",
 category="<category>", [isSingle="<true/false>'], [values="<values>"])
```

Script Mode

```
./wlst.sh createAttribute.py -appStripe <appStripeName>
 -attributeName <attributeName> [-displayName <displayName>]
 [-description <description>] -type <type> -category  <category>
```

```
[-isSingle <true/false>][-values <values>]
```

Arguments

- appStripeName – The Application (also referred to as an Application Policy) name. Required.
- attributeName – Name of the attribute to be created. Required.
- displayName – Display name for the attribute. Optional.
- description – Short description of the attribute. Optional.
- type - Fully qualified class name of the type. Required.
- category - Attribute category. Either “Dynamic” or “Resource”. Required.
- isSingle – Whether the attribute is single-valued or multiple-valued. Default value is true. Optional.
- values – Initial attribute values. Optional.

C.2.9 updateAttribute

Update values of an Attribute already defined in an Application. Default attributes are not supported.

Interactive Mode

```
updateAttribute (-appStripe="<appStripeName>", attributeName="<attributeName>",
 [displayName="<displayName>"], [description="<description>"],
 [values="<values>"])
```

Script Mode

```
./wlst.sh updateAttribute.py -appStripe <appStripeName>
 -attributeName <attributeName> [-displayName <displayName>]
 [-description <description>] [-values <values>]
```

Arguments

- appStripeName – The Application (also referred to as an Application Policy) name. Required.
- attributeName – Name of the Attribute to be created. Required.
- displayName – Display name for the Attribute. Optional.
- description – Short description of the attribute. Optional.
- values - A comma-separated list of Attribute values. In the case where this value is an empty string, the Attribute will be updated to have no assigned value. Optional.

C.2.10 deleteAttribute

Delete the specified Attribute from the specified Application. Default Attributes are not supported.

Interactive Mode

```
deleteAttribute (appStripe="<appStripeName>", attributeName="<attributeName>",
 [cascade="<true/false>"])
```

Script Mode

```
./wlst.sh deleteAttribute.py -appStripe <appStripeName>
 -attributeName <attributeName>
 [-cascade <true/false>]
```

Arguments

- appStripeName – The Application (also referred to as an Application Policy) name. Required.
- attributeName – Name of the Attribute to be created. Required.
- cascade – A boolean value indicating whether the delete is to be cascaded. If unspecified, the value defaults to false. Optional.

C.2.11 listAttributes

List all Attributes defined in the given Application.

Online Mode

```
listAttributes (appStripe="<appStripeName>", [hideBuiltIn="<hideBuiltIn>"])
```

Offline Mode

```
./wlst.sh listAttributes.py -appStripe <appStripeName>
  [-hideBuiltIn <hideBuiltIn>]
```

Arguments

- appStripeName – The Application (also referred to as an Application Policy) name. Required.
- hideBuiltIn – A boolean value indicating whether to hide built-in attributes. The value defaults to true. Optional.

C.2.12 createFunction

Create a Function in the specified Application.

Interactive Mode

```
createFunction (appStripe="<appStripeName>", functionName="<functionName>",
 [displayName="<displayName>"], [description="<description>"],
 className="<className>", returnType="<returnType>",
 [paramTypes="<paramTypes>"])
```

Script Mode

```
./wlst.sh createFunction.py -appStripe <appStripeName>
 -functionName <functionName> [-displayName <displayName>]
 [-description <description>] -className <className>
 -returnType <returnType> [-paramTypes <paramTypes>]
```

Arguments

- appStripeName – The Application (also referred to as an Application Policy) name. Required.
- functionName – Name of the the Function being created. Required.
- displayName – Display name for the Function. Optional.
- description – Short description of the Function. Optional.

- className - Name of the class to which the function belongs. Required.
- returnType – Return type of the Function. Required.
- paramTypes – List the types of parameters concatenated using a comma (,). If unspecified, the function doesn't take any parameter as input. Optional.

C.2.13 updateFunction

Update the Function with the given values.

Interactive Mode

```
updateFunction (appStripe="<appStripeName>", functionName="<functionName>"
 [displayName="<displayName>"], [description="<description>"],
 [className="<className>"], [returnType="<returnType>"],
 [paramTypes="<paramTypes>"])
```

Script Mode

```
./wlst.sh updateFunction.py -appStripe <appStripeName>
 -functionName <functionName> [-displayName <displayName>]
 [-description <description>] [-className <className>]
 [-returnType <returnType>] [-paramTypes <paramTypes>]
```

Arguments

- appStripeName – The Application (also referred to as an Application Policy) name. Required.
- functionName – Name of the Function being updated. Required.
- displayName – Display name for the Function. Optional.
- description – Short description of the Function. Optional.
- className - Name of the class to which the Function belongs. Required.
- returnType – Return type of the Function. Required.
- paramTypes – List the types of parameters concatenated using a comma (,). If unspecified, parameter types of the function are not changed. If an empty string is the value, the Function is updated to take no input parameter. Optional.

C.2.14 deleteFunction

Delete the Function in the specified Application.

Interactive Mode

```
deleteFunction (appStripe="<appStripeName>", functionName="<functionName>",
 [cascade="<true/false>"])
```

Script Mode

```
./wlst.sh deleteFunction.py -appStripe <appStripeName>
 -functionName <functionName> [-cascade <true/false>]
```

Arguments

- appStripeName – The Application (also referred to as an Application Policy) name. Required.
- functionName – Name of the Function being updated. Required.

- cascade – A boolean value indicating whether the delete is to be cascaded. If unspecified, the value defaults to false. Optional.

C.2.15 listFunctions

List all Functions in the specified Application.

Interactive Mode

```
listFunctions (appStripe="<appStripeName>", [hideBuiltIn="<hideBuiltIn>"]
```

Script Mode

```
./wlst.sh listFunctions.py -appStripe <appStripeName>
 [-hideBuiltIn <hideBuiltIn>]
```

Arguments

- appStripeName – The Application (also referred to as an Application Policy) name. Required.
- hideBuiltIn – A boolean value indicating whether to hide built-in functions. The value defaults to true. Optional.

C.2.16 getFunction

Get details of the given Function in the specified Application.

Interactive Mode

```
getFunction (appStripe="<appStripeName>", functionName="<functionName>")
```

Script Mode

```
./wlst.sh getFunction.py -appStripe <appStripeName>
 -functionName <functionName>
```

Arguments

- appStripeName – The Application (also referred to as an Application Policy) name. Required.
- functionName – Name of the Function being obtained.

C.3 Creating Policy with a Script

Example C–1 illustrates how to create a single policy with a rule (containing one attribute comparison and one function invocation) and with an Obligation (containing two Obligation attributes).

Example C–1 Sample Script For Policy Creation

```
createApplicationPolicy(appStripe="jpsWebApp_wlst_test")
createResourceType(appStripe="jpsWebApp_wlst_test", resourceTypeName="resType1",
  provider="myProvider", matcher="myMatcher",
  allowedActions="action1,action2", delimiter=",")
createResourceType(appStripe="jpsWebApp_wlst_test", resourceTypeName="resType2",
  provider="myProvider", matcher="myMatcher",
  allowedActions="action1,action2", delimiter=",")
createResource(appStripe="jpsWebApp_wlst_test",name="res1",type="resType1")
createResource(appStripe="jpsWebApp_wlst_test",name="res2",type="resType2")
createResource(appStripe="jpsWebApp_wlst_test",name="res3",type="resType2")
```

```
createAttribute(appStripe="jpsWebApp_wlst_test", attributeName="attr1",
  type="date", category="dynamic")
createAttribute(appStripe="jpsWebApp_wlst_test", attributeName="attr2",
  type="time", category="dynamic")
createAttribute(appStripe="jpsWebApp_wlst_test", attributeName="abc",
  type="int", category="dynamic")
createAttribute(appStripe="jpsWebApp_wlst_test", attributeName="c",
  type="int", category="dynamic")
createAttribute(appStripe="jpsWebApp_wlst_test", attributeName="aString",
  type="string", category="dynamic")
createFunction(appStripe="jpsWebApp_wlst_test", functionName="boolFunction",
  className="com.example.myClassName",
  returnType="oracle.security.jps.service.policystore.info.OpssBoolean")
  paramTypes="oracle.security.jps.service.policystore.infoOpssString,
  oracle.security.jps.service.policystore.info.OpssInteger")
createPolicy(appStripe="jpsWebApp_wlst_test", policyName="policy1",
  ruleExpression="rule1:grant: !(abc > -1162) && (c==2)&& (\"a String\" ==
  aString)", resourceActions="resType1:res1(act1,
  act2),resType2:res2(act1),resType2:res3(act2)",
  principals="admin:com.example.MyPrincipal", obligations="ob3(attr1,str1:\"a
  String\"),ob4(a2:attr2)", semantic="or"
updatePolicy(appStripe="jpsWebApp_wlst_test", policyName="policy1",
  ruleExpression="rule3:deny:boolFunction(\"abc\",c)",
  resourceActions="-resType2:res3(act2),resType2:res3(act1)",
  obligations="-ob1,email(addr:\"me@mycompany.com\")")
listPolicies(appStripe="jpsWebApp_wlst_test")
deletePolicy(appStripe="jpsWebApp_wlst_test", policyName="policy1")
```

Index

A

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

W

X