Oracle ® Entitlements Server 10g (10.1.4.3)

Integration Guide

September 2008

Integration Guide, Oracle $^{®}$ Entitlements Server 10g (10.1.4.3)

Introduction

This section includes the following topics:

> **Note:** Oracle Entitlements Server was previously known as BEA Aqualogic Enterprise Security. Some items, such as schema objects, paths, and so on may still use the term "ALES."

- "Document Scope and Audience" on page 1-1
- "Guide to this Document" on page 1-1
- "Related Documentation" on page 1-2

Document Scope and Audience

This document primarily intended for administrators who are responsible for configuring Oracle Entitlements Server components, integrating its components into application environments, and securing application resources.

The topics in this document are relevant during the staging, production deployment, and production use phases of a software project. For links to other documentation and resources, see "Related Documentation" on page 1-2.

Guide to this Document

This document is organized as follows:

- "Securing Web Servers" on page 2-1

- "Securing WebLogic Servers" on page 3-1
- "Securing WebLogic Workshop Applications" on page 4-1
- "Securing Oracle Data Service Integrator" on page 5-1
- "Securing WebLogic Portal Applications" on page 6-1
- "Storing and Versioning Policy with Oracle Enterprise Repository" on page 7-1
- "Securing Oracle Service Bus Runtime Resources" on page 8-1
- "OES Adapter for Sun Identity Manager" on page 1-1

Related Documentation

Additional documents are available as described below:

- Getting Started Tutorials—Provides tutorials on installing OES and securing application resources.
- Introduction —Provides overview, conceptual, and architectural information for Oracle Entitlements Server.
- Administration Server Installation Guide—Describes installation of the Administration Server.
- SSM Installation and Configuration Guide—Describes installing and configuring Security Service Modules.
- Policy Managers Guide—Defines the policy model and describes how to generate, import and export policy data.
- Administration Reference—Describe how to write custom security provider extensions, describes how to extend the AuditContext interface, and describes how to use the Custom Extensions API.
- Programming Security for Java Applications—This document describes how to implement security in Java applications. It includes descriptions of the security service Application Programming Interfaces and programming instructions.
- Developing Security Providers—This document provides security vendors and security and application developers with the information needed to develop custom security providers.

- API Documentation—API documentation can be accessed under **API Reference** on the documentation’s main page.

CHAPTER 2

Securing Web Servers

This section provides information about using the Web Server SSM to secure Microsoft IIS and Apache Web Server.

Overview

A Web Server SSM is used to secure web server resources hosted on IIS and Apache Web Server.

Deploying the Web Server SSM also requires deployment of Web Services SSM on the same machine. The Web Server SSM communicates with the OES security framework through the Web Services SSM.

The IIS and Apache SSMs are bound to the web server as follows:

- **Internet Information Server (IIS)**

  The IIS SSM is installed as an ISAPI filter (wles_isapi.dll). For instructions, see the *SSM Installation and Configuration Guide*.

- **Apache Web Server**

  The Apache SSM is loaded as a module (`mod_wles.so` on UNIX or `mod_wles.dll` on Windows) using the **LoadModule** and **WLESConfigDir** directives. For instructions, see the *SSM Installation and Configuration Guide*.

Single Sign-On

Web single sign-on enables users to log on to one web server and gain access to other web servers in the same domain without supplying login credentials again, even if the other web servers have different authentication schemes or requirements. For information and instructions, see "Implementing Single Sign-On with ALES Identity Asserter" on page 2-9.

Constraints and Limitations

The Web Server SSM has the following constraints and limitations:

- Does not support cookie-based cross domain single sign-on except through SAML Browser/POST profile.
- Does not support cookie-based cross domain forced logoff.
- To support web farms, local configurations on each web server machine must be manually synchronized.
- Requires use of transient, non-persistent cookies to maintain session state.
- Does not support SAML Browser/Artifact profile.
- Does not preserve the original POST data during redirection to an authentication form.
- Does not save and transfer credentials between machines when more than one machine attempts to authenticate a user. Within a web farm, it is possible for a series of questions (on a form) to start on one machine and be transparently redirected to another machine within that web farm. The SSM does not save credentials that have already been entered in

the same authentication attempt. Therefore, users are forced to re-enter credential information when more than one machine is involved.

Prerequisites

The instructions provided in this chapter assume the following:

- Access to the Administration Console on the Administration Server.
- Installation of IIS or Apache web server.
- Installation and configuration of the IIS SSM or Apache SSM and creation of the SSM instance on the web server machine.
- Installation of the Web Services SSM on the web server machine.

Integration Tasks

The integration tasks described in this chapter are based on a scenario where the web server is secured by policies that determine who is able to log on. A template logon form is provided when the SSM instance is created. By following the provided instructions, you can deploy this form to the web server and test the policies.

1. Define the security providers as described in "Define the Security Providers" on page 2-3.
2. Define the web server resources in OES as described in "Define Web Server Resources" on page 2-5.
3. Define the policies that secure web server resources as described in "Define Policies" on page 2-7.
4. Distribute the policies to the SSM as described in "Distribute the Policies" on page 2-8.
5. For testing purposes, set up the web server as described in "Set Up and Test the Sample Application" on page 2-8.
6. (Optional) Set up SSO as described in "Implementing Single Sign-On with ALES Identity Asserter" on page 2-9.

Define the Security Providers

The required security providers for securing web servers are:

— Authentication
— ASI Authorization
— ASI Role Mapping
— ALES Identity Assertion
— ALES Credential Mapping

The following table provides information about each provider. For step-by-step instructions, see *Configuring Security Providers* in the Administration Console help system.

Table 2-1 Security Providers Used with Web Server SSMs

Provider	Setting
Authentication	See *Authentication Providers* in the console's help system for information on configuring this provider. For using the ALES Identity Assertion Provider, see description below.
ASI Authorization	(Required) See *Configuring an ASI Authorization Provider* in the console's help system for information on configuring this provider. Any field that this provider and the ASI Role Mapping provider have in common must be set to the same value.
ASI Role Mapping	(Required) See *Configuring an ASI Role Mapping Provider* in the console's help system for information on configuring this provider. Any field that this provider and the ASI Role Mapping provider have in common must be set to the same value.
Log4j Auditor (Optional)	See *Configuring a Log4j Audit Channel Provider* in the console's help system for information on configuring this provider.
ASI Adjudicator	See *Configuring an ASI Adjudication Provider* in the console's help system for information on configuring this provider.

Table 2-1 Security Providers Used with Web Server SSMs

Provider	Setting
ALES Identity Assertion	See *Configuring an ALES Identity Assertion Provider* in the console's help system for information on configuring this provider. **Important:** This provider and the ALES Credential Mapping provider (see below) must have the same values in the following fields: Trusted Keystore Trusted Keystore Type Trusted Cert Alias Trusted Cert Alias Confirm Trusted Cert Alias
ALES Credential Mapping	See *Configuring a ALES Credential Mapping Provider* in the console's help system for information on configuring this provider. **Important:** Some fields must have the same values set on the ALES Identity Assertion provider. See description above.

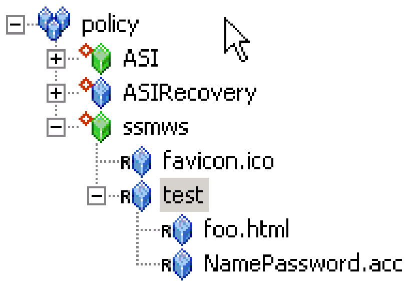
Define Web Server Resources

To secure web server resources, those resources must be defined in OES. When defining web server resources in OES, consider the following:

- Resources are identified by canonical name. For example, a web server with the names www.bea.com, www.beasys.com, www.web.internal.bea.com, and 204.236.43.12 may have a canonical name of www.bea.com.
- The Web Server SSM provides HTTP headers, cookies, query arguments, and form values to the OES security subsystem. It also decodes all URL-encoded context elements before presenting them to the security subsystem.
- A resource is presented as the path element of a URL and the file or application name. For example, `http://www.example.com/framework.jsp?FP=/products/aqualogic/` is presented as `/framework.jsp`. Query arguments/values (for example, `FP=/products/aqualogic`) are made available in the application context.

One way to secure a web site is to grant access based on a successful login to the site. This section describes how you would define the resources for securing a web server using the sample logon form provided when the IIS or Apache SSM is installed. These resources can be defined in OES as shown in Figure 2-1.

Figure 2-1 Resources Tree



To create these resources, perform the following steps:

1. In the Administration Console's left pane, select the **Resources** node.
2. In the right pane, right-click `policy` at the top of the tree and select **Add Resource**.
3. On the **Create Resource** dialog, enter `ssmws` in the **Name** field and select **binding** from the **Type** dropdown field. Then click **Ok**.

   The `ssmws` resource appears in the tree.
4. Right-click the `ssmws` resource and select **Configure Resource**. Then select the **Distribution Point** checkbox and click **Ok**.
5. Right-click the `ssmws` resource again and select **Add Resource**. Then enter `favicon.ico` in the **Name** field and click **Ok**.

   Note: The `favicon.ico` file is an icon requested by the Internet Explorer and Mozilla browsers for bookmarking URLs.
6. Right-click the `ssmws` resource and select **Add Resource**. Then enter `test` in the **Name** field and click **Ok**.
7. Right-click the `test` resource and select **Add Resource**. Then enter `foo.html` in the **Name** field and click **Ok**.
8. Right-click the `foo.html` resource and select **Add Resource**. Then enter `NamePasswordForm.acc` (for IIS) or `NamePasswordForm.html` (for Apache) in the **Name** field and click **Ok**.

Define Policies

When defining policies to secure a web server consider the following:

- Since the SSM presents the full URL (including the protocol, server name, port, full path, and query string) to the OES security framework, this information can be used in policy definitions.
- Authorization policies must contain the privilege specifying the HTTP method being allowed. Default privileges provided by OES include standard HTTP methods (GET, POST, PUT, etc.) Additional HTTP methods must be defined as Privileges before they can be used in policies.
- Policies can return response attributes to the SSM for use by some logic on the web server.

The previous section described how to create resources for the sample logon form provided when the IIS or Apache SSM is installed. This section describes the Authorization and Role Mapping policies needed to secure these resources.

Authorization Policies

Table 2-2 lists Authorization policies to secure the web server using the `NamePasswordForm` form.

Table 2-2 Sample Authorization Policies

Policy	Description
`grant(GET, //app/policy/ssmws/favicon.ico, //role/Everyone) if true;`	Allows unauthenticated users to access images used on the application login page.
`grant(GET, POST, //app/policy/ssmws/test/NamePasswordForm.acc, //role/Everyone) if true;`	Grants `GET` and `POST` privileges on the logon form for those in the `Everyone` role. **Note:** Use `NamePasswordForm.html` for Apache.
`grant(GET, //app/policy/ssmws/test/foo.html, //role/Admin) if true;`	Grants `GET` privileges for those in the `Admin` role to access the `foo.html` page.

To create the authorization polices listed in Table 2-2, perform the following steps:

1. Expand the **Policy** node in the left pane and select **Authorization Policies**. Then click **New** at the bottom of the right pane.
2. On the **Create Authorization Policy** dialog, select the **Grant** radio button and do the following:
   – On the **Privileges** tab, select `GET` in the **Select Privileges** list and click **<<Add**.
   – On the **Resources** tab, expand `ssmws`. Then select `favicon.ico` and click **<<Add**.
   – On the **Policy Subjects** tab, select `Everyone` and click **<<Add**.
3. Repeat these steps to define the remaining two authorization policies. Notice that the `Admin` role is assigned to the `foo.html` resource.

Role Mapping Policies

This section describes how to modify two existing Role Mapping policies so that they apply to the web server resources created above.

1. Expand the **Policy** node in the left pane and select **Role Mapping Policies**. Then select the **Admin** role in the right pane and click **Edit.**
2. On the **Role Mapping Policies** page, select the `Admin` role for ASI and click `Edit`. This opens the **Edit Role Mapping Policy** dialog.
3. On the **Resources** tab, select `ssmws` in the **Child Resources** list and click **<<Add**.
4. Repeat these steps for the `Everyone` role.

Distribute the Policies

To distribute information to the SSM:

1. To make sure the providers are bound to the Web Server resources, expand the SSM configuration in the left pane and select the `ASIAuthorizationProvider`. Then open the **Bindings** tab, select `//app/policy/ssmws` from the dropdown field and click **Bind**.
2. Select **Deployment** in the left pane. Then use the **Policy** and **Configuration** tabs to distribute the policy and configuration information to the SSM.

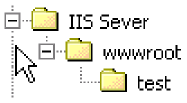
Set Up and Test the Sample Application

1. Start the Web Services SSM.

When the Web Server SSM starts, it attempts to connect to the Web Services SSM.

2. Set up the `../wwwroot/test` directory as shown in Figure 2-2 and copy the following files to it:
   - NamePasswordForm.acc—(IIS only) located in `BEA_HOME\ales32-ssm\iis-ssm\instance\<instance>\templates`
   - NamePasswordForm.html—(Apache only) located in `BEA_HOME\ales32-ssm\apache-ssm\instance\<instance>\wret\templates`
   - NamePasswordForm.html—not provided, use your version of this file.
   - atnfailure.html—not provided, use your version of this file.
   - atzfailure.html—not provided, use your version of this file.

Figure 2-2 Deploying the Sample Application



3. Modify `foo.html` to redirect to the logon form.
   - For IIS, use: `<FORM METHOD=POST ACTION="test/NamePasswordForm.acc">`
   - For Apache, use: `FORM METHOD=POST ACTION="/test/NamePasswordForm.html">`
4. Start the web server, open a browser and go to `foo.html`.
   - For IIS: `http://<machine_name_with_DNS_suffix>:80/test/foo.html`.
   - For Apache: `http://<hostmachine.cookiedomain>:8088/test/foo.html`.
5. When the browser is redirected to the logon form, enter the administrator username/password (the defaults are *admin* and *password*) and click **OK**. You are granted access to `foo.html`.

Implementing Single Sign-On with ALES Identity Asserter

You can implement web single sign-on (SSO) for the following use cases:

- Bi-directional SSO between Web Server SSMs

  Any user that authenticates to one Web Server SSM can access any other Web Server SSM in the cookie domain without having to re-authenticate.

- Uni-directional SSO between Web Server SSMs and WebLogic Server SSMs

  Any user that authenticates to one Web Server SSM can access any other WebLogic Server SSM in the cookie domain without having to re-authenticate. However, a user that authenticates to a WebLogic Server SSM cannot access another WebLogic Server SSM or another Web Server SSM without re-authenticating.

SSO Between Web Servers

To implement SSO between Web Server SSMs:

Note: For step-by-step instructions, see the Administration Console help file.

1. Using the Administration Console, configure the ALES Identity Assertion and ALES Credential Mapping providers for each participating Web Server SSM.
2. Configure the ALES Identity Assertion provider and the ALES Credential Mapping provider in each of the Web Server SSMs to use the same Trusted Cert Alias, Trusted Keystore, and Trusted Keystore Type.
3. Deploy these changes to the SSMs.

SSO Between Web Server SSM and WLS SSM

To implementing SSO between the Web Server and WLS SSMs:

Note: For step-by-step instructions, see the Administration Console help file.

1. Using the Administration Console, configure the ALES Identity Assertion and ALES Credential Mapping providers for each participating Web Server SSM and WebLogic Server SSM.

   Note: For each WLS SSM, make sure the ALES Identity Assertion provider does not use Base64 Decoding. This checkbox is located on the provider's **Details** tab.
2. Configure all providers to use the same Trusted Cert Alias, Trusted Keystore, and Trusted Keystore Type.
3. Deploy these changes to the SSMs.

Securing WebLogic Servers

In addition to providing links to other documents containing information about securing applications on WebLogic Servers, this chapter describes how to secure administrative access to WebLogic servers, how to run WebLogic server as a service, how to set up the WLS SSM to secure a WebLogic Server cluster.

- "Securing WebLogic Server Applications" on page 3-1
- "Securing Administrative Access to WebLogic Server" on page 3-2
- "Running WebLogic Server as a Service" on page 3-10
- "Setting Up WLS SSM on a WebLogic Cluster" on page 3-11

Securing WebLogic Server Applications

General instructions for securing applications hosted on WebLogic servers can be found in the following documents:

- For SSM installation instructions, see Installing SSMs in the *SSM Installation and Configuration Guide*.
- For instructions on using the ConfigTool to create the SSM instance and define an initial policy set, see Configuring SSMs Using the ConfigTool.
- All SSMs can be configured by manually defining the SSM's configuration and the policies to enforce when securing an application. Detailed instructions are provided in a number of documents, particularly the Policy Manager's Guide, Getting Started Tutorials,

and the help systems for the Administration Console and Entitlements Administration Application.

Securing Administrative Access to WebLogic Server

This chapter describes how to integrate OES with WebLogic Server and define a policy to secure administrative access to the server and the WebLogic console.

Prerequisites

This chapter assumes the following:

- Installation of WebLogic Server and creation of a WebLogic domain.
- Installation and configuration of the WLS SSM or WLS 8.1 SSM and creation of the SSM instance on the WebLogic machine.
- Access to the Administration Console on the Administration Server securing the WebLogic server.

Integration Tasks

The major tasks to perform are:

1. Define the security providers.
   - Security providers for WebLogic 8.1, are described in "WebLogic 8.1 Security Providers" on page 3-3.
   - Security providers for WebLogic 9.x/10.x, are described in "WebLogic 9.x/10.0 Security Providers" on page 3-4.

2. Define the WebLogic administrative user in OES as described in "WebLogic Administrative User" on page 3-7.
3. Define the WebLogic Server resources as described in "WebLogic Server Resources" on page 3-7.
4. Define the administrative policy as described in "Policies" on page 3-8.
5. Distribute the configuration and policy to the SSM.

WebLogic 8.1 Security Providers

This section provides information about the recommended security providers for securing administrative access to WebLogic 8.1. For step-by-step instructions using the Administration Console administration console, see the console's help system.

Table 3-1 Portal Security Configuration

Security Provider	Configuration Settings
ASI Adjudication Provider	Clear the **Require Unanimous Permit** checkbox.
Log4j Auditor	Use the default settings
Database Authentication	Set the **Control Flag** to `SUFFICIENT`. On the **Details** tab, set **Identity scope** to `myusers`. For other settings, use the defaults.
WebLogic Authentication	Define this provider only after defining the Database Authenticator. Set the Control Flag to SUFFICIENT. **Note:** The WebLogic Authentication provider can be replaced with another authentication provider that supports write access to users and groups.
ASI Authorization	On the **General** tab, accept the default settings. On the **Details** tab, set the **Identity Scope** to `myusers` and the **Application Deployment Parent** to `//app/policy/myrealm`. On the **Bindings** tab, bind to `//app/policy/myrealm`.
WebLogic Authorization Provider	Clear the **Policy Deployment Enabled** checkbox.
WebLogic Credential Mapper	Clear the **Credential Mapping Deployment Enabled** checkbox.

Table 3-1 Portal Security Configuration (Continued)

Security Provider	Configuration Settings
ASI Role Mapping Provider	On the **General** tab, accept the default settings. On the **Details** tab, set the **Identity Scope** to `myusers`.
WebLogic Role Mapper Provider	Clear the **Role Deployment Enabled** checkbox.

WebLogic 9.x/10.0 Security Providers

Defining the security providers for securing administrative access to WebLogic Server 9.2/10.0 involves tasks in both the WebLogic and the OES consoles.

The security providers plugin is required to manage OES security providers from within the WebLogic administration console. For instructions, see the next section.

Security Providers Extension

To install the plugin:

1. Make a copy of `ales_security_provider_ext.jar` located in the following directory:

   `BEA_HOME/ales32-ssm/wls-ssm/lib`

2. Move the file to `BEA_HOME/WLS_HOME/domains/<domain_name>/console-ext`, where `<domain_name>` is the domain name.

Using the WebLogic Console

This section describes how to define the security providers for using the WebLogic console. At a minimum an ASI Authorizer, ASI Role Mapper, and Log4J Auditor provider is needed.

Notes:

– When using multiple ASI Authorizers, also define an ASI Adjudicator.

– For WebLogic Portal, the security realm must also include XACML Authorizer and XACML Role Mapper providers.

To define OES security providers using the WebLogic Server 9.x/10.x administration console:

1. Make a backup copy of the `config.xml` file in the domain directory.

2. Start the WebLogic Server instance and log into the administration console.

The default URL for the console is `http://localhost:7001/console`.

3. In the Change Center, click **Lock & Edit** in the upper left part of the page.
4. In the left pane under **Domain Structure**, select **Security Realms**.
5. On the **Summary of Security Realms** page, click **New** and create a security realm using the same name as the configuration ID used by the WLS SSM instance. For the purposes of this procedure, the security realm name is `mywls9ssm`.
6. On the **Summary of Security Realms** page, select the `mywls9ssm` security realm.
7. On the **Configuration: General** page, set **Security Model Default** to **Advanced** and clear the **Combined Role Mapping Enabled** checkbox. Then click **Save**.
8. If **Check Role and Policies** is not visible, click **Advanced** and set **Set Check Role and Policies** to **All Web applications and EJBs**. Then click **Save**.
9. Select the **Providers** tab and define the following providers:

Provider Type	**Settings**
ASI Database Authenticator	Provide a name and set the type as `Database Authenticator`. On the **Configuration: Common** page, set **Control Flag** to `REQUIRED.` On the **Configuration: Provider Specific** page, set the database login, password, JDBC driver class name and JDBC Connection URL.
ASI Authorization	Provide a name and set the type as `ASIAuthorizationProvider`. On the **Configuration: Provider Specific** page, set Identity Directory and Application Deployment Parent.
ASI Role Mapper	Provide a name and set the type as `ASIRoleMapperProvider`. On the **Configuration: Provider Specific** page, set the Identity Directory and Application Deployment Parent.
Log4j Auditing\	Provide a name and set the type as `Log4jAuditor`. This provider is required in order to support logging for OES providers.

ASI Adjudicator (If using multiple ASI Authorizers)	Provide a name and set the type as `ASIAdjudicator`. On the **Configuration: Provider Specific** page, clear **Require Unanimous Permit**. **Note:** Because WLS and ASI adjudicators may return different results, the ASI Adjudicator is recommended in order to obtain appropriate adjudication results. For example, if unanimous permit is *false* and multiple authorization providers return *abstain*, the ASI Adjudicator returns *false* (denying access), while the WLS Adjudicator returns *true* (allowing access).
Credential Mapping	Provide a name and set the type as `DefaultCredentialMapper`.
Certification Path	Set the type as `WebLogicCertPathProvider` and use it to replace the existing builder.
XACML Authorizer	When securing WebLogic Portal, define a XACML Authorizer and make sure it is the first authorization provider in the list.
XACML Role Mapper (For WebLogic Portal)	When securing WebLogic Portal, define a XACML Role Mapper and make sure it is the first role mapping provider in the list.

10. Return to the console's left pane and select the domain.
11. On the **Settings** page, expand **Security > General** and select `mywls9ssm` as the default security realm and click **Save**.
12. Click **Activate Changes**.

Using the OES Administration Console

After defining the providers in the WebLogic console, perform the following steps in the OES console:

1. Log into the Administration Console. The default URL for the console is https://<*host_name*>:7010/asi.
2. Create an Identity directory using the same name specified in the WebLogic console.
3. Create an SSM configuration using the same name as the WebLogic Server security realm and define the following providers in this configuration.

Provider Type	Settings
ASI Authorizer	Set the **Identity Directory** to the directory created in step 1. Set the **Application Deployment Parent** to //app/policy/wlsserver
ASI Role Mapper	Set the **Identity Directory** to the directory created in step 1. Set the **Application Deployment Parent** to //app/policy/wlsserver

WebLogic Administrative User

The WebLogic administrative user must be defined in OES in order to start the WebLogic Server instance. To create this user:

1. Launch the Administration Console.
2. In the left pane, select the Identity node and click **New** at the bottom of the right pane.
3. On the Create Directory dialog, enter **alesusers** as the name and click OK.
4. Under this directory, create a user with the same name and password as the WebLogic administrative user. For example, if you are using the WebLogic defaults, you would use weblogic for both the username and password.

   Note: The same username and password must be specified in the WebLogic domain's `boot.properties` file.

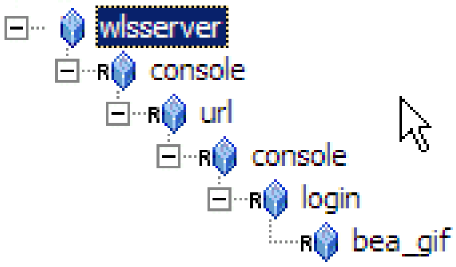
WebLogic Server Resources

WebLogic Server components must be defined in OES as resources. To create these resources using the Administration Console:

1. In the left pane, select the **Resources** node and click **New** at the bottom of the right pane.
2. In the **Name** box, type `wlsserver`, select `binding` from the **Type** dropdown list and click **OK**.

   Note: This resource will serve as the parent resource for WebLogic Server components.
3. Select `wlsserver` and click **Configure**. Then select the **Distribution Point** checkbox and click **OK**.

4. Select `wlsserver` and click **New**. Then enter `shared` in the **Name** box and then click **OK**.
5. Select `shared` and click **Configure**. Then select the **Allow Virtual Resources** checkbox and click **OK**.
6. Select `shared`, and click **New**. Then enter `svr` in the **Name** box and click **OK**.
7. Select wlsserver and create the following resource tree under it. These resources are necessary for logging into the WebLogic console.



8. Return to the left pane, expand the SSM configuration containing the defined security providers and select the ASIAuthorizer. Then open the Bindings tab in the right pane.
9. Select `//app/policy/wlsserver` from the dropdown list and click **Bind**.

Policies

A number of Authorization and Role Mapping policies must be defined to give the administrative user the necessary rights to start and manage the WebLogic Server instance. After defining these policies, distribute them to the WLS 8.1 SSM.

Authorization Policies

This policy grants the `Admin` role access to the `svr` resource:

```
grant(any, //app/policy/wlsserver/shared/svr, //role/Admin) if true;
```

To create this policy:

1. Expand the **Policy** node in the left pane and click **Authorization Policies**.
2. On the **Authorization Policies** page, click **New**.

3. On the **Create Authorization Policy** dialog, select the **Privileges** tab. Then select the `any` privilege and click **Add**.
4. On the **Resources** tab, expand the `wlsserver` and `shared` nodes in the **Child Resources** list box, select `svr`, and then click **Add**.
5. On the **Policy Subjects** tab, select `Admin` from the **Roles List** list box and click **Add**.
6. To define access to the WebLogic console:, repeat these steps to create the following policies:

```
grant(any, //app/policy/wlsserver/console, //role/Admin) if true;
grant( //priv/GET,
//app/policy/wlsserver/console/url/console/login/bea_logo.gif,
//sgrp/alesusers/allusers/) if true;
```

Role Mapping Policies

This policy assigns the `weblogic` user to the Admin role.

```
grant(//role/Admin, //app/policy/wlsserver, //user/alesusers/weblogic/)
   if true;
```

Note: When creating this policy, replace `weblogic` with the actual user name.

To create this policy:

1. Expand the **Policy** node in the left pane and click **Role Mapping Policies**.
2. On the **Role Mapping Policies** page, click **New**.
3. On the **Create Role Mapping Policy** dialog, select the **Roles** tab. Then select `Admin` from the **Available Roles** list and click **Add**.
4. On the **Resources** tab, select `wlsserver` in the **Child Resources** list and click **Add**.
5. On the **Policy Subjects** tab, select `Users` from the **Select Policy Subjects From** dropdown field and change the directory to `alesusers`. Then select `weblogic` from the list and click **Add**.

Distribute the Policies

To distribute information to the SSM:

1. To make sure the providers are bound to the Web Server resources, expand the SSM configuration in the left pane and select the `ASIAuthorizationProvider`. Then open the **Bindings** tab, select `//app/policy/wlsserver` from the dropdown field and click **Bind**.

2. Select **Deployment** in the left pane. Then use the **Policy** and **Configuration** tabs to distribute the policy and configuration information to the SSM.

Running WebLogic Server as a Service

The domain directory for a WebLogic Server normally contains a boot `startWebLogic.cmd` file that runs as part of a command or shell prompt. In order to make the server run as a service or daemon process, perform the following steps:

1. Using an editor, modify `<WLS_SSM_INSTANCE_HOME>/config/WLESWebLogic.conf` as described in Table 3-2.

   The parameters to modify have comments that start and end with ***.

Table 3-2 Updates to WLESWebLogic.conf

Parameter	Description
wrapper.working.dir	Specify the WebLogic working directory, for example `BEA_HOME/weblogic92`.
weblogic.RootDirectory	Specify the WebLogic working directory, for example `BEA_HOME/weblogic92`.
wles.user.alias	The boot user specified when the domain was created. **Note:** Make sure to use the `asipassword.bat|sh` tool to add the password for this alias to the `password.xml` file.
weblogic.Name	The WebLogic server that is part of the domain.

2. Depending on the operating system, edit `<WLS_SSM_INSTANCE_HOME>/bin/WLESWebLogic.bat|sh` as described in Table 3-3.

Table 3-3 Modifying WLESWebLogic.bat | sh

Operating System	
Windows	In `WLESWebLogic.bat`, uncomment the *@rem goto beenedited* line, for example: `goto beenedited`
UNIX	In `WLESWebLogic.sh`, comment out the *exit 1* line, for example: `# exit 1`

3. Open a command prompt or unix shell and go to the `<WLS_SSM_INSTANCE_HOME>/bin` directory.
4. Run `WLESWebLogic.bat|sh register`.

After this, the server can be started or stopped by executing `WLESWebLogic.bat|sh start` or `WLESWebLogic.bat|sh stop`.

Setting Up WLS SSM on a WebLogic Cluster

This document provides the high-level steps for setting up OES to protect a cluster of WebLogic Server 9.x/10.x domain instances.

- "Topology" on page 3-11
- "Steps" on page 3-12

Topology

This document assumes the following deployment of OES components:

- The Administration Server is running on a separate machine.
- Two WLS SSM instances running on WebLogic cluster server 1. One WLS SSM instance is used to secure the cluster's administrative server; the other to secure application resources on managed server 1.
- One WLS SSM instance running on WebLogic cluster server 2 to secure application resources on managed server 2.

Steps

1. Install Administration Server and verify the installation by logging in to the Administration Console.
2. Create a domain called **cluster_admin** on cluster server 1. Verify correct setup by starting the server and logging in to the WebLogic administration console. Then stop the server.
3. Install the WLS SSM on both cluster server 1 and cluster server 2 in the same BEA_HOME as the WebLogic server. When prompted for the SCM instance, use the same name on both machines (for example, `cluster_scm`).
4. If necessary, perform the enrollment process and run the `asipassword` utility as described in chapter 3 of the SSM Installation and Configuration Guide.
5. Make a copy of `BEA_HOME/ales32-ssm/wls-ssm/adm\myssm_config.properties` and name it something like `cluster_admin_config.properties`.
6. Edit `cluster_admin_config.properties` so that it points to the WebLogic domain directory on cluster server 1. Then edit other properties as needed.

   Note: For the **Config ID**, use a name similar to `cluster_ssm`.
7. Run `ConfigTool -check cluster_admin_config.properties` to verify that all settings are correct. When there are no errors, run `ConfigTool -process cluster_admin_config.properties.`

   This enables OES on the WebLogic domain.
8. Start the WebLogic server using `startWebLogic.cmd` located in the domain's `/bin` directory.
9. Log in to the WebLogic console and create the cluster (`ales_cluster`) and managed servers (managed server 1 is also located on cluster server 1; managed server 2 is located on cluster server 2).

   Instructions for WebLogic 9.x can be found at http://e-docs.bea.com/wls/docs92/ConsoleHelp/taskhelp/clusters/ClusterRoadmap.html. For WebLogic 10.0, see http://e-docs.bea.com/wls/docs100/ConsoleHelp/taskhelp/clusters/ClusterRoadmap.html
10. On cluster server 1:

    a. Run the WLS SSM instance wizard and create the SSM instance with the same Config ID used in step 6.

   b. Make a copy of managed server 1’s `startWebLogic.cmd` named `startWeblogicM1.cmd`. Update the file so that the file references `set-wls-env.cmd` in the WLS SSM instance’s `/bin` directory.

   c. Make a copy of `startManagedServer.cmd` named `startM1Server.cmd` so that it references `startWebLogicM1.cmd` and its arguments when calling that file are `M1 http://<cluster_server1_IP>:<port>`.

      This allows you to start the managed server by running `startM1Server.cmd`.

11. On cluster server 2:

    a. Run the WLS SSM instance wizard and create the WLS SSM instance using the same **Config ID** used in step 6 above (for example, `cluster_ssm`).

    b. Make a copy of `startWebLogic.cmd` named `startWeblogicM2.cmd`. Update the file so that it references `set-wls-env.cmd` in the SSM instance’s `/bin` directory.

       Note: Also update the CLASSPATH and JAVA_OPTIONS to be similar to those in `startWebLogicM1.cmd`.

    c. Make a copy of `startManagedServer.cmd` named `startM2Server.cmd` so that it references `startWebLogicM2.cmd` and its arguments when calling that file are `M1 http://<cluster_server1_IP>:<port>`.

       This allows you to start the managed server by running `startM1Server.cmd`.

12. Start the managed servers on both servers. They should be able to locate the cluster_admin instance, obtain the realm information, and boot up correctly.

    Note: The realm name will be the same Config ID name used in step 6 above (for example, `cluster_ssm`).

CHAPTER 4

Securing WebLogic Workshop Applications

This section describes how to secure applications developed using Workshop for WebLogic and discusses how to use the Annotations Plugin and the Tag Library.

- “Overview” on page 4-1
- “OES Tag Library for Workshop” on page 4-8
- “Tag Library Reference” on page 4-14

Overview

The Annotations Plugin can be used to annotate EJB objects in Workshop with security related metadata. The metadata can then help you to:

- Generate policy files, with lists of resources and their attributes, that can be imported into OES.
- Facilitate creation of policy rules by providing a higher level of indirection, achieved by writing the policies against the predefined metadata instead of against EJB class and method names.
- Divide responsibilities of an EJB developer and a security specialist.
- Improve maintainability of the security model. For example, since security policies are written against the metadata, you don’t need to modify them if an EJB class or EJB method name has been changed or added until the proper metadata is attached.

Integration Tasks

This section provides instructions for creating an EJB with OES annotations and then securing it with policy. The integration tasks provided here are based on those instructions.

1. Set up the plugin in Workshop as described in "Set Up the Annotations Plug-in" on page 4-2
2. Annotate the EJB and export the policy file to OES. For an example, see "Using OES Annotations in a WebLogic Bean Class" on page 4-3.
3. Define OES policies for securing the EJB as described in "Define Policies for the Imported Policy File" on page 4-8.

Set Up the Annotations Plug-in

The Annotations plugin is provided in two JAR files for use with Workshop 9.2 or 10.0:

```
com.bea.wlw.ales.annotations_9.2.0.jar
com.bea.wlw.ales.annotations_10.0.0.jar
```

To install the plugin:

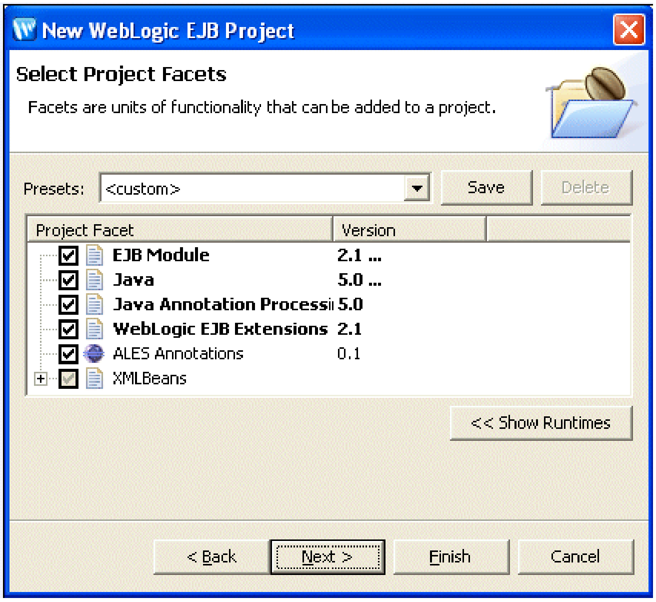
1. Copy the appropriate file from `BEA_HOME/ales32-admin/lib/eclipsePlugins` to the following directory:

   ```
   BEA_HOME/WORKSHOP_HOME/workshop4WP/eclipse/plugins
   ```

2. Restart Workshop.

   You should see the Annotations facet in a new or existing WebLogic EJB project.

Figure 4-1 Project Facets


New WebLogic EJB Project

Select Project Facets

Facets are units of functionality that can be added to a project.

Presets: <custom>

Project Facet	Version
☑ EJB Module	2.1 ...
☑ Java	5.0 ...
☑ Java Annotation Processi	5.0
☑ WebLogic EJB Extensions	2.1
☑ ALES Annotations	0.1
☑ XMLBeans	

Using OES Annotations in a WebLogic Bean Class

This section contains the following topics:

- "Create a WebLogic SessionBean" on page 4-3
- "Add OES Annotations to the WebLogic Bean Class" on page 4-4
- "Add Information to the Project" on page 4-5
- "Export the Policy File from Workshop" on page 4-6
- "Import the Policy File" on page 4-6
- "Define Policies for the Imported Policy File" on page 4-8

Create a WebLogic SessionBean

The first step in this example is to use Workshop to create a WebLogic Session Bean:

1. Create a new project named `EjbExample`.

2. Select `EjbExample` in the left panel, right click the `src` node and select **New** > **Package**. Then name the package `beans`.

3. Right click on `beans` and select **New** > **WebLogic Session Bean**. Then set the file name to `AccountService`.

Add OES Annotations to the WebLogic Bean Class

To add security annotations to the `AccountService` class:

1. In the edit window for the Weblogic Session Bean class `AccountService.java`, import the following two classes:

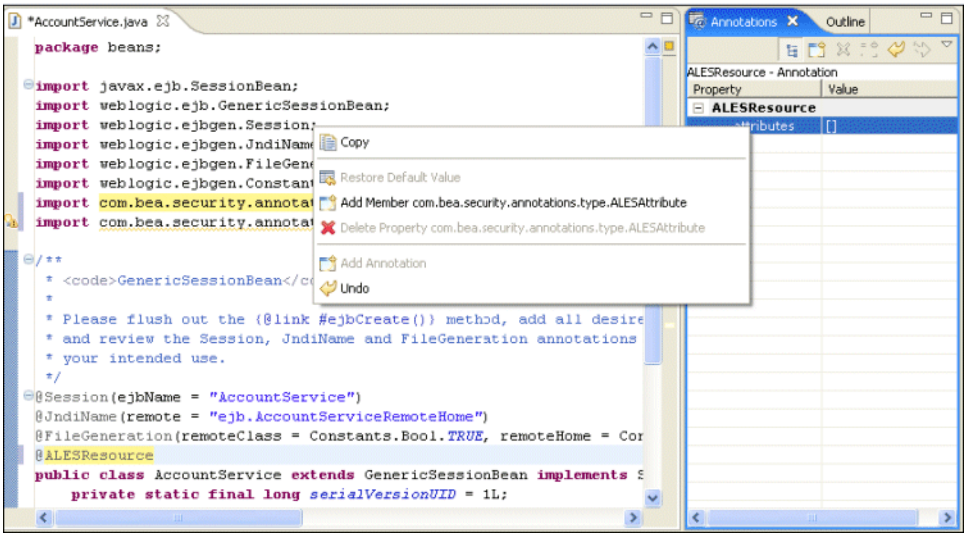
   ```
   import com.bea.security.annotations.type.ALESResource;
   import com.bea.security.annotations.type.ALESAttribute;
   ```

2. To define the Bean class `AccountService` as an OES resource:

   a. Before the class definition, add the annotation `@ALESResource`. The **ALESResource – Annotation** property appears in the Annotations viewer.

   b. Under `ALESResource`, right click **attributes** and click **Add Member com.bea.security.annotations.type.ALESAttribute**. Attributes may now be assigned to this resource.

Figure 4-2 Assigning ALESAttributes


*AccountService.java
Annotations
Outline
Copy
Restore Default Value
Add Member com.bea.security.annotations.type.ALESAttribute
Delete Property com.bea.security.annotations.type.ALESAttribute
Add Annotation
Undo

```
package beans;

import javax.ejb.SessionBean;
import weblogic.ejb.GenericSessionBean;
import weblogic.ejbgen.Session;
import weblogic.ejbgen.JndiName
import weblogic.ejbgen.FileGene
import weblogic.ejbgen.Constant
import com.bea.security.annotat
import com.bea.security.annotat

/**
 * <code>GenericSessionBean</c
 *
 * Please flush out the {@link #ejbCreate()} method, add all desire
 * and review the Session, JndiName and FileGeneration annotations
 * your intended use.
 */
@Session(ejbName = "AccountService")
@JndiName(remote = "ejb.AccountServiceRemoteHome")
@FileGeneration(remoteClass = Constants.Bool.TRUE, remoteHome = Cor
@ALESResource
public class AccountService extends GenericSessionBean implements S
    private static final long serialVersionUID = 1L;
```

ALESResource - Annotation

Property	Value
ALESResource	
attributes	[]

3. Expand the **ALESAttribute** node under **ALESResource > attributes** in the right panel and set the name and value of the ALESAttribute to `beantype = account`.
4. Similarly, we can define any methods inside this class as OES resources and assign OES attributes to them using the Annotations plug-in. For example, annotate the `ejbCreate()` method to define it as an OES resource with the ALESAttribute `operation = create`.

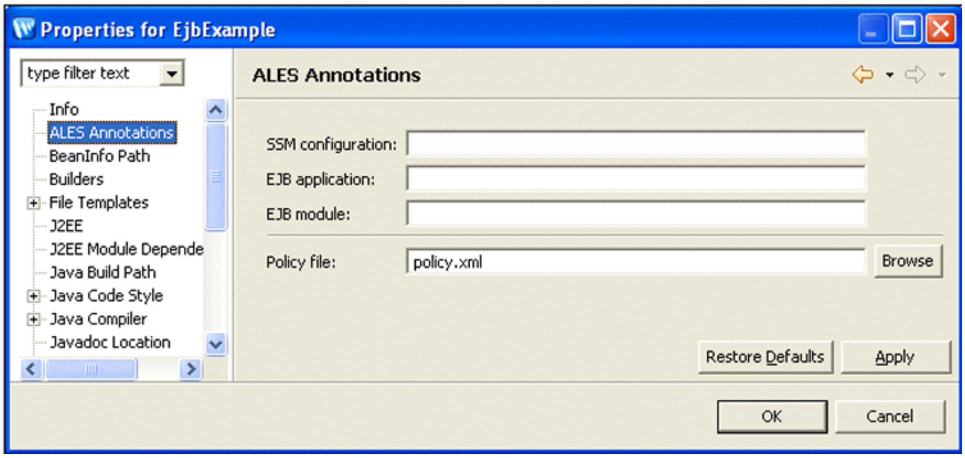
   ```
   @ALESResource(attributes={@ALESAttribute(name = "operation", value =
   "create")})
   ```

Add Information to the Project

To define OES-specific properties for the project:

1. In Workshop's left pane, right -lick the **EjbExample** project node and click **Properties**. The **Properties** window appears.

Figure 4-3 OES Annotation Project Properties



SSM configuration:
EJB application:
EJB module:
Policy file: policy.xml

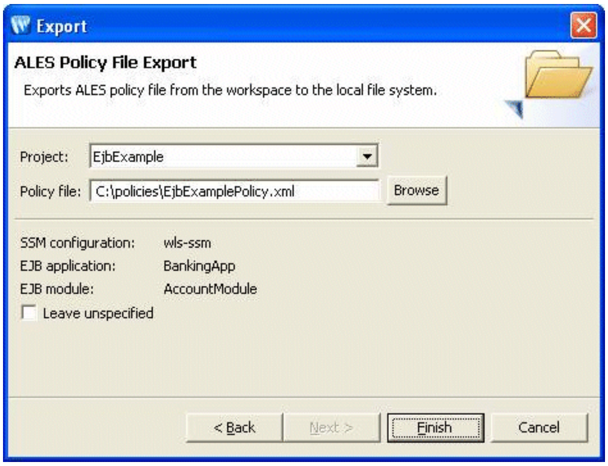
2. Select **ALES annotations** and configure the following four properties:
   – SSM configuration—SSM to bind to the EJB application with; for example,`wls-ssm`.
   – EJB application—EJB application to deploy; for example, `BankingApp`.
   – EJB module—EJB module to deploy; for example, `AccountModule`.
   – Policy file—absolute path of the policy file to export the OES annotations.

Export the Policy File from Workshop

After annotating the project in the example, the security policy file can be exported from Workshop and then imported into OES:

1. In the Package Explorer or Navigator panel of Workshop, right-click the **EjbExample** project node, select **Export** > **ALES Export Policy File,** and then click **Next**.

Figure 4-4 Policy File Export Window


Export
ALES Policy File Export
Exports ALES policy file from the workspace to the local file system.
< Back
Next >
Finish
Cancel
Browse

Project: EjbExample
Policy file: C:\policies\EjbExamplePolicy.xml
SSM configuration: wls-ssm
EJB application: BankingApp
EJB module: AccountModule
☐ Leave unspecified

2. In the **ALES Policy File Export** window, specify the project name and the pathname for the policy file (for example, `EjbExamplePolicy.xml`).

   If the **Leave unspecified** checkbox is checked, the policy file will includes tokens for SSM configuration, EJB application, and EJB module instead of the values you specified for them. Later, the EJB application deployer can replace these tokens. This functionality is useful when the developer does not know all the deployment parameters of the target machine, or the EJB application is going to be deployed on multiple machines with different configurations.

Import the Policy File

This section describes how to use the policyIX utility to load the Annotations policy file created in Export the Policy File from Workshop into OES.

1. If you have not already done so, start the Administration Server and load the admin policies running the `install_ales_schema` script.

2. If **Leave unspecified** was selected when the policy file was created, the file includes tokens instead of specific values for the SSM configuration, EJB application, and EJB module. Before importing the policy, you must replace these tokens with the actual values. In `ales32-admin/config/annotation_config.properties`, replace the tokens with the corresponding values:

   `ap.ssm.id`—SSM configuration
   `ap.ejb.app`—EJB application
   `ap.ejb.mod`—EJB module
   `exported.res.file`—pathname of the policy file

   After replacing the tokens, run `BEA_HOME/ales32-admin/bin/annotation_transform` to create the policy file using the actual values. The file will have the same name as the `exported.res.file` parameter with the extension `.import` appended.

3. Use the policyIX utility to import the policy file. For example:

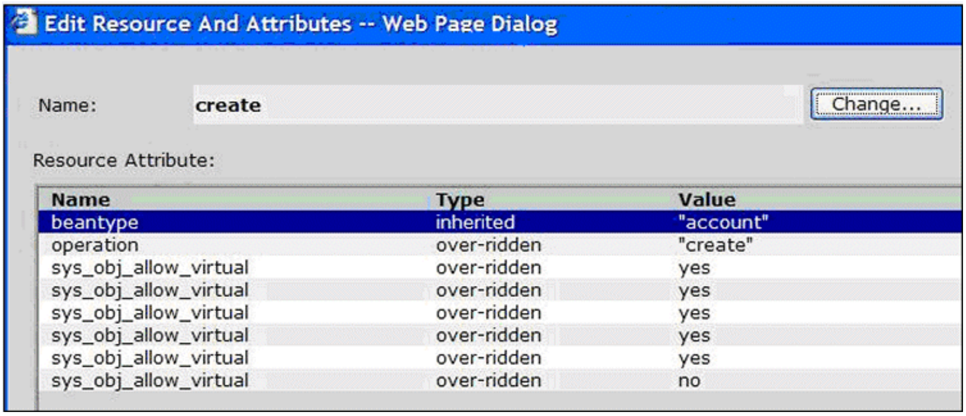
   ```
   <BEA_HOME>/ales32-admin/bin> policyIX -import config.xml
   EjbExamplePolicy.xml
   ```

   For more information, see policyIX in the *Administration Reference*.

After importing the policy, use the Administration Console to view the resources and resource attributes. Under `wls-ssm/BankingApp/ejb/AccountModule`, you can see:

- `AccountService` has the attribute `beantype="account"`
- The `AccountService.create` method inherits the attribute `beantype="account"` and has its own attribute `operation="create"`.

Figure 4-5 Resource Attributes for the create Method



Name: create

Resource Attribute:

Name	Type	Value
beantype	inherited	"account"
operation	over-ridden	"create"
sys_obj_allow_virtual	over-ridden	yes
sys_obj_allow_virtual	over-ridden	yes
sys_obj_allow_virtual	over-ridden	yes
sys_obj_allow_virtual	over-ridden	yes
sys_obj_allow_virtual	over-ridden	yes
sys_obj_allow_virtual	over-ridden	no

Define Policies for the Imported Policy File

After importing the policy file, you may create policies that define access rules for the resources defined with OES annotations. For example, the following rule grants the `execute` privilege to the `AccountManagerRole` for all the resources under `BankingApp` if `beantype=”account”`:

```
grant( //priv/execute, //app/policy/wls-ssm/BankingApp,
//role/AccountManagerRole) if beantype="account"
```

Since all methods inherit attributes of the bean class, the resource `AccountService` and all its methods satisfy the constraint, defining this rule enables the AccountManagerRole to execute all `AccountService` methods.

The following rule allows a user with the Customer role to execute any method in the banking application, but only if the method is annotated with the attribute `accessLevel="customer"`. The rule also allows the user to create an instance of the EJB by calling `create` method of the corresponding EJB's home interface:

```
grant( //priv/execute, //app/policy/wls-ssm/BankingApp, //role/Customer) if
accessLevel="customer" or operation="create"
```

OES Tag Library for Workshop

The OES Tag Library for Workshop can be used to add security functionality to Java Server Pages (JSPs).

Tag libraries provide a way to abstract functionality used by a JSP page, which allows for less-complex JSP pages. A tag library packages functions into a tag handler class. A JSP does not have to directly invoke this tag handler. Instead, you place simple tags in your JSP pages. When the container executes a JSP at runtime and comes across a tag, the tag handler is invoked and provides the desired functionality.

Prerequisites

The requirements for using the Tag Library are:

- Installation of Workshop 9.x or 10.0.
- Installation of the WLS SSM on the Workshop machine.
- Access to the Administration console and/or the Entitlements Administration Application.

Library Tags

The following tags are available in the Tag Library for Workshop. See "Tag Library Reference" on page 4-14 for additional information and attributes.

- **isAccessAllowed** – If access is allowed, display the body of the tag. If not, skip the body. It returns true or false and a variable to the body of the JSP that can be used to process responses.
- **isAccessNotAllowed** – If access is not allowed, display the body of the tag. Otherwise, skip the body. It returns true or false and a variable to the body of the JSP that can be used to process responses.
- **isAccessAllowedQueryResources** – Returns the set of granted and denied responses from the query resources functionality of isAccessAllowed. This returns a variable to the JSP that can be used to process responses.
- **getUserRoles** – Makes the set of user roles available to the application. This returns a variable to the JSP that can be used for processing.
- **isUserInRole** – Returns true if the current user has a specific role.
- **recordEvent** – Passes an audit event into OES. This is an event tag that provides input to the OES auditing system.
- **setSecurityContext** – Sets up data for the other tags. You set a value to be used as a prefix for all other resources on the page. For example: `<setSecurityContext value="/mybank/loanApplicationForm"/>`. Later on the page, when a resource is

specified for an `isAccessAllowed` call, it will be prepended with `/mybank/loanApplicationForm`.

Any attributes set within the tag are passed to every OES API call. For example if you set `foo=1` in the security context and then use the `isAccessAllowed` tag, `foo=1` would be available during policy evaluation as an application context variable.

Integration Tasks

This section provides instructions for adding OES tags to a JSP page. The integration tasks provided here are based on those instructions.

1. Add the OES Tag Library to Workshop as described in "Add the Tag Library to Workshop" on page 4-10
2. Add OES tags to the JSP pages as described in "Using OES Tags in JSP Pages" on page 4-11.
3. In OES, define the policies for securing the JSP components as described in "Define the Policies to Secure JSP Components" on page 4-12.
4. Distribute the policy data to the WLS SSM as described in "Deploy the JSP Application" on page 4-13.

Add the Tag Library to Workshop

A file named `alestags.jar` contains the tag library and supporting classes. This file is packaged in one of the following WebLogic version-specific files located in the `BEA_HOME\ales32-admin\lib\eclipsePlugins` directory:

- `com.bea.wlw.ales.tags_9.2.0.jar`
- `com.bea.wlw.ales.tags_10.0.0.jar`

Follow these steps to integrate the tag library JAR file:

1. Copy `com.bea.wlw.ales.tags_9.2.0.jar` or `com.bea.wlw.ales.tags_10.0.0.jar` from `BEA_HOME\ales32-admin\lib\eclipsePlugins` to the following directory:

   `BEA_HOME``\<workshop_version>\workshop4WP\eclipse\plugins`

   where

   `<workshop_version>`—the Workshop directory for versions 9.x or 10.0
2. Restart Workshop.
3. If creating a new project:

   a. Create and provide a name for a new dynamic web project.
   b. Then click and select the **OES Tag Support** project facet and click **Finish**.
   c. Click **Window->Show View->JSP Design Palette** to display the JSP Design Palette.

4. If adding to an existing project:
   a. Right-click on the project and select **properties**.
   b. Click the project facets properties in the left navigation bar of the popup.
   c. Click add/remove facets.
   d. Select the **OES Tag Support** project facet and click finish.
   e. Click **Window->Show View->JSP Design Palette** to display the JSP Design Palette.

At the conclusion of these steps, `alestags.jar` should be located in the `web-inf/lib` directory and the JSP Design Palette should display the OES Tags.

Using OES Tags in JSP Pages

The OES tags can wrap the JSP components and they will be rendered if allowed. There is also an `else` tag for the case where a component is denied.

Listing 4-1 shows an example of using OES tags in a JSP page.

Listing 4-1 Example JSP Page With OES Tags

```
<%@ taglib prefix="ales" uri="http://www.bea.com/ales/tags"%>
<%@ taglib  prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>IsAccessAllowedSample</title>
</head>
<ales:setSecurityContext value="/testtagapp">
<ales:attribute name="foo" value="1"/>
</ales:setSecurityContext>
```

```
<body>
<ales:isAccessAllowed resource="/isAllowed" action="view">
<ales:then>You are allowed to see the secret text</ales:then>
<ales:else>DenyReason: <c:out value='${responses["denyreason"]}'/>
</ales:else>
</ales:isAccessAllowed>
</body>
</html>
```

In Listing 4-1 the `<@taglib prefix=”ales” uri=http://www.bea.com/ales/tags%>` line signifies that all tags prefixed with `ales:` will call the OES tag library.

The `ales:isAccessAllowed` tag takes a resource and an action.

Resources on the JSP page are relative to the WLS SSM’s binding node. In addition, one of the parameters you pass in to the (optional) `setSecurityContext` tag is a resource URI. This URI is relative and pre-pended to the SSM binding node at runtime. In the example, the resource URI is `testtagapp.`

Therefore, after adding the binding node and resource URI, the fully qualified resource name at runtime is `//app/policy/<`*`binding_node`*`>/testtagapp/isAllowed`.

Define the Policies to Secure JSP Components

Use the Entitlements Administration Application to create policies for the resources on the JSP page.

Based on the example, we would create a resource `//app/policy/<binding_node>/testtagapp/isAllowed` where `<binding_node>` would be the SSM’s binding node from Step 1. We could then write policies based on that resource to determine what users are allowed to see the secret text.

The parameter passed in to the (optional) `setSecurityContext` tag is a resource URI, which is a value to be used as a prefix for all other resources on the page. For example: `<setSecurityContext value=”/mybank/loanApplicationForm”/>`.

This URI is relative and prepended with the SSM binding node at runtime. This resource URI uniquely identifies the resources.

Any attributes set within `setSecurityContext` are passed to every API call. For example, if you set `foo=1` in the security context and then use the `isAccessAllowed` tag, `foo=1` would be available to policies as an application context variable.

Use the resources and actions specified in the tags when creating the policy definitions. Be aware that the fully qualified name of the resource is:

```
//app/policy/<binding node>/<SecurityContext-Value>/resource
```

Using Policies to Return Response Attributes

There is nothing unique about policies that protect a resource referenced in a JSP file by a tag library (beyond the general requirement that the resource names and actions you specify in tags must correlate with the policy). However, by using result and response attributes such as `isAccessAllowed.resultVar`, the policy can be used to return and then test those response attributes.

As described in Using Response Attributes, response attributes are typically specified using built-in evaluation functions that report name/value pairs. There are two functions for returning attributes: `report()` and `report_as()`. These functions always return TRUE (if there are no errors), and the information is passed to the application as response attributes.

The JSP Standard Tag Library (JSTL) provides a set of core functionality common to most web applications, including generic iterators and Boolean checks. As such, the tag library does not implement its own set of iterators. The data returned by the tags to the JSP can be processed using the JSTL.

Therefore, a JSP file can contain a JSTL tag used in the following way:

```
<ales:isAccessAllowed resource="/creditScore" action="view"
resultVar="canViewCreditScore">
:
:
<c:if test="${canViewCreditScore == true}">
  Show customer credit score
</c:if>
```

Deploy the JSP Application

When the WebLogic Server container on the local system executes a JSP at runtime and encounters an OES tag, the tag handler is invoked and interacts with the WLS 9.x SSM to determine if access is allowed to the resource, get the user roles, and so forth.

You do not need to supply an authenticated subject to the OES tags for Workshop. This is because WebLogic Server determines and authenticates the subject and then makes the authenticated subject available for authorization decisions. No special action in the JSP page is required.

JSP pages contain OES tags can be tested even if the SSM is not deployed. OES tags will allow both ‘allow’ and ‘deny’ cases to display on the page and data retrieval methods return empty results.

Tag Library Reference

This section describes reference and usage information for the tag library.

isAccessAllowed

The `isAccessAllowed` tag calls the OES runtime to determine if the user is allowed to perform the requested action on the requested resource. If `true` is returned, it allows the container to continue processing the body of the tag.

For convenience, the result of calling `isAccessAllowed` can be placed within the `resultVar` for later use.

`isAccessAllowed` looks for the ARME configuration file in the system properties. If not found, all elements in the body tag are displayed.

Table 4-1 isAccessAllowed

Attribute	Return Type	Description	Required
resource	n/a	The resource used when calling isAccessAllowed	Yes
action	n/a	The action used when calling isAccessAllowed. The default action is view	No
resultVar	Boolean	The name of the scripting variable used to tell if access is allowed.	No
resultVarScope	n/a	The scope of the resultVar (page, request, session, or application). The default scope is page.	No
responseVar	Collection of Strings	The name of the variable used for returning responses from calling isAccessAllowed	No
responseVarScope	n/a	The scope of the variable containing responses from is access allowed (page, request, session, or application). The default scope is page.	No

isAccessAllowed Concepts

Listing 4-2 shows the concepts of using `isAccessAllowed` in a JSP.

Listing 4-2 Using isAccessAllowed in a JSP

```
<%-- set up the global context for this jsp page --%>
<ales:setSecurityContext value="/mybank/loanApplicationForm">
 <attribute name="loanAppId" value="<%=
request.getParameter(\"appId\")%>"/>
</ales:setSecurityContext>

<ales:isAccessAllowed resource="/creditScore" action="view"
resultVar="canViewCreditScore">
<attribute name="customerId"
value="<%=request.getParameter(\"customerId\")%>

<ales:then>
       <bankapp:displayCreditScore>
</ales:then>
<ales:else>
 <B>You are not authorized to view the customer's credit score</B>
</ales:else>
</ales:isAccessAllowed>
```

Alternatively, after the `isAccessAllowed` tag with the `resultVar canViewCreditScore` attribute, you could use a JSTL tag in the following way:

```
<c:if test="${canViewCreditScore == true}">
  Show customer credit score
</c:if>
```

isAccessNotAllowed

The `isAccessNotAllowed` tag calls the OES runtime to determine if the user is allowed to perform the requested action on the requested resource. If `true` is returned, it allows the container to continue processing the body of the tag.

For convenience, the result of calling `isAccessAllowed` can be placed within the `resultVar` for later use.

`isAccessNotAllowed` looks for the authorization and role mapping engine (ARME) configuration file in the system properties. If not found, all elements in the body tag are displayed.

Table 4-2 isAccessNotAllowed

Attribute	Return Type	Description	Required
Resource	n/a	The resource used when calling isAccessAllowed	Yes
Action	n/a	The action used when calling isAccessAllowed. The default action is view	No
resultVar	Boolean	The name of the scripting variable used to tell if access is allowed.	No
resultVarScope	n/a	The scope of the resultVar (page, request, session, or application). The default scope is page.	No
responseVar	Collection of Strings	The name of the variable used for returning responses from calling isAccessAllowed	No
responseVarScope	n/a	The scope of the variable containing responses from is access allowed (page, request, session, or application). The default scope is page.	No

isAccessNotAllowed Concepts

Listing 4-3 shows the concepts of using `isAccessNotAllowed` in a JSP.

Listing 4-3 Using isAccessNotAllowed in a JSP

```
<%-- set up the global context for this jsp page --%>
<ales:setSecurityContext value="/mybank/loanApplicationForm">
 <attribute name="loanAppId" value="<%=
request.getParameter(\"appId\")%>"/>
</ales:setSecurityContext>
```

```
<ales:isAccessNotAllowed resource=”/creditScore” action=”view”
resultVar=”canNotViewCreditScore”>
<ales:then>
<B>You are not authorized to view the customer’s credit score</B>
</ales:then>
<ales:else>
 <attribute name=”customerId” value=”<%=
request.getParameter(\”customerId\”)%>

 <bankapp:displayCreditScore>
</ales:else>
</ales:isAccessNotAllowed>
```

Alternatively, after the `isAccessNotAllowed` tag using the `resultVar` `canNotViewCreditScore` attribute, you could use a JSTL tag in the following way:

```
<c:if test=”${canNotViewCreditScore == false}”>
  Show customer credit score
</c:if>
```

isAccessAllowedQueryResources

The `isAccessAllowedQueryResources` tag calls the OES runtime using the query resource extra attribute. This tag does not have a body associated with it. Instead, it returns a set of data that can be consumed by the JSP.

The `grantedVar` attribute sets a variable containing the granted response set from the ARME. The `deniedVar` attribute sets a variable containing the denied response set from the ARME.

If the ARME configuration file is not found, this tag does not set any data for the JSP to consume.

Table 4-3 isAccessAllowedQueryResources

Attribute	Return type	Description	Required
resource	n/a	The resource used when calling isAccessAllowed	Yes
action	n/a	The action used when calling isAccessAllowed. The default action is view	No

Table 4-3 isAccessAllowedQueryResources

Attribute	Return type	Description	Required
responseVar	Collection of Strings	The name of the variable used for returning responses from calling isAccessAllowed	No
responseVarScope	n/a	The scope of the variable containing responses from is access allowed (page, request, session, or application). The default scope is page.	No
grantedVar	Collection of Strings	The set of granted responses returned from the ARME	No
grantedVarScope	n/a	The scope of the grantedVar variable (page, request, session, or application). The default scope is page.	No
deniedVar	Collection of Strings	The set of denied responses returned from the ARME	No
deniedVarScope	n/a	The scope of the deniedVar variable (page, request, session, or application). The default scope is page	No

isAccessAllowedQueryResources Concepts

Listing 4-4 shows the concepts of using `isAccessAllowedQueryResources` in a JSP.

Listing 4-4 Using isAccessAllowedQueryResources in a JSP

```
<%-- Get a list of currencies that a trader can exchange to put inside a
dropdown list --%>
<ales:isAccessAllowedQueryResources resource=”
/mybank/currencyTrader/availableCurrencies”
grantedVar=”grantedCurrencies”>
<attribute name=”currencyToTradeFrom” value=”USD”/>
</ales:isAccessAllowedQueryResources>
```

```
<%--This fake sample tag takes in a collection of strings and lists them in
a drop down--%>
<myuitag:dropdownlist values=”${%grantedCurrencies%}”/>
```

getUserRoles

The `getUserRoles` tag queries the OES system for the set of roles that a user currently has in the system for the requested action and requested resource. It will return the collection of role names as a variable defined by the `resultVar` attribute.

Table 4-4 getUser Roles

Attribute	Return Type	Description	Required
Resource	n/a	The resource used when calling getRoles	Yes
Action	n/a	The action used when calling getRoles. The default action is view	No
resultVar	Collection of Strings	The name of the variable to set that contains the list of user’s roles	Yes
Scope		The scope of the variable containing responses from is access allowed (page, request, session, or application). The default scope is page.	No

getUserRoles Concepts

Listing 4-5 shows the concepts of using `getUserRoles` in a JSP.

Listing 4-5 Using getUserRoles in a JSP

```
<%-- Get the list of roles the user has for a particular resource --%>
<ales:getUserRoles resource=”/mybank/loanApprovalForm”
resultVar=”userRoles”>
 <attribute name=”customerId” value=”${currentCustomerId}”/>
</ales:getUserRoles>
<%--iterate over each role and print it out--%>
```

```
<c:forEach var=”userRole” items=”${userRoles}”>
<c:out value=”${userRole}”/>
</c:forEach>
```

isUserInRole

The `isUserInRole` tag is a conditional and cooperative tag. If the user is in the role specified, the body of the tag will be processed. If the user has the role passed into the tag “Attribute” on page 4-22, the body of the tag will be processed. In addition. the `resultVar` can be used for later processing.

Table 4-5 isUserInRole

Attribute	Return Type	Description	Required
Role	n/a	The name of the role to check against the user	Yes
Resource	n/a	The name of the resource to check the user’s roles against. The default value will be the current JSP page	No
Action	n/a	The action against the resource to check the user’s role against. The default value will be view	No
resultVar	Boolean	A variable to hold the result from isUserInRole for use later	No

isUserInRole Concepts

Listing 4-6 shows the concepts of using `getUserRoles` in a JSP.

Listing 4-6 Using isUserInRole in a JSP

```
<%-- Check if the user is in the appropriate role before showing the buttons
on the page --%>
<isUserInRole role=”Administrator”
Resource=”/myBankingApp/loanApproval/submit”>
```

```
<fake:submitButton …/>
</isUserInRole>
```

setSecurityContext

The `setSecurityContext` tag is used to set the base resource for all elements on a JSP page. If you use `setSecurityContext`, the value of this tag will be prepended to all other resource attributes on the page. In addition, variables that should be set globally for the application context can be set in the body of this tag.

Any attributes set within the tag are passed to every OES API call. For example if you set `foo=1` in the security context and then use the `isAccessAllowed` tag, `foo=1` would be available to policies as an application context variable.

Table 4-6 setSecurityContext

Attribute	Return Type	Description	Required
Value	n/a	The value of the security context will be used to specify the prefix for all OES tags that have a resource attribute on the page.	Yes

setSecurityContext Concepts

Listing 4-7 shows the concepts of using setSecurityContext in a JSP.

Listing 4-7 Using setSecurityContext in a JSP

```
<%-- Set the security context for this page --%>
<ales:setSecurityContext value=”/mybank/loanApplicationForm”/>
 <ales:attribute name=”customer_name”
 value=”<%=request.getParameter(\”customerId\”)%=>”
</ales:setSecurityContext>
```

recordEvent

The `recordEvent` tag is an input tag that causes an audit message to be written to the audit log of OES. The body of this tag can also take an `attribute` tag, as described in “Attribute” on page 4-22. All attributes are added to the audit context as additional information for the audit event.

Table 4-7 recordEvent

Attribute	Return Type	Description	Required
Severity	n/a	The severity of the audit message. The possible values are: ERROR, FAILURE, or INFORMATIONAL. The default value is INFORMATIONAL	No
Message	n/a	The message to be passed to the audit log	YES

recordEvent Concepts

Listing 4-8 shows the concepts of using `recordEvent` in a JSP.

Listing 4-8 Using recordEvent in a JSP

```
<c:if test=${trade_submitted == true}>
 <%--record that the trade has been successfully committed to the system—%>
<ales:recordEvent message=”Trade successfully submitted to the system”>
 <%--include the person who submitted this trade in the audit log--%>
<attribute name=”traderId” value=”<%traderId%>”/>
 <%--Include the trading confirmation number in the audit log--%>
<attribute name=”trade_confirmation” value=”<%trade_confirmation%>
</ales:recordEvent>
</c:if>
```

Attribute

The `attribute` tag can be used by any of the other OES tags to pass extra attributes down in the application context. Any of these variables can then be used to write constraints against in OES policies.

Table 4-8 attribute

Attribute	Return Type	Description	Required
Name	n/a	The name of the attribute to set in the OES application context	YES
Value	n/a	The value of the attribute to set in the OES application context	YES

attribute Concepts

Listing 4-9 shows the concepts of using attribute in a JSP.

Listing 4-9 Using attribute in a JSP

```
<%-- set up the global context for this jsp page --%>
<ales:setSecurityContext value=”/mybank/loanApplicationForm”>
<%--An attribute to pass to the application context for all calls on this
page--%>
 <attribute name=”loanAppId” value=”<%=
request.getParameter(\”appId\”)%>”/>
</ales:setSecurityContext>
<ales:isAccessAllowed resource=”/creditScore” action=”view”
resultVar=”canViewCreditScore”>
<attribute name=”customerId”
value=”<%=request.getParameter(\”customerId\”)%>
<bankapp:displayCreditScore>
<else>
 <B>You are not authorized to view the customer’s credit score</B>
</ales:isAccessAllowed>
```

Securing Oracle Data Service Integrator

This section provides information about securing Oracle Data Service Integrator (ODSI) 3.2, formerly known as AquaLogic Data Services Platform (ALDSP). It includes the following topics:

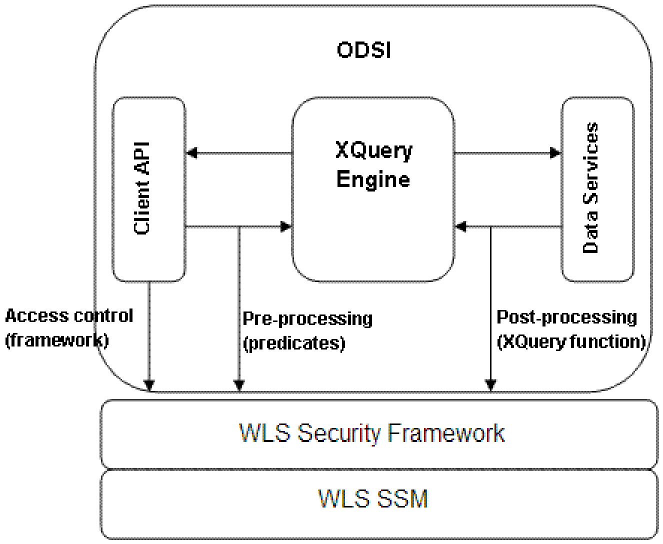
Overview

OES can be used to manage access control to entire ODSI data services or specific data service elements. In addition to simply returning an authorization decision that grants or prevents access, OES policies can return information to ODSI for use when performing pre- or post-processing data redaction. As a result, the user receives a redaction of the data requested.

For information about policies used for data redaction, see "Pre-Processing Data Redaction" on page 5-14 and "Post-Processing Data Redaction" on page 5-19.

The following diagram illustrates how OES components can be integrated with ODSI.

Figure 5-1 ODSI Integration Overview


ODSI
Client API
XQuery Engine
Data Services
Access control (framework)
Pre-processing (predicates)
Post-processing (XQuery function)
WLS Security Framework
WLS SSM

Use-Case

Integration with ODSI supports the following use-case scenario:

- OES governs access to ODSI data services and data service elements.
- OES policies are specified using the OES administration console, the Entitlements Administration Application, or the Policy Management API.
- The OES Administration Server is responsible for access to J2EE applications deployed on the ODSI WebLogic Server.
- The ODSI Administration Console continues to be the management point for data services.

Prerequisites

The document assumes the following:

- Installation of WebLogic Server and ODSI 3.2. Also supported is AquaLogic Data Services Platform (ALDSP) 3.0.1 or later.
- Installation of the OES Administration Server.

- Installation of the WLS SSM (for WebLogic Server 9.x, 10.0) on the ODSI machine as described in the SSM Installation and Configuration Guide. Be sure to make the necessary changes to `WLESarme.properties` as described in those chapters.
- Creation of the SSM instance. When setting up the instance add one of the following to the instance's `set-wls-env.bat|sh` and restart the server:

  On Windows:
  ```
  set WLES_JAVA_OPTIONS=%WLES_JAVA_OPTIONS%
  -Dales.dbAtnProvider.addDirectoryPrincipal=false
  ```

  On UNIX:
  ```
  WLES_JAVA_OPTIONS="$WLES_JAVA_OPTIONS
  -Dales.dbAtnProvider.addDirectoryPrincipal=false”
  ```

Integration Tasks

The major integration tasks are:

Note: In addition to the steps below, other tasks are required to define policies for data redaction. For pre-processing data redaction, see “Pre-Processing Data Redaction” on page 5-14. For post-processing data redaction, see “Post-Processing Data Redaction” on page 5-19.

1. Create the startWebLogic.cmd file for the ODSI domain as described in the SSM Installation and Configuration Guide.
2. Create a file named `security.properties` that specifies the ODSI security realm and copy it to the domain directory. The file should contain the following two entries:

   ```
   wles.realm=<aldsprealm>
   wles.default.realm=<aldsprealm>
   ```

   where `<aldsprealm>` is the name of the security realm.
3. Define the security providers as described in “Define Security Providers” on page 5-4.
4. In ODSI, enable security on the data service elements to be secured as described in “Enable ODSI Elements for Access Control” on page 5-4.
5. Define ODSI identities in OES. For instructions using the RTLApp sample application, see “Define ODSI Identities in OES” on page 5-5.
6. Define ODSI resources in OES. This chapter provides instructions for resources that represent RTLApp resources. See “Define ODSI Resources in OES” on page 5-6.

7. Define the Authorization and Role Mapping policies that secure RTLApp as described on "Define Policies for ODSI" on page 5-8.
8. Distribute the policies to the SSM by following the instructions in "Distribute Policies" on page 5-10.

Define Security Providers

The following table provides information about OES providers for securing ODSI data services.

Note: Before creating these providers in OES, use the WebLogic console to define a new security realm and add some providers to this realm.

Provider	Configuration Settings
ASI Adjudication Provider	Use the defaults for all settings.
Log4j Auditor	Use the defaults for all settings.
Database Authentication	Set the Control Flag to SUFFICIENT and the Identity Scope to `aldspusers`. Use the defaults for all other settings.
ASI Authorization	Set the Identity Scope to `aldspusers`. Use the defaults for all other settings.
WebLogic Credential Mapper	Deselect the **Credential Mapping Deployment Enabled** checkbox.
ASI Role Mapping	Set the Identity Scope to `aldspusers`. Use the defaults for all other settings.

Enable ODSI Elements for Access Control

Access control must be set on the data service elements to be secured so that OES is invoked to determine if access to the data should be granted. The following steps describe how to enable security on RTLApp data service elements to be secure with OES:

1. Open a browser and go to `http://<hostname>:<port>/dspconsole`. Then log in as an administrator.
2. Select **Lock & Edit** to enable editing.
3. Specify the dataspace to be controlled by OES:

   a. Select the **RetailDataspace** and click the **Security Configuration** tab.
   b. On the General tab, select **Enable Access Control** and **Enable JDBC Metadata Access Control**. Then click **Save**.

4. Specify the data services elements to be controlled by OES:
   a. Expand **RetailDataspace > RetailApplication > CustomerManagement > CustomerService** and select the **Security Configuration** tab.
   b. Select the **Secured Elements** tab. Then select **CUSTOMER/ORDERS/ORDER_SUMMARY/OrderDate** as an element to be secured. (This allows the element call to go to the security check.)
   c. Expand **RetailDataspace > RetailApplication > CustomerManagement > CustomerService** and click **CUSTOMER/ORDERS/ORDER_SUMMARY/OrderDate**.
   d. Select **Secured Elements Configuration**.
   e. Select **Remove** and click **Save** to save the changes.
   f. Click **Activate Changes**.

Define ODSI Identities in OES

To create the Identity directory and users:

1. In the OES Administration Console's left pane, select the **Identity** node and click **New** at the bottom of the right pane.
2. On the **Create Directory** dialog, enter the directory name (for example, `aldspusers`) and click **OK**.
3. Under the **Identity** node, select **Groups** and click **New** at the bottom of the right pane.
4. On the **Create Group** dialog, enter the group name (for example, `LDSampleUsers`) and click **OK**.
5. Under the **Identity** node, create the following users and add them to the `LDSampleUsers` group:

   `Jack` (password: weblogic)—RTLApp user
   `Steve` (password: weblogic)—RTLApp user

`Tim` (password: weblogic)—RTLApp user
`weblogic` (password: weblogic)—ldconsole administrator

Define ODSI Resources in OES

This section describes how to use the OES Administration Console to define the RTLApp and ODSI resources to be protected by OES.

RTLApp Application Resources

To create RTLApp application resources, perform the following steps:

1. Expand the **Resources** node and select **Resources**.
2. In the right pane, select **Policy** and click **New**.
3. On the **Create Resource** dialog, enter `aldsprealm` as the name, select **Binding** in the **Type** field, and click **Ok**.
4. Select the `aldsprealm` resource and click **Configure**.
5. On the **Configure Resource** dialog, select **Binding Application** in the **Type** field, select the **Distribution Point** checkbox and click **Ok**.
6. Modify the ASI Authorization and ASI Role Mapper providers as follows:
   – Set the Application Deployment Parent to `//app/policy/aldsprealm`
   – On the Bindings tab, bind to `//app/policy/aldsprealm`

ODSI Resources

Figure 5-2 shows the ODSI resource tree with all nodes expanded except the RTLApp node. You must create the resources shown in both Figure 5-2 and Figure 5-3.

When defining the resources, make sure the following resources are defined as “virtual resources”:

– `//app/policy/aldsprealm/consoleapp`
– `//app/policy/aldsprealm/dspconsole`
– `//app/policy/aldsprealm/ElectronicsWS/ld`
– `//app/policy/aldsprealm/RetailDataspace/ld`
– `//app/policy/aldsprealm/RTLApp/url`
– `//app/policy/aldsprealm/shared`

Figure 5-2 ODSI Resource Tree with RTLApp Node Collapsed

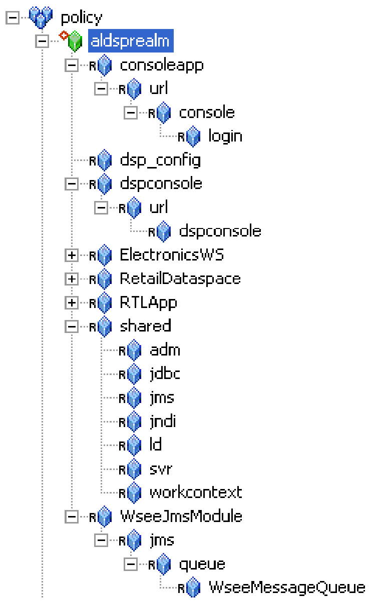
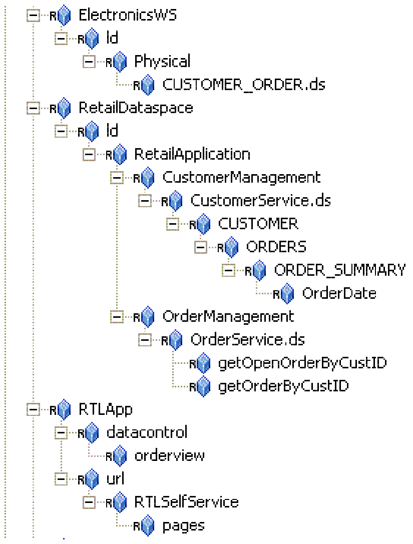


Figure 5-3 ODSI Resource Tree with RTLApp Node Expanded



Define Policies for ODSI

This section describes how to create the Authorization and Role Mapping policies to protect the ODSI 3.2 sample application.

Authorization Policies

Define the Authorization policies shown in Table 5-1.

Table 5-1 Authorization Policies for ODSI 3.2

Policies	Description

grant(any, [//app/policy/aldsprealm/shared/adm,//app/policy/aldsprealm/shared/svr], //role/Admin) if true;
 	
grant(any, //app/policy/aldsprealm/WseeJmsModule, //role/Admin) if true;
 	
grant(any, //app/policy/aldsprealm/shared, //role/Admin) if true;
	Grants Admin Role and/or weblogic user permission to boot WLS and perform administrative tasks.

grant(any, //app/policy/aldsprealm/dspconsole, //role/Admin/) if true;
 	
grant([//priv/GET,//priv/POST], //app/policy/aldsprealm/dspconsole/url/dspconsole, //role/Everyone) if true;
	Gives WebLogic full access to the DSP console.

grant([//priv/GET,//priv/POST], //app/policy/aldsprealm/consoleapp/url/console/login, //role/Everyone) if true;
 	
grant(//priv/receive, //app/policy/aldsprealm/WseeJmsModule/jms/queue/WseeMessageQueue, //role/Everyone) if true;
 	
grant(//priv/reserve, //app/policy/aldsprealm/shared/jdbc, //role/Everyone) if true;
 	
grant(//priv/lookup, [//app/policy/aldsprealm/shared/jdbc,//app/policy/aldsprealm/shared/jndi], //role/Everyone) if true;
 	
grant(//priv/read, //app/policy/aldsprealm/shared/workcontext, //role/Everyone) if true;
 	
grant(//priv/lookup, //app/policy/aldsprealm/shared/jms, //role/Everyone) if true;
 	
grant(//priv/send, //app/policy/aldsprealm/shared/jms, //role/Everyone) if true;
	Grants Everyone role (including the anonymous user) access all of the shared open resources.

grant(any, //app/policy/aldsprealm/RTLApp/url, [//sgrp/aldspusers/LDSampleUsers/, //role/Admin]) if true;	Allows all users to access the sample application.

Table 5-1 Authorization Policies for ODSI 3.2

Policies	Description

grant(any, //app/policy/aldsprealm/shared/ld, //role/Everyone) if true;
grant(any, [//app/policy/aldsprealm/shared/svr,//app/policy/aldsprealm/shared/adm,//app/policy/aldsprealm/consoleapp], //role/Everyone) if true;
grant(any, //app/policy/aldsprealm/ElectronicsWS/ld, //app/policy/aldsprealm/RetailDataspace/ld [//role/Admin,//sgrp/aldspusers/LDSampleUsers/]) if true;
	Grants permission for data services.

Deny(any, //app/policy/aldsprealm/RetailDataspace/ld/RetailApplication/CustomerManagement/CustomerService.ds/CUSTOMER/ORDERS/ORDER_SUMMARY/OrderDate, //user/aldspusers/Steve/) if true;	Deny the Steve user to access the dataservice element.

Role Policies

Define the role policies shown in Table 5-2.

Table 5-2 Role Mapping Policies for ODSI 3.2 Sample Application

Policies	Description
grant(//role/Everyone, //app/policy/aldsprealm, //sgrp/aldspusers/allusers/) if true;	Allows Everyone role to be used in `aldsprealm` Identity directory.
grant(//role/Admin, //app/policy/aldsprealm, //user/aldspusers/weblogic/) if true;	Grants the weblogic user Admin role within `aldsp` realm.

Distribute Policies

After defining the identities, resources, and policies to secure the ODSI data service(s), deploy the results as follows:

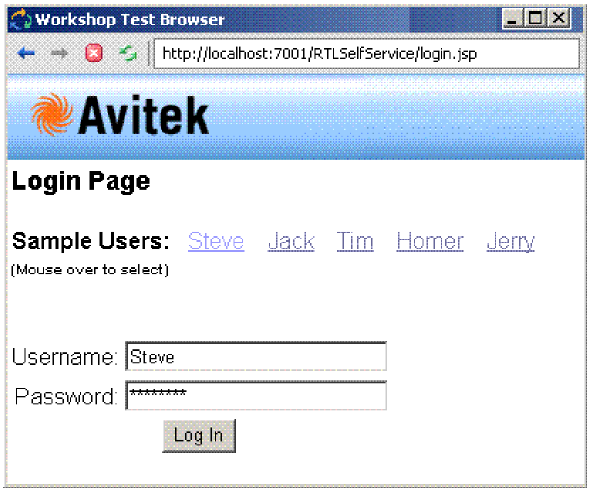
1. In the OES Administration Console's left pane, select the **Deployment** node.
2. On the **Policy** tab in the right pane, select the checkbox before the name of the ODSI resource parent and click Distribute Policy.

3. If you made any changes to the SSM configuration used to secure the ODSI domain, display the **Configuration** tab and select the checkbox for the SSM configuration. Then click **Distribute Configuration Changes**.

After this, you can test access to RTLApp using the following steps:

1. Start a WebLogic Server instance by changing to the `<odsi_home>\samples\domains\aldsp` directory and running `startWebLogicALES.cmd|sh`.
2. Access the RTLApp by pointing a browser to `http://<hostname>:<port>/RTLSelfService` where `<hostname>` is the machine on which RTL application is running. The browser is redirected to the authentication page (see Figure 5-4).

Figure 5-4 Authentication Page



http://localhost:7001/RTLSelfService/login.jsp

Login Page

Sample Users: Steve Jack Tim Homer Jerry

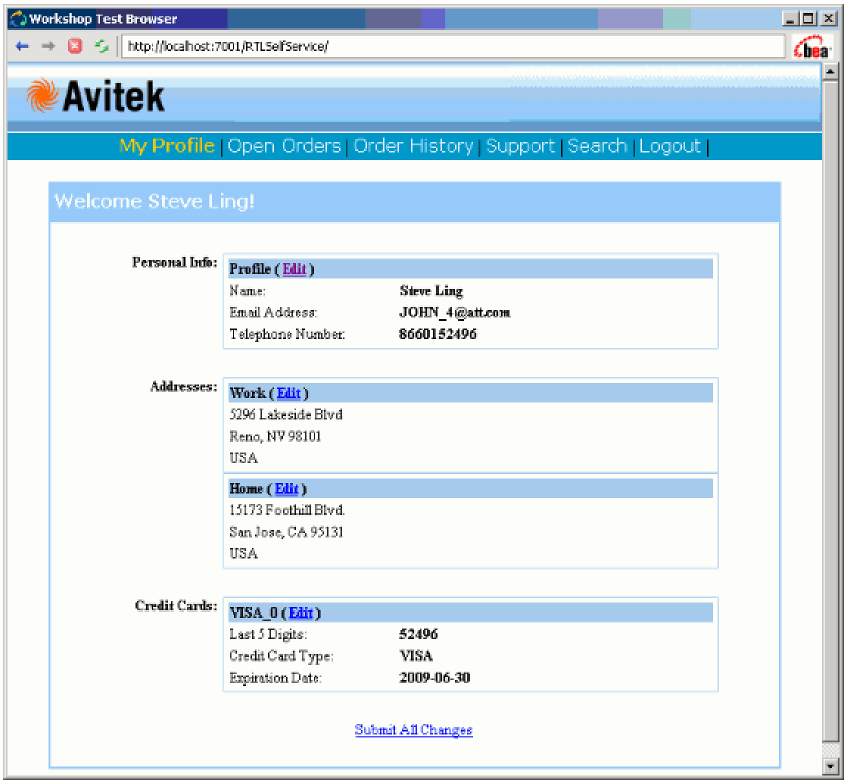
(Mouse over to select)

Username: Steve

Password: ********

3. Log in as Steve and access should be granted to the Profile Page (see Figure 5-5).

Figure 5-5 Profile Page


Workshop Test Browser
http://localhost:7001/RTLSelfService/
Avitek
My Profile | Open Orders | Order History | Support | Search | Logout |

Welcome Steve Ling!

Personal Info:

Profile (Edit)

Name:	Steve Ling
Email Address:	JOHN_4@att.com
Telephone Number:	8660152496

Addresses:

Work (Edit)
5296 Lakeside Blvd
Reno, NV 98101
USA

Home (Edit)
15173 Foothill Blvd.
San Jose, CA 95131
USA

Credit Cards:

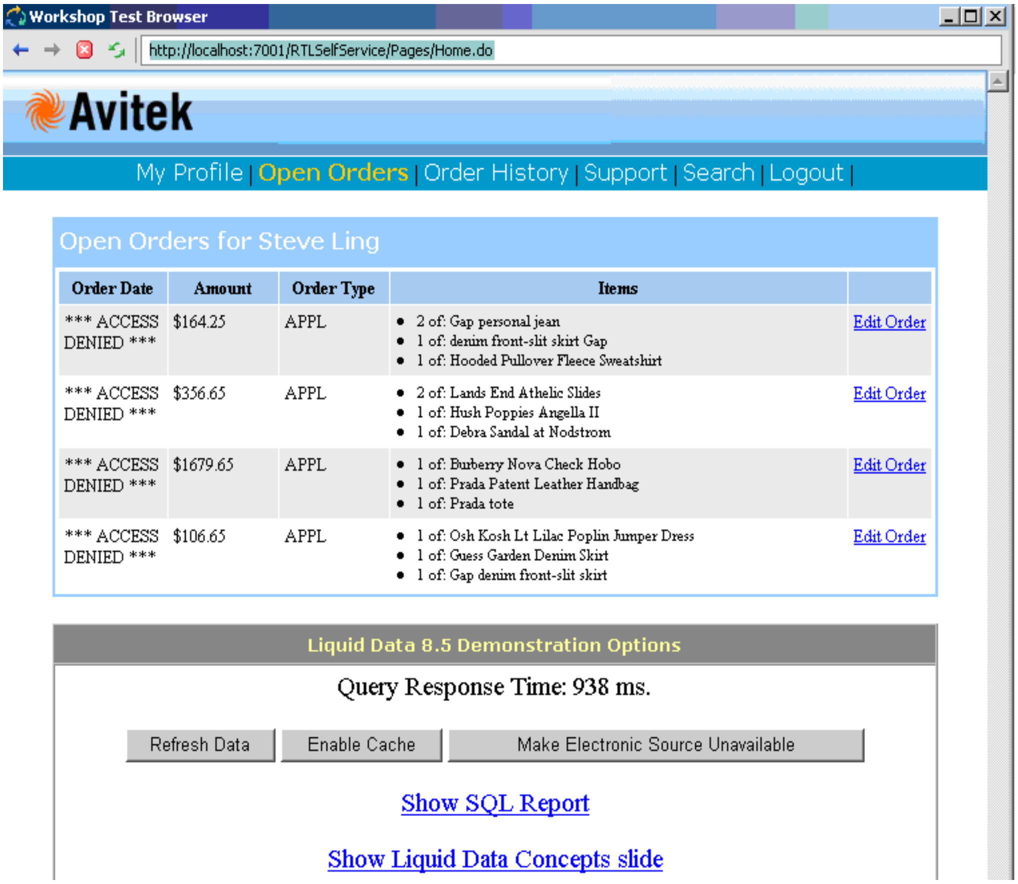
VISA_0 (Edit)

Last 5 Digits:	52496
Credit Card Type:	VISA
Expiration Date:	2009-06-30

Submit All Changes

4. Select **Open Orders Page**. Open orders should be visible (see Figure 5-6), while access to Order Data should be denied.

Figure 5-6 Open Orders Page



Workshop Test Browser
http://localhost:7001/RTLSelfService/Pages/Home.do
Avitek
My Profile | Open Orders | Order History | Support | Search | Logout |

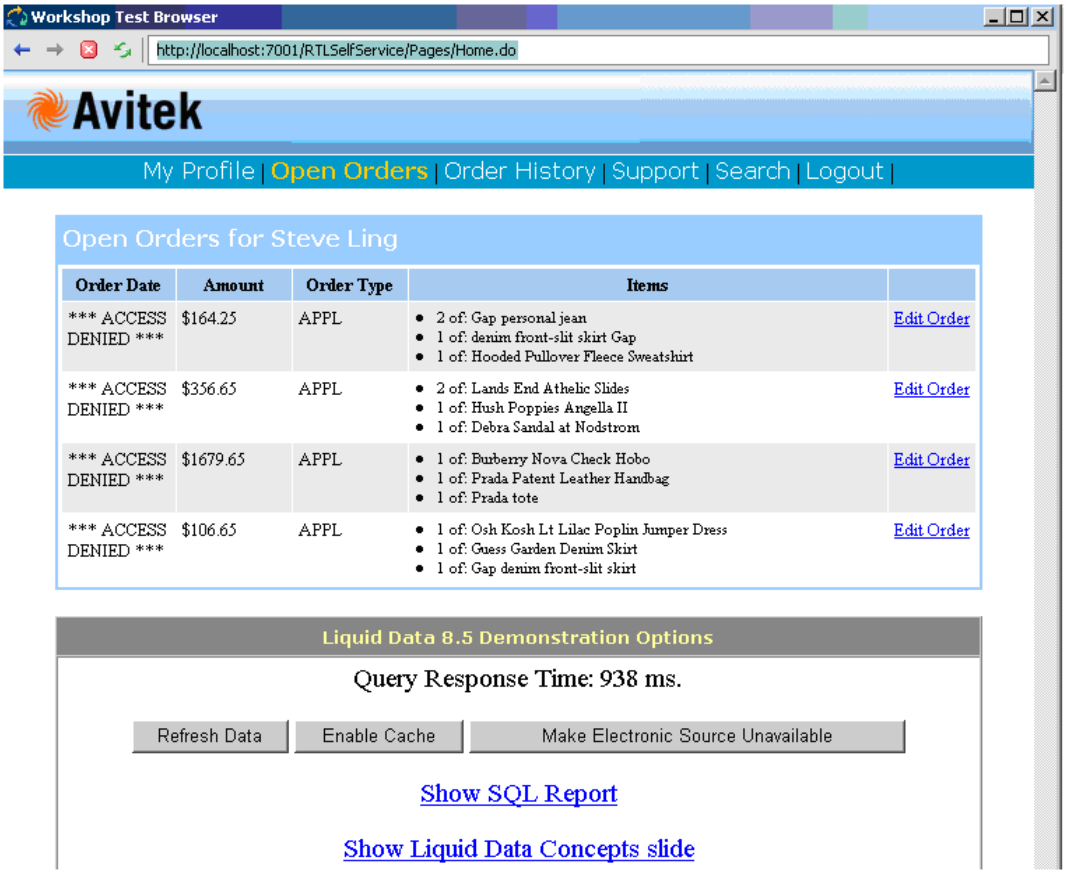
Open Orders for Steve Ling

Order Date	Amount	Order Type	Items	
*** ACCESS DENIED ***	$164.25	APPL	• 2 of: Gap personal jean • 1 of: denim front-slit skirt Gap • 1 of: Hooded Pullover Fleece Sweatshirt	Edit Order
*** ACCESS DENIED ***	$356.65	APPL	• 2 of: Lands End Athelic Slides • 1 of: Hush Poppies Angella II • 1 of: Debra Sandal at Nodstrom	Edit Order
*** ACCESS DENIED ***	$1679.65	APPL	• 1 of: Burberry Nova Check Hobo • 1 of: Prada Patent Leather Handbag • 1 of: Prada tote	Edit Order
*** ACCESS DENIED ***	$106.65	APPL	• 1 of: Osh Kosh Lt Lilac Poplin Jumper Dress • 1 of: Guess Garden Denim Skirt • 1 of: Gap denim front-slit skirt	Edit Order

Liquid Data 8.5 Demonstration Options

Query Response Time: 938 ms.

Refresh Data
Enable Cache
Make Electronic Source Unavailable
Show SQL Report
Show Liquid Data Concepts slide


Workshop Test Browser
http://localhost:7001/RTLSelfService/Pages/Home.do
Avitek
My Profile | Open Orders | Order History | Support | Search | Logout |

Open Orders for Steve Ling

Order Date	Amount	Order Type	Items	
*** ACCESS DENIED ***	$164.25	APPL	• 2 of: Gap personal jean • 1 of: denim front-slit skirt Gap • 1 of: Hooded Pullover Fleece Sweatshirt	Edit Order
*** ACCESS DENIED ***	$356.65	APPL	• 2 of: Lands End Athelic Slides • 1 of: Hush Poppies Angella II • 1 of: Debra Sandal at Nodstrom	Edit Order
*** ACCESS DENIED ***	$1679.65	APPL	• 1 of: Burberry Nova Check Hobo • 1 of: Prada Patent Leather Handbag • 1 of: Prada tote	Edit Order
*** ACCESS DENIED ***	$106.65	APPL	• 1 of: Osh Kosh Lt Lilac Poplin Jumper Dress • 1 of: Guess Garden Denim Skirt • 1 of: Gap denim front-slit skirt	Edit Order

Liquid Data 8.5 Demonstration Options

Query Response Time: 938 ms.

Refresh Data
Enable Cache
Make Electronic Source Unavailable
Show SQL Report
Show Liquid Data Concepts slide

Pre-Processing Data Redaction

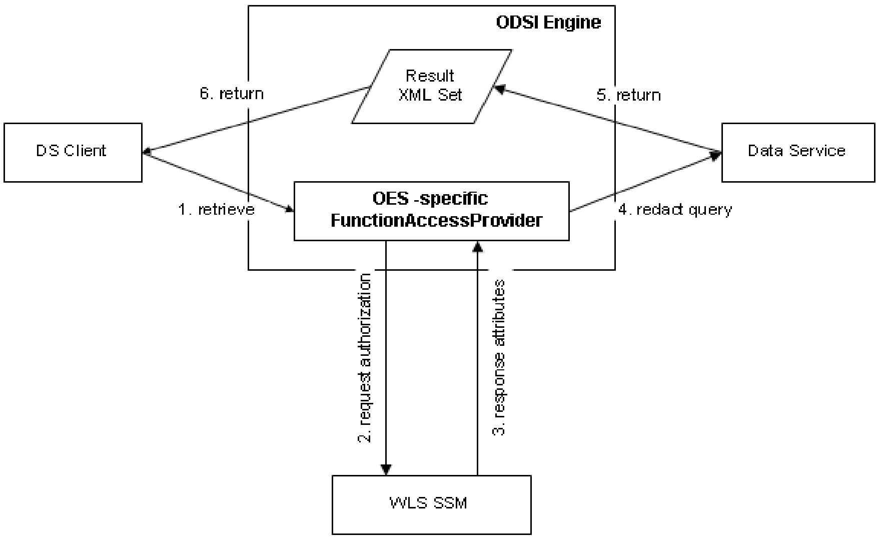
Pre-processing data redaction involves modifying the client request in some way before ODSI forwards the request to the data service. This is achieved through the use of an OES plug-in that calls the Java API to authorize the user request, gets the response attributes from the authorization response, and returns them to ODSI.

Here is the sequence of events that occur during pre-processing redaction:

1. ODSI receives a data service client request and invokes the OES plug-in.

2. OES plug-in calls the OES Java API for authorization. The authorization decision returns additional predicates as responses.
3. OES plug-in returns the authorization decision and responses to ODSI.
4. ODSI adds the predicates and/or function name and calls the data service
5. ODSI engine receives the results from the data service and returns it to the client.

Figure 5-7 Overview of Pre-Processing Solution



Pre-Processing Response Types

OES supports two types of responses that return information to ODSI in the form of security predicates. They can be used separately or together.

- **Replacement Function**

  The replacement function is called instead of the original user-requested function. For example, if a user requests a call to the `getOrders` function and authorization is granted, the `getOrdersThatAmountLessThan1000` replacement function is returned to ODSI and is called to return a lesser result than the original.

- **XQuery Expression**

  The returned information is the name of a XQuery Expression that is applied to the invoked function or its replacement, as a predicate. Multiple predicates may be returned and applied to the function.

  For example, if a user request calls the `getOrders` function and authorization is granted, OES returns "`./order/amount < 1000`" as a predicate. This expression is applied to the `getOrders` function so that only a subset of orders will be returned to the user.

The OES plug-in (`com.bea.security.ales.aldsp.ALESFunctionAccessProvider`) calls the Java SSM to do authorization and return response attributes to ODSI. For example, consider the following policy:

```
grant (
//priv/ALDSP_QUERY,
//app/policy/aldsprealm/RetailDataspace/ld/RetailApplication/OrderManageme
nt/getOrderByCustID,
//user/aldspusers/Steve/

) if report_as("aldsp_replacement_function",
"{ld:RetailApplication/OrderManagement/OrderService}getOrdersThatAmountLes
sThan1000")
```

This policy grants the *system* user the `ALDSP_QUERY` privilege on the `OrderView` data service's `getOrders` function. If the authorization decision is *true*, it returns the `aldsp_replacement_function` attribute with a value of `getOrdersThatAmountLessThan1000`. ODSI then calls that replacement function (instead of the original). This function must the same signature; no additional verifications are performed at runtime.

Required OES Response Attributes

One of three OES response attributes must be used to return replacement functions or XQuery expressions. They must be returned by the OES evaluation functions `report` and `report_as` or by user-defined evaluation functions.

OES response attribute names are all lower case.

- **aldsp_replacement_function**

This attribute provides the name of an ODSI replacement function that ODSI calls instead of the original one. The function name should be fully-qualified, including the namespace:

```
report_as("aldsp_replacement_function",
"{ld:RetailApplication/OrderManagement/OrderService}:getOpenOrdersByCus
tID")
```

- **aldsp_xquery_expression**

  This attribute defines an XQuery expression that will be used as a filter in ODSI. For example:

  ```
  if report_as(“aldsp_xquery_expression”, “./order/amount < 1000”)
  ```

  The attribute value can be a list. If so, ODSI applies the AND operator to the list values.

- **aldsp_namespace_binding**

  This attribute defines the namespace mapping for prefixes and is used in XQuery expressions. For example:

  ```
  report_as(“aldsp_namespace_binding”, “f1=http://com.bea.security”) and
  report_as(“aldsp_xquery_expression”, “./f1:region eq 'west'”)
  ```

  For “`f1=http://com.bea.security`” to be usable, the namespace map should contain the entry that maps the string prefix (`f1`) to the namespace (`http://com.bea.security`).

Additional Integration Tasks

Additional tasks are required implement pre-processing data redaction.

- “Modify the Start WebLogic Script” on page 5-17
- “Write Replacement Functions” on page 5-18
- “Define Policies for Replacement Functions” on page 5-18
- “Define Policies for XQuery Expressions” on page 5-18
- “Define Namespace Bindings” on page 5-19

Modify the Start WebLogic Script

Modify `set-wls-env.bat|sh` in the WLS SSM instance directory. To do this:

1. Navigate to the WLS SSM instance directory and open `set-wls-env.bat|sh` in an editor.
2. Add `ldintegration.jar` to the end of `WLES_POST_CLASSPATH` environment variable.

3. Add the JVM option `-Dcom.bea.ld.security.FunctionAccessQuery=com.bea.security.ales.aldsp.ALESFunctionAccessProvider` to `WLES_JAVA_OPTIONS`.

Write Replacement Functions

Replacement functions must be implemented on the target data service and have the same return type and number/type of parameters as the function being replaced. For example, to restrict `OrderView.getOrders` to return only orders totalling less than $1000.00, write an XQuery function to implement the restriction. This function must have the same return type, and number types of parameters as `getOrders`.

Define Policies for Replacement Functions

To use a replacement function to protect data services, define a policy that allows access to the target data service's function and returns the `aldsp_replacement_function` attribute with the name of the replacement function. There are no additional access control checks performed for the replaced function.

Note: All policies for pre-processing data redaction must use the `ALDSP_QUERY` privilege.

For example, the following policy returns a replacement function named `getOrdersThatAmountLessThan100`:

```
grant (
//priv/ALDSP_QUERY,
//app/policy/aldsprealm/RetailDataspace/ld/RetailApplication/OrderManageme
nt/getOrderByCustID,
//user/aldspusers/Steve/
) if report_as("aldsp_replacement_function",
"{ld:RetailApplication/OrderManagement/OrderService}getOrdersThatAmountLes
sThan1000")
```

Define Policies for XQuery Expressions

To use an XQuery expression, define a policy that returns `aldsp_xquery_expression` and the name of an XQuery expression. For example:

```
grant(//priv/ALDSP_QUERY,//RTLApp/ld/DataServices/RTLServices/OrderView.ds
/getOrders,//user/asi/anonymous/)if report_as(“aldsp_xquery_expression”,
“./order/amount < 1000”)
```

Note: All policies for pre-processing data redaction must use the `ALDSP_QUERY` privilege.

Define Namespace Bindings

You must define namespace binding used in an XQuery expression. (Namespace bindings are not used for replacement function names; they must be fully qualified, including the namespace.) For example, consider the following policy:

```
grant (
//priv/ALDSP_QUERY,
//app/policy/RTLApp/ld/DataServices/RTLServices/OrderView.ds/getOrders,//u
ser/asi/anonymous/
) if report_as(“aldsp_xquery_expression”, “./ns1:order/amount < 1000”)
```

In this case, you need to define the mapping of namespace `ns1` and return it. Therefore, you need to add another response attribute, as follows:

```
grant (
//priv/ALDSP_QUERY,
//app/policy/RTLApp/ld/DataServices/RTLServices/OrderView.ds/getOrders,
//user/asi/anonymous/
) if report_as(“aldsp_xquery_expression”, “./ns1:order/amount < 1000”) and
report_as(“aldsp_namespace_binding”, “ns1=http://com.bea.security”)
```

Post-Processing Data Redaction

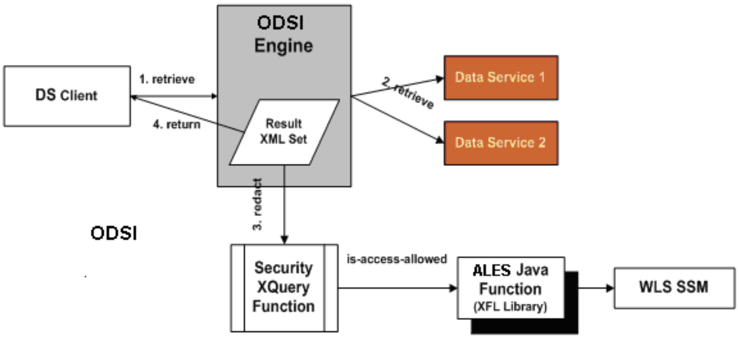
Post-processing data redaction is achieved by invoking a security XQuery function after the ODSI engine retrieves the data from the data service. The XQuery function determines whether to grant access and return the data. **Note**: This approach may not be suitable for fast operations or very large data sets.

Here is the sequence of events that occur during post-processing redaction:

1. Client sends a data retrieving request to ODSI.
2. ODSI engine retrieves the data from the data service(s).
3. ODSI invokes the relevant security XQuery function for the data element.
4. Security XQuery function invokes an OES Java method, passing in the resource name and one or more attribute names/values.
5. OES Java method invokes the WLS SSM and gets the authorization result and optional responses defined in the queries.

6. OES Java method returns the authorization decision and a set of responses to the security XQuery function.
7. XQuery function may use the OES-supplied responses to make the decision. For example, the policy decision may evaluate as *true*, but based on the responses the function may return *false*.

Figure 5-8 Overview of the Post-Processing Solution



ODSI Security XQuery Functions

The ODSI security XQuery functions enable you to specify custom security policies that can be applied to data elements. The functions are useful for creating policies based on data values. For example, to deny access to an element if the order amount exceeds a given threshold.

OES provides two Java methods that can be called by security XQuery functions. These methods invoke the WLS SSM which determines whether the access request should be granted.

The OES Java methods can be used instead of, or in addition to, the following ODSI access control function extensions:

- `fn-bea:is-access-allowed`
- `fn-bea:is-user-in-group`
- `fn-bea:is-user-in-role`
- `fn-bea:userid`

For example, in the XQuery function shown below, `fn-bea:is-access-allowed` could be replaced with one of the OES Java method.

Listing 5-1 Example Security XQuery Function

```
declare namespace demo="demo";
declare namespace retailerType="urn:retailerType";
declare function demo:secureOrders($order as
element(retailerType:ORDER_SUMMARY) ) as xs:boolean {
if (fn-bea:is-access-allowed("LimitAccess",
"ld:DataServices/RTLServices/OrderSummaryView.ds")) then
fn:true()
:
:
```

Note: Because the security XQuery function depends on the data service schema. You must create the security XQuery function based on the custom data service schema.

Custom security XQuery functions must be created in Workshop, instead of the ODSI console, because the console compiler does not access the custom functions used in it.

OES Java Methods

The two OES Java methods are contained in `com.bea.security.ales.aldsp.AccessController` class.

- **is_access_allowed**

  ```
  public static boolean is_access_allowed(String resource, String[]
  attributeNames, String[] attributeValues)
  ```

- **is_access_allowed_with_response_attributes**

  ```
  public static String[] is_access_allowed_with_response_attributes
  (String resource, String[] attributeNames, String[] attributeValues)
  ```

Parameters

These methods have three parameters as shown in the following example:

```
let $result := f1:is_access_allowed_with_response_attributes
              ("RTLApp/datacontrol/orderview",
              ("totalorderamount"),
              (fn:string(fn:round ($order/TotalOrderAmount)))) return
```

- The first parameter is the resource name represents the secured ODSI data element. This resource name must be specified in the OES policy.

- The second parameter is a string array containing the attribute name(s).
- The third parameter is a string array containing the attribute value(s).'

  For example, assume that the second parameter contains the array `an1,an2` and the third parameter contains the array `av1, av2`. This indicates two attributes. The first attribute's name is `an1` and its value is `av1`. The second attribute's name is `an2` and its value is `av2`.

  These attributes and values are then evaluated in the context of the OES policy.

  Note: If the data type of the attribute value is not *string*, convert it to *string* via *fn:string()*.

  In addition, OES attributes support only integer numbers. If the attribute value is a decimal number, truncate it by using `fn:round()` before converting to `string`.

Return Values

The `is_access_allowed` method returns a boolean value (*true* or *false*) representing the access permission. You can return this value directly to the security XQuery function or do some additional operation based on the result.

The `is_access_allowed_with_response_attributes` method returns:

- the access permission decision, either `true` or `false`.
- the response attributes. One example is (`'true', 'ALESResponse', 'ran1', 'rav1_1', 'rav1_2', 'ALESResponse', 'ran2', 'rav2'`), where:
  - `true` is the access permission.
  - `ALESResponse` is a response attribute separator.
  - `ran1` and `ran2` are response attribute names.
  - `rav1_1` and `rav1_2` are the value of response attribute `ran1`.
  - the response attribute `ran1` is a list value.
  - `rav2` is the value of response attribute `ran2`.
  - the response attribute `ran2` is a single value.

You can test the access permission by comparing the first element of the string array with `true` or `false`.

Note: In addition, you can use the response attribute value to implement additional logic, as described below.

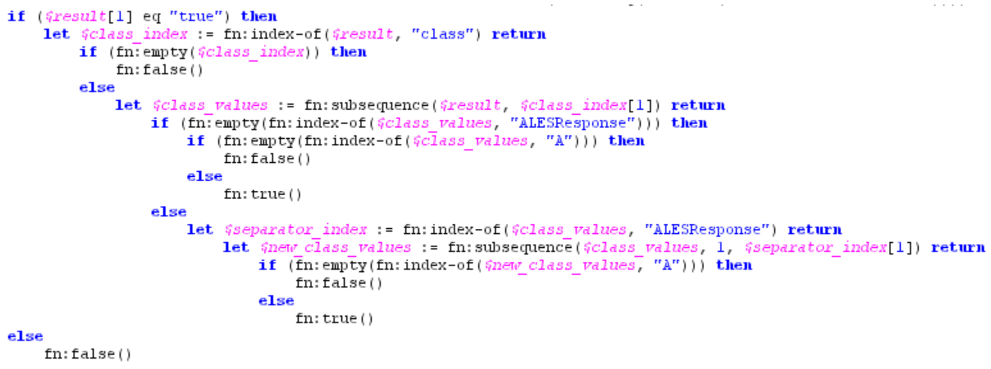
Policies Returning Attributes to ODSI

If you use the `is_access_allowed_with_response_attributes` method, you can create a policy that returns response attributes and then test those attributes.

As described in Using Response Attributes, response attributes are typically specified using built-in evaluation functions that report name/value pairs. There are two functions for returning attributes: `report()` and `report_as()`. These functions always return *true* (if there are no errors), and their information is passed to your application as response attributes embedded within the `ResponseContextCollector`.

The `report_as` function allows you to write the policy to specify both the attribute name and value. For example, `report_as("class", "A")`. The security XQuery function can then test the return response attributes as shown in Figure 5-9:

Figure 5-9 Testing Return Response Attributes



```
if ($result[1] eq "true") then
    let $class_index := fn:index-of($result, "class") return
        if (fn:empty($class_index)) then
            fn:false()
        else
            let $class_values := fn:subsequence($result, $class_index[1]) return
                if (fn:empty(fn:index-of($class_values, "ALESResponse"))) then
                    if (fn:empty(fn:index-of($class_values, "A"))) then
                        fn:false()
                    else
                        fn:true()
                else
                    let $separator_index := fn:index-of($class_values, "ALESResponse") return
                        let $new_class_values := fn:subsequence($class_values, 1, $separator_index[1]) return
                            if (fn:empty(fn:index-of($new_class_values, "A"))) then
                                fn:false()
                            else
                                fn:true()
else
    fn:false()
```

The `report_as` function loads a named response attribute with a value that specifies an attribute, constant, or a string literal. When returning multiple values, the response attribute is returned as a list.

The `report()` and `report_as()`functions are not policy constraints. The attributes are returned only if the authorization decision is *true*.

Defining a Security XQuery Function

Use Workshop to add a security XQuery function, as follows:

1. Import the OES Java method via an XFL library in the current Workshop application, as described in "Integrating the OES Java Methods" on page 5-24.

   The OES Java method is an XFL function. The XQuery Function Library (XFL) is a facility for providing auxiliary functions across multiple data services.

2. To obtain the data service elements, import the namespace of the data service XML schema.

3. Add an XQuery Function and specify the root element of the data service as the parameter.

4. In of the XQuery Function, specify all of the attributes that may be returned by the OES policies protecting the resource. Use two string arrays for names and values.

   If OES policies for the affected DSP resource require any context parameters to be passed with the request, those parameters should be extracted in the custom XQuery Security function and passed to the SSM via the OES Java function.

   The OES Java method is able to determine the authenticated subject to use for authorization, and you do not need to supply it.

5. Invoke the OES Java Function with the resource name, attributes, and values.

Integrating the OES Java Methods

Before you can integrate the OES Java methods into the ODSI security XQuery function, you must configure the WLS SSM to protect the ODSI domain, as outlined in "Integration Tasks" on page 5-3.

After you have done this, then:

1. Configure and distribute the policy in OES:

   a. Define a resource to indicate the current data element of data service.

   b. Define a policy for the resource. If necessary, declare some attributes, and use them in the policy constraints. These attributes must later be passed into the OES Java method in Step 3.d.

   c. Distribute the policy change.

2. Import the OES Java method as an XFL library in the current Workshop application:

   a. Copy `alesxfl.jar` and `api.jar` from the WLS SSM `lib` directory to the ODSI application's `DSP-INF/lib` directory.

   b. In the ODSI application, right-click the node of the data service project and select **Import Source Metadata**.
   c. Select **Java Function as the Data Source Type** and click **Next**.
   d. Expand `alesxfl.jar` and select the class `com.bea.security.ales.aldsp.AccessController.class`.
   e. In the next page, based on your use case, select either `is_access_allowed` or `is_access_allowed_with_response_attributes`, and finish the wizard.

3. Use Workshop to create a security XQuery function in the ODSI application:
   a. Open the XFL file created in Step 2.
   b. Import the namespace of the data service.
   c. Add an XQuery function and define one parameter whose type is the whole data service.
   d. In the XQuery function, supply the OES Java method with the resource name, and the attributes and values you want to test. For example:

```
let $result := f1:is_access_allowed_with_response_attributes
              ("RTLApp/datacontrol/orderview",
              ("totalorderamount"),
              (fn:string(fn:round ($order/TotalOrderAmount)))) return
```

   The first parameter is the resource name as defined in OES. The second parameter is a string array that contains attribute names. In the example, there is only one attribute, named `totalorderamount`. The third parameter is a string array that contains attribute values.

   A detailed example is provided in "OES Security XQuery Function" on page 5-26.

4. Specify the data service element to protected:
   a. Log in to the ODSI console.
   b. Click **Lock & Edit**.
   c. Select **Security Configuration** on the bottom right.
   d. Select proper secured element from data space navigation tree and add the XQuery security function in right panel, e.g. `ld:Physical/ALES_ACCESS_CONTROL` for namespace and `secureOrders` for local name.
   e. Click **Activate Changes**.

5. Redeploy the application in the Weblogic Server Administration Console:
   a. Log in to the Weblogic Server Administration Console.
   b. Expand the **Deployments | Applications** node and select the ALDSP application node.
   c. In right tab, select the Deploy tab.
   d. Click the Redeploy Application button.

      When the status is Success, the application has been redeployed.
6. Restart the WebLogic Server.

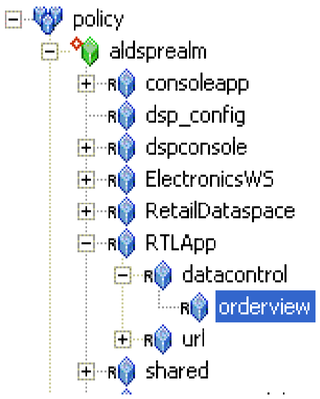
OES Security XQuery Function

In this example, based on the RTLApp that is shipped by ODSI 3.2, the data service `OrderView` is configured with a security XQuery function to protect its data elements. It is assumed that the application RTLApp has been deployed on an ODSI domain that is protected by the WLS SSM.

The integration example follows these steps:

1. Configure and distribute the policy in OES:
   a. In the OES Administration console, define resources named `datacontrol` and `orderview` under the `RTLApp` resource, as shown in Figure 5-10.

Figure 5-10 ODSI Resource Tree



   b. Define a dynamic attribute named `totalorderamount` whose type is integer.

c. Define and distribute the following an authorization policy:

```
grant( view, //app/policy/aldsprealm/RTLApp/datacontrol/orderview,
//sgrp/aldspusers/LDSampleUsers/) IF (totalorderamount < 1000) and
report_as (“class”,”A”);
```

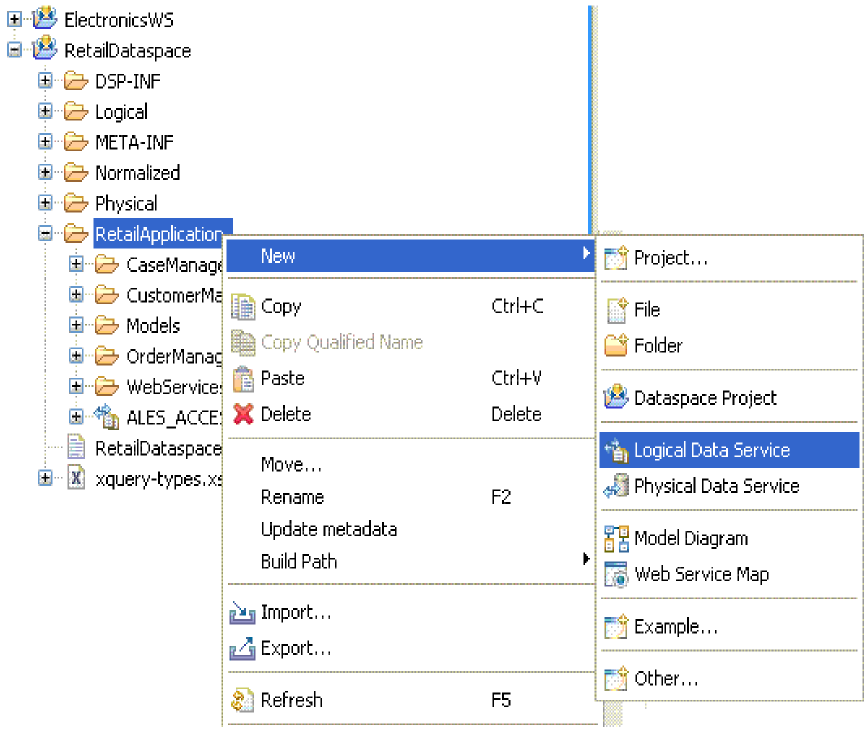
The privilege is view. The subject is `LDSampleUsers`. The constraint is `totalorderamount < 1000`. Response attributes are returned via report_as("class", "A").

2. Import the OES Java function as an XFL library in the current Workshop application:

a. Copy `alesxfl.jar` and `api.jar` from the WLS SSM `lib` directory to the ODSI application’s `DSP-INF/lib` directory.

b. In the ODSI 3.2 application, right-click **RetailDataspace** > **RetailApplication** folder and select **New** > **Physical Data Service**, as shown in Figure 5-11.

Figure 5-11 Import OES Java Function



c. Select `Java Function` as the **Data Source Type** and click **Next**.

d. Locate and expand `alesxfl.jar`. Then select the class `com.bea.security.ales.aldsp.AccessController.class`.

e. On the next page, select the `is_access_allowed_with_response_attributes` method and click **Next**.

f. Enter `ALES_ACCESS_CONTROL` for the new data service name and finish the wizard.

3. Add a security XQuery function in ALES_ACCESS_CONTROL.ds, as shown in Figure 5-12.

   The following bullet points explain the function shown in the figure:

   – Line 36: define a security XQuery function **secureOrders**.

   – Line 37: invoke the OES Java method. The first parameter is the resource name as defined in OES. The second parameter is a string array that contains attribute names. In the example, there is only one attribute, named **totalorderamount**. The third parameter is a string array that contains attribute values.

   – Line 39: the **TotalOrderAmount** element type is xsd:decimal. The **fn:round()** function converts the element into a integer. The **fn:string()** function converts the element into a string.

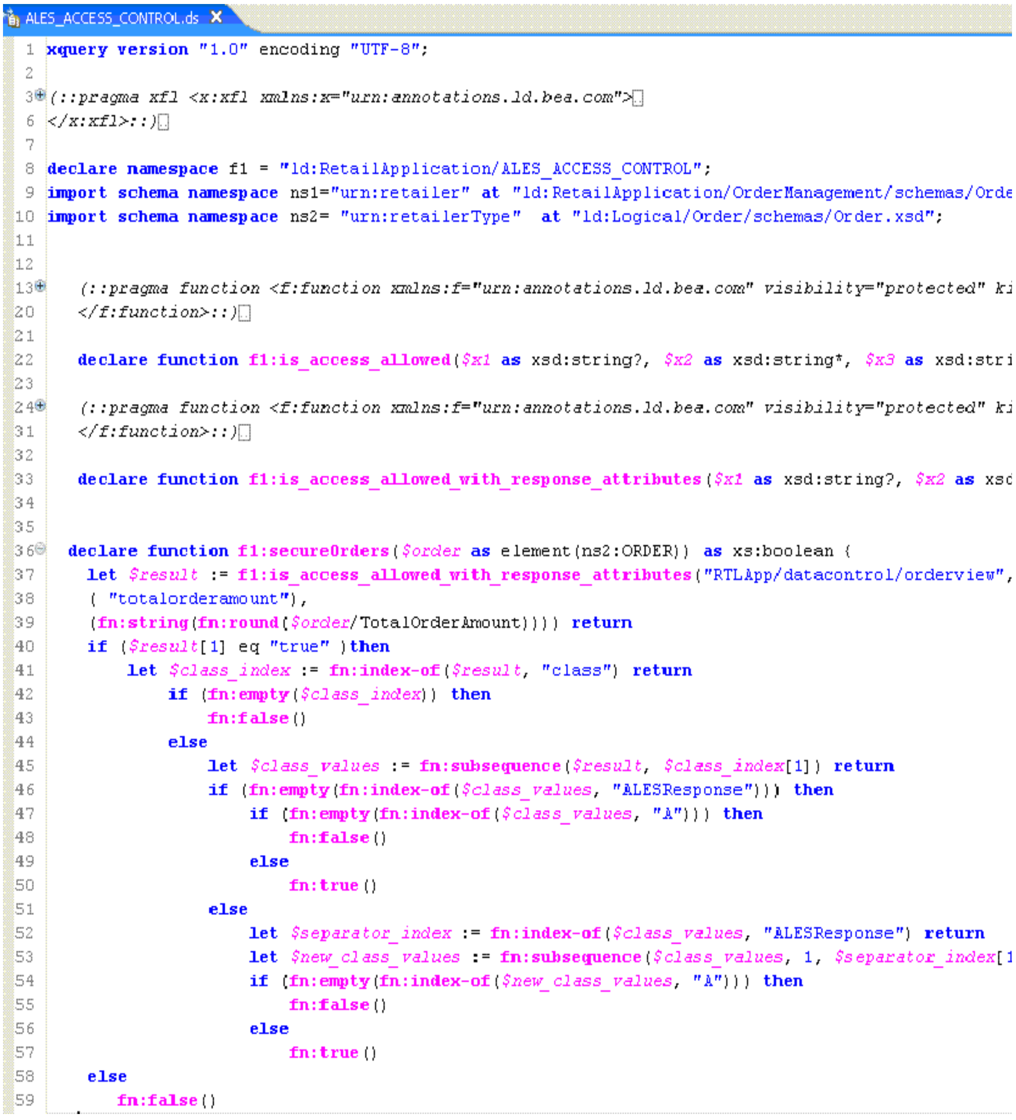
   – Line 40: if the first element is true, it indicates that the current operation is permitted.

   – Line 41: find the response attribute class (defined in step d on page 5-25).

   – Line 42: if the response attribute class is not found, return false.

   – Line 44 to 57: check if the response attribute class contains the value **A**.

Figure 5-12 Security XQuery Function for ODSI 3.2



```
xquery version "1.0" encoding "UTF-8";

(::pragma xfl <x:xfl xmlns:x="urn:annotations.ld.bea.com">
</x:xfl>::)

declare namespace f1 = "ld:RetailApplication/ALES_ACCESS_CONTROL";
import schema namespace ns1="urn:retailer" at "ld:RetailApplication/OrderManagement/schemas/Orde
import schema namespace ns2= "urn:retailerType"  at "ld:Logical/Order/schemas/Order.xsd";


   (::pragma function <f:function xmlns:f="urn:annotations.ld.bea.com" visibility="protected" ki
   </f:function>::)

   declare function f1:is_access_allowed($x1 as xsd:string?, $x2 as xsd:string*, $x3 as xsd:stri

   (::pragma function <f:function xmlns:f="urn:annotations.ld.bea.com" visibility="protected" ki
   </f:function>::)

   declare function f1:is_access_allowed_with_response_attributes($x1 as xsd:string?, $x2 as xsd


  declare function f1:secureOrders($order as element(ns2:ORDER)) as xs:boolean {
    let $result := f1:is_access_allowed_with_response_attributes("RTLApp/datacontrol/orderview",
    ( "totalorderamount"),
    (fn:string(fn:round($order/TotalOrderAmount)))) return
    if ($result[1] eq "true" )then
        let $class_index := fn:index-of($result, "class") return
            if (fn:empty($class_index)) then
                fn:false()
            else
                let $class_values := fn:subsequence($result, $class_index[1]) return
                if (fn:empty(fn:index-of($class_values, "ALESResponse"))) then
                    if (fn:empty(fn:index-of($class_values, "A"))) then
                        fn:false()
                    else
                        fn:true()
                else
                    let $separator_index := fn:index-of($class_values, "ALESResponse") return
                    let $new_class_values := fn:subsequence($class_values, 1, $separator_index[1
                    if (fn:empty(fn:index-of($new_class_values, "A"))) then
                        fn:false()
                    else
                        fn:true()
    else
       fn:false()
```

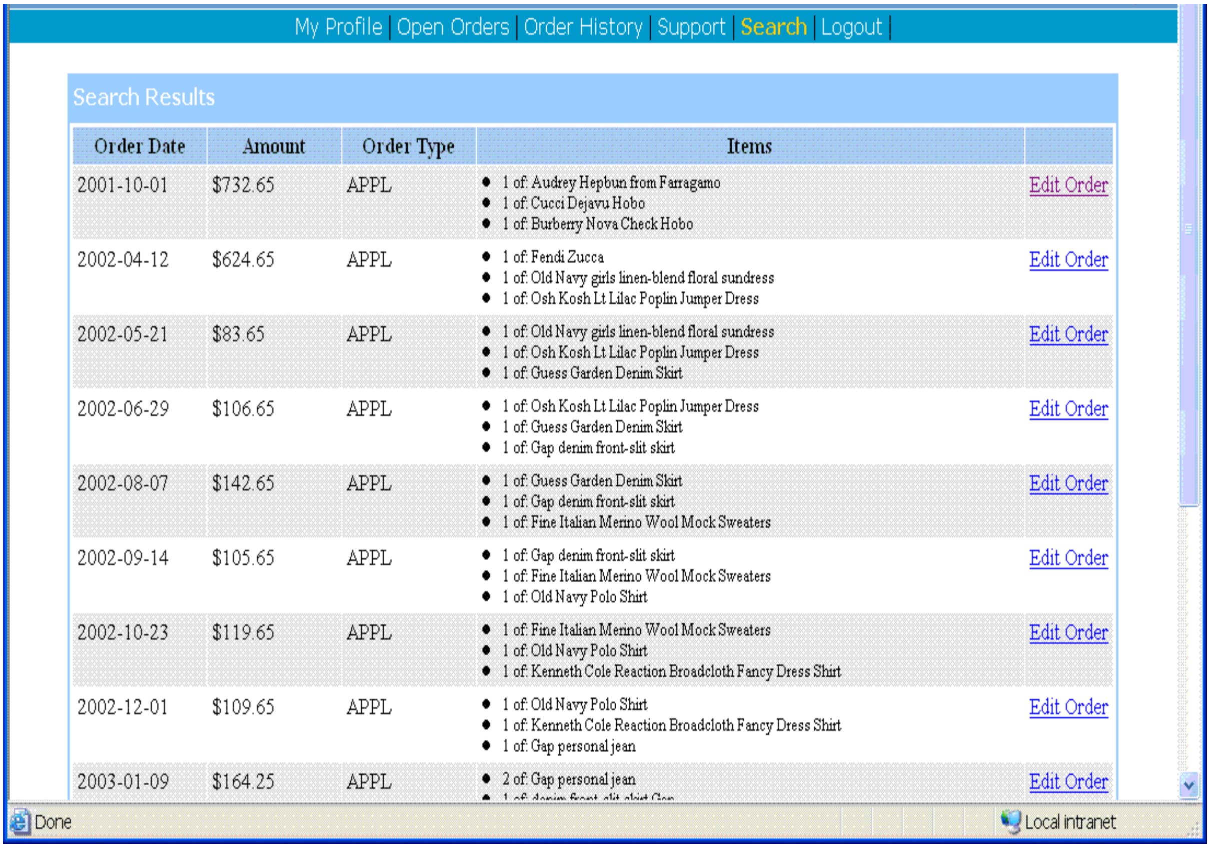
4. Redeploy the dataspace in ODSI Studio:

   a. Navigate to **RetailDataspace** and select the node.

   b. Navigate to **Project** at the menu bar and select **Clean.**
   c. Select **RetailDataspace** again. Then right-click the dataspace and select **Deploy Project**.

5. Specify which element of the data service is protected in the ODSI console:
   a. Login ODSI console and click **Lock & Edit**.
   b. Select **Security Configuration** on bottom right and navigate to **ALDSP Domain>RetailDataspace>RetailApplication>OrderManagement>OrderService**.
   c. On the **Secured Elements** tab, select the ORDER/ORDER checkbox and click **Save**.
   d. Navigate to **ALDSP Domain** > **RetailDataspace** > **RetailApplication**> **OrderManagement**>**OrderService**>**ORDER/ORDER** from data space navigation tree and click **ORDER/ORDER**.
   e. On the **Secured Elements Configuration** tab, enter `ld:RetailApplication/ALES_ACCESS_CONTROL` for the namespace. Then enter `secureOrders` for local name and click **Add**.
   f. Click **Activate Changes**.
6. Open RTLSelfService application (`http://localhost:7001/RTLSelfService`) and select the user **Steve** to log on.
7. Open the Search tab page and click the **Search Orders** button. Only items less than $1000.00 are displayed as shown in Figure 5-13.

Figure 5-13 Search Results with OES Protection


My Profile | Open Orders | Order History | Support | Search | Logout |
Done
Local intranet

Search Results

Order Date	Amount	Order Type	Items	
2001-10-01	$732.65	APPL	• 1 of: Audrey Hepbun from Farragamo • 1 of: Cucci Dejavu Hobo • 1 of: Burberry Nova Check Hobo	Edit Order
2002-04-12	$624.65	APPL	• 1 of: Fendi Zucca • 1 of: Old Navy girls linen-blend floral sundress • 1 of: Osh Kosh Lt Lilac Poplin Jumper Dress	Edit Order
2002-05-21	$83.65	APPL	• 1 of: Old Navy girls linen-blend floral sundress • 1 of: Osh Kosh Lt Lilac Poplin Jumper Dress • 1 of: Guess Garden Denim Skirt	Edit Order
2002-06-29	$106.65	APPL	• 1 of: Osh Kosh Lt Lilac Poplin Jumper Dress • 1 of: Guess Garden Denim Skirt • 1 of: Gap denim front-slit skirt	Edit Order
2002-08-07	$142.65	APPL	• 1 of: Guess Garden Denim Skirt • 1 of: Gap denim front-slit skirt • 1 of: Fine Italian Merino Wool Mock Sweaters	Edit Order
2002-09-14	$105.65	APPL	• 1 of: Gap denim front-slit skirt • 1 of: Fine Italian Merino Wool Mock Sweaters • 1 of: Old Navy Polo Shirt	Edit Order
2002-10-23	$119.65	APPL	• 1 of: Fine Italian Merino Wool Mock Sweaters • 1 of: Old Navy Polo Shirt • 1 of: Kenneth Cole Reaction Broadcloth Fancy Dress Shirt	Edit Order
2002-12-01	$109.65	APPL	• 1 of: Old Navy Polo Shirt • 1 of: Kenneth Cole Reaction Broadcloth Fancy Dress Shirt • 1 of: Gap personal jean	Edit Order
2003-01-09	$164.25	APPL	• 2 of: Gap personal jean • 1 of: denim front-slit skirt Gap	Edit Order

CHAPTER 6

Securing WebLogic Portal Applications

This section covers the following topics:

Overview

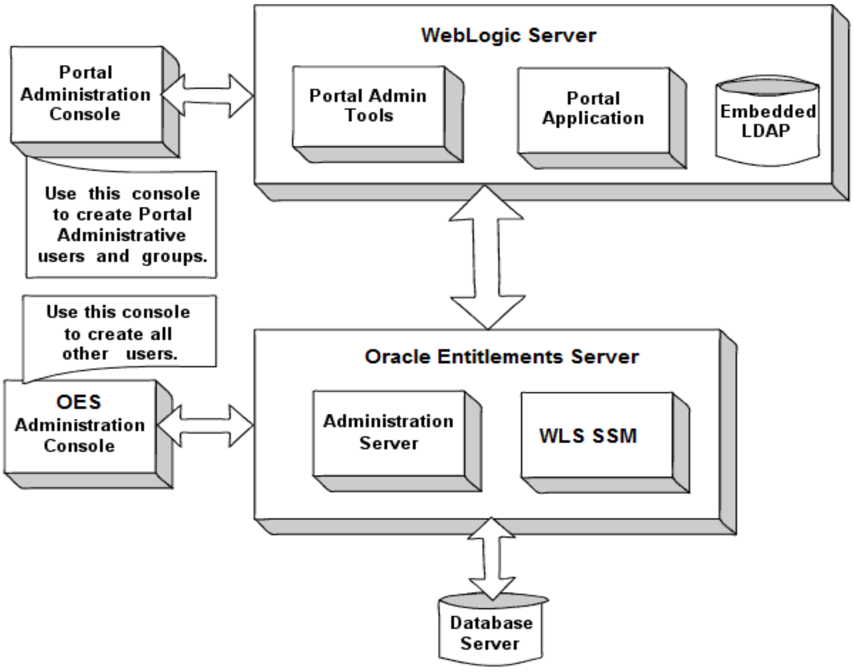
WebLogic Portal supports an open protection model and therefore requires a different policy model than the one used for other applications where access to a resource is denied unless a policy explicitly grants it. The following access model is enforced:

- If a Portal resource is not defined in OES, access to it is allowed for anyone.

- If a Portal resource is defined in OES, access to it is determined by the policies that apply to it. If no policies apply to it, access is denied.

OES does not replace all of the management functionality provided by the Portal Administration Tools. For example, as shown in Figure 6-1, WebLogic Portal administrative tools are used to manage administrative users and resources associated with Portal Delegated Administration and Portal Content Management.

Figure 6-1 Portal Integration Overview


WebLogic Server
Portal Admin Tools
Portal Application
Embedded LDAP
Portal Administration Console
Use this console to create Portal Administrative users and groups.
Use this console to create all other users.
OES Administration Console
Oracle Entitlements Server
Administration Server
WLS SSM
Database Server

Use-Case Scenario

The following use-case scenario is supported:

- The Administration Server manages the policies that secure Portal visitor entitlements and J2EE resources associated with portal applications and administration tools.
- Portal Administration Tools are responsible for the rest of portal management and administration, including the creation and management of administrative users and groups.

Constraints and Limitations

Integration with WebLogic Portal has the following constraints and limitations:

- Portal application administrators do not use WebLogic Portal Administration Tools to create and manage visitor entitlements on portal desktops, books, pages, and portlets.
- Application deployment descriptors are not used to deploy policy.
- Migration of existing portal application policy is not supported.

  OES does not to support the migration of visitor entitlements policy for existing portal applications. There are no facilities for migrating any information from the WebLogic Server embedded LDAP store.
- OES does not replace or in any way interfere with the use of the Portal Administration Tools for the management of resource structures associated with Portal Delegated Administration and Portal Content Management.

Prerequisites

This chapter assumes the following:

- Installation of WebLogic Platform/Portal 9.2 or 8.1, with Service Pack 4 or 5 and access to the WebLogic administration console to set the security configuration.
- Creation of a domain and a portal desktop.
- Installation and configuration of the WLS SSM or WLS 8.1 SSM and creation of the SSM instance on the portal machine.
- Access to the Administration Console on the Administration Server securing the WebLogic Portal.

Integration Tasks

The major tasks performed are:

1. Define the security providers as described in "Define the Security Providers" on page 6-4.
2. Define the Portal identities in OES as described in "Define Portal Identities in OES" on page 6-5.

3. Define the Portal resources in OES as described in "Define Portal Resources in OES" on page 6-5.
4. Define the policies that secure Portal resources as described in "Define Policies" on page 6-9.
5. Distribute the policies to the SSM.

Define the Security Providers

Note: Providers for WebLogic 9.x/10.0 are defined using the WebLogic console. For details, "WebLogic 9.x/10.0 Security Providers" on page 3-4.

The following providers are recommended for use in securing Portal resources:

- ASI Authorization
- ASI Role Mapping
- XACML Provider
- ASI Adjudicator

The following table provides information about each provider. For set-by-step instructions, see *Configuring Security Providers* in the Administration Console's help system.

Table 6-1 Security Providers Used with Web Server SSMs

Provider	Setting
ASI Authorization	(Required) See *Configuring an ASI Authorization Provider* in the console's help system for information on configuring this provider. Any field that this provider and the ASI Role Mapping provider have in common must be set to the same value.
XACML	This is the default WebLogic authorization provider.
ASI Role Mapping	(Required) See *Configuring an ASI Role Mapping Provider* in the console's help system for information on configuring this provider. Any field that this provider and the ASI Role Mapping provider have in common must be set to the same value.
Log4 Auditor (Optional)	See *Configuring a Log4j Audit Channel Provider* in the console's help system for information on configuring this provider.

Table 6-1 Security Providers Used with Web Server SSMs

Provider	Setting
ASI Adjudicator	See *Configuring an ASI Adjudication Provider* in the console's help system for information on configuring this provider. Make sure the **Require Unanimous Permit** checkbox is *not* selected.
WebLogic Authenticator	Perform the following steps: 1. In the Portal Administration console, go to **Service Administration**. 2. Select **Authentication Hierarchy Service**. 3. Add **WebLogicAuthenticator** to the A**uthentication Providers to Build** list.

Define Portal Identities in OES

Note: To implement the use-case scenario described in "Use-Case Scenario" on page 6-2, you must use identities described in this section.

To create the Identity directory and users using the OES administration console:

1. In the left pane, select **Identity** and click **New**.
2. On the **Create Directory** dialog box, enter `myusers` as the name and click **OK**. The `myusers` directory appears in the list of Identity directories.
3. In the left pane, select **Users** and click **New** at the bottom of the right pane.
4. Create the users that will visit your portal application.

   If you are using the WLS SSM (for WebLogic 9.x/10.0), create a user with the `Admin` role to access the WebLogic Server Administration Console. The default WebLogic Server Admin user and password is `weblogic/weblogic`.

Define Portal Resources in OES

When defining Portal resources in OES, keep in mind that OES does not deny access to a portal resource if it is not defined in OES *and* is accessible by anonymous access. In these cases, OES returns *abstain*, instead of *true* or *false*. This is an important difference in the way that OES makes decisions about other application resources where access is denied unless it is explicitly granted.

This means that you do not have to define any resources in OES that you want to leave unprotected.

The instructions below describe how to define the resources belonging to the sample portal application to be secured by OES.

Note: To implement the use-case scenario described in "Use-Case Scenario" on page 6-2, you must use the resources described in this section.

Realm Resource

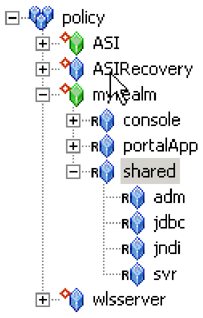
To create a realm resource, perform the following steps:

1. Select **Resources** in the left pane.
2. In the right pane, right-click `policy` at the top of the tree and select **Add Resource**.
3. On the **Create Resource** dialog box enter `myrealm` as the realm name and select `binding` from the **Type** dropdown list. Then click **Ok**. The realm resource appears under the **Policy** node.
4. Right-click the `myrealm` resource and select **Configure Resource**. Then select both the **Distribution Point** and **Allow Virtual Resources** checkboxes and click **Ok**.
5. Return to the left pane, expand the SSM configuration containing the defined security providers, and select the ASIAuthorizer. Then open the **Bindings** tab in the right pane.
6. Select `//app/policy/myrealm` from the dropdown list and click **Bind**.

Shared Resources

Figure 6-2 shows the shared resources required for securing the sample application.

Figure 6-2 Shared Resources



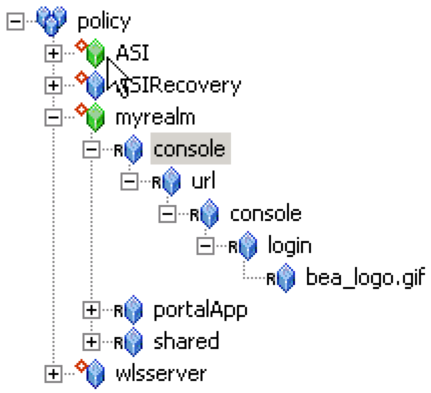
To create the `shared` resources, perform the following steps:

1. In the right pane, right-click the `myrealm` resource and select **Add Resource**.
2. On the **Create Resource** dialog box, enter `shared` in the **Name** field and click **Ok**. The `shared` resource appears under `myrealm`.
3. Right-click the `shared` resource and select **Configure Resource**. Then select **Allow Virtual Resources** and click **Ok**.
4. Right-click the `shared` resource and click **Add Resource**.
5. On the **Create Resource** dialog box, enter `adm` in the **Name** field and click **Ok**. The `adm` resource appears under the `shared` resource.
6. Repeat steps 4 and 5 to create the `jdbc`, `jndi`, and `svr` resources shown in Figure 6-2.

Console Resources

Figure 6-3 shows the required console resources.

Figure 6-3 Console Resources



To create the `console` resources, perform the following steps:

1. In the right pane, right-click the `myrealm` resource and select **Add Resource**.
2. On the **Create Resource** dialog box:
   - For WebLogic Portal 9.2 or 10.0, enter `consoleapp` in the **Name** field and click **Ok**.
   - For WebLogic Portal 8.1, enter `console` in the **Name** field and click **Ok**.
3. To create the `url`, `console`, `login`, `and bea_logo.gif` resources as shown in Figure 6-3, repeat steps 1 and 2 for each resource.
4. Right-click the `console` or `consoleapp` resource directly under `myrealm` and select **Configure Resource**. Then select the **Allow Virtual Resources** checkbox and click **Ok**.

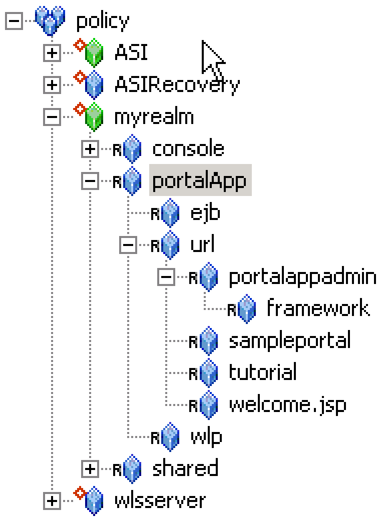
PortalApp Resources

Figure 6-4 shows the required resources. In addition, for WebLogic Portal 9.2, you must create these virtual resources under `myrealm`:

```
bea_wls_internal
wl_management_internal1
wl_management_internal2
pf-proliferation-jms
```

Figure 6-4 PortalApp Resources

To create the portalApp resources, perform the following steps:



1. Right-click the `myrealm` resource and select **Add Resource**.
2. On the **Create Resource** dialog box, enter `portalApp` in the **Name** field and click **Ok**.
3. Right-click the newly created `portalApp` resource and click **Add Resource**.
4. On the **Create Resource** dialog box, enter `ejb` in the **Name** field and click **Ok**. The `ejb` resource appears under the `portalApp` resource.
5. Right-click the `ejb` resource and select **Configure Resource**.
6. On the **Configure Resource** dialog box, select **Allow Virtual Resources** and click **Ok**.
7. Define the remaining resources shown in Figure 6-4.

Define Policies

Securing the sample application requires the authorization and role mapping policies described in this section.

Note: To implement the use-case scenario described in "Use-Case Scenario" on page 6-2, you must use the policies described in this section.

Authorization Policies

Table 6-2 describes the admin policy for the `shared/svr` resource and its children. To create this policy:

1. Expand the **Policy** node in the left pane and click **Authorization Policies** under it. Then click **New** near the bottom of the right pane.
2. On the **Create Authorization Policy** dialog box, select the **Grant** radio button and complete each tab as follows:

Table 6-2 Authorization Policy for Shared/Svr

Tab	Description
Privileges	Select `any` privilege in the **Select Privileges** list and click **Add.**
Resources	Select `shared/svr` in the **Child Resources** list and click **Add**.
Policy Subjects	Select `Admin` role from the **Roles List** and click **Add**.

3. Click **OK** on the **Create Authorization Policy** dialog box.

   Note: To assign multiple resources to a single privilege and role, specify all of the resources in one authorization policy.

4. Repeat these steps to define the remaining authorization policies. These are shown in Table 6-3 below.

Table 6-3 Portal Application Authorization Policies

Authorization Policies	Description

`grant(any, //app/policy/myrealm/shared/svr, //role/Admin) if true;`
`grant(any, //app/policy/myrealm/shared/adm, //role/Admin) if true;`
	Authorization policies for booting the WebLogic Portal server and performing administrative tasks.

`grant(any, //app/policy/myrealm/portalApp/url/sampleportal, //role/Everyone) if true;`
`grant(any, //app/policy/myrealm/portalApp/url/tutorial, //role/Everyone) if true;`
`grant(any, //app/policy/myrealm/portalApp/url/welcome.jsp, //role/Everyone) if true;`
	Grants permission to those in the role Everyone (includes the anonymous user) to access all of the tutorial and sample portal url resources. This authorization policy creates Portal open by default orientation for these two sample portals.

`grant(GET, //app/policy/myrealm/portalApp/url/portalappadmin/framework, //role/Everyone) if true;`	Allows unauthenticated users to access images used on the Administration Portal login page.
`grant(any, //app/policy/myrealm/portalApp/url/portalappadmin, //role/ PortalSystemAdministrator) if true;`	Grants permission for those is the role PortalSystemAdministrator to access the WebLogic Portal Administration url Portal resources
`grant(lookup, //app/policy/myrealm/shared/jndi, //role/Everyone) if true;`	Grants permission for those is the Everyone role to lookup JNDI resources.
`grant(reserve, //app/policy/myrealm/shared/jdbc, //role/Everyone) if true;`	Grants permission for those is the Everyone role to reserve JDBC resources.
`grant(any, //app/policy/myrealm/console, //role/Admin) if true;`	Grants permission for those in the Admin role to access the url resources of the WebLogic Server console.

Table 6-3 Portal Application Authorization Policies (Continued)

Authorization Policies	Description
`grant(GET, //app/policy/myrealm/console/url/console/login/bea_logo.gif, //role/Everyone) if true;`	Grants permission for those in the Everyone role to get access to the `bea_logo.gif` image resource in the WebLogic Server console
`grant(any, //app/policy/myrealm/portalApp/ejb, //role/Everyone)`	Initially allows access to all EJB methods.

Role Mapping Policies

Caution: If this policy is not created exactly as described, the authorization policies defined in Table 6-3 and Table 6-4 that use the `Everyone` role will not work properly.

Follow these steps to define a role mapping policy that allows the `Everyone` role to be used in the `myrealm` Identity directory.

1. Select **Role Mapping Policies** under the **Policies** node and click **New** at the bottom of the right pane.
2. Select the **Grant** radio button and complete the tabs as follows:

Tab	Description
Roles	Select `Everyone` and click **Add.**
Resources	Select `myrealm` and click **Add**.
Policy Subjects	Select `allusers` role and click **Add**.

The policy definition is as follows:

```
\grant(//role/Everyone, //app/policy/myrealm, //sgrp/myusers/allusers/) if
true;
```

Policies for Visitor Entitlements

WebLogic Portal uses visitor entitlements to determine who may access portal application resources and what they may do with those resources. OES provides a means of defining role-based policy for portal resources. The resources that can be entitled within a portal application include:

- desktops
- books
- pages
- portlets
- look and feels

Table 6-4 shows the capabilities of each of these resources:

Table 6-4 Capabilities According to Resource Type

Resource Type	View	Minimize	Maximize	Edit	Remove
Desktop	x				
Book	x	x	x		
Page	x				
Portlet	x	x	x	x	x
Look & Feel	x				

The capabilities listed in Table 6-4 are defined as follows:

- View—Determines if the user can see the resource.
- Minimize/Maximize—Determines if the user may minimize/maximize the portlet/book. This applies to books within a page, not to the primary book.
- Edit—Determines if the user can edit the resource properties.
- Remove—Determines if the user can remove the portlet from a page.

Policies for Desktops

Because there can be one or more desktops per portal, the portal is effectively a container for the desktops. A desktop is referenced as a resource in OES in the following manner:

```
//app/policy/myrealm/portalapp/wlp/sampleportal/com_bea_p13n/Desktop/<samp
lePortal>
```

where <*samplePortal*> is the label definition of the desktop.

Authorization policies specifying the `samplePortal` resource will control access at the `samplePortal` desktop level. Table 6-5 shows an authorization policy that allows visitors in the SampleVisitor role to view the `samplePortal` desktop.

Table 6-5 SamplePortal Authorization Policy

Effect	Privilege	Resource	Policy Subject
Grant	view	`/myrealm/portalapp/wlp/sampleportal/com_bea_p13n /Desktop/samplePortal`	SampleVisitor role

Policies for Books

A book is referenced as a resource in OES in the following manner:

```
//app/policy/myrealm/portalapp/wlp/sampleportal/com_bea_p13n/Book/<book_1>
```

where <*book_1*> is the label definition of the book.

Authorization policies specifying the `book_1` resource will control access at the `book_1` book level. Table 6-6 shows an authorization policy that allows the SampleVisitor role to view to `book_1`.

Table 6-6 Book_1 Authorization Policy

Effect	Privilege	Resource	Policy Subject
Grant	view	`/myrealm/portalapp/wlp/sampleportal/com_bea_p13n /Book/book_1`	SampleVisitor role

Policies for Pages

A page is the primary holder of individual portal elements such as portlets. A page is referenced as a resource in OES in the following manner:

`//app/policy/myrealm/portalapp/wlp/sampleportal/com_bea_p13n/Page/<page>`

where <*page*> is the label definition of the page.

Authorization policies specifying the `page_2` resource will control access at the page_2 page level. Table 6-7 shows an authorization policy that allows visitors in the SampleVisitor role to view `page_2`.

Table 6-7 Page_2 Authorization Policy

Effect	Privilege	Resource	Policy Subject
Grant	view	`/myrealm/portalapp/wlp/sampleportal/com_bea_p13n/Page/page_2`	SampleVisitor role

Policies for Portlets

Portlets are the visible components that act as the interface to applications and content. A portlet is referenced as a resource in OES in the following manner:

`//app/policy/myrealm/portalapp/wlp/sampleportal/com_bea_p13n/Portlet/<portlet_login>`

where <*portlet_login*> is the label definition of the portlet. (The WebLogic Portal name is `portlet_login_1`. This name is mapped to `portlet_login` by the WebLogic Portal Resource Converter.)

Authorization policies specifying the *`portlet_login`* resource will control access at the `portlet_login_1` portlet level. Table 6-8 shows an authorization policy that allows visitors in the SampleVisitor role to view the `portlet_login_1` portlet.

Table 6-8 Portlet_login_1 Authorization Policy

Effect	Privilege	Resource	Policy Subject
Grant	view	`/myrealm/portalapp/wlp/sampleportal/com_bea_p13n/Portlet/portlet_login`	SampleVisitor role

View Access to com_bea_p13n

Unless a policy grants the View privilege to the `com_bea_p13n` resource, authorization errors will occur when a portlet is accessed. Due to inheritance, assigning this privilege to `com_bea_p13n` also assigns it to its child resources. If this is not appropriate for the application, define a policy that restricts this inheritance. For example, the following policy prevents the inheritance of the View privilege to child resources of `com_bea_p13n`:

```
GRANT(
  //priv/view,
  //app/policy/someRoot/portalear/wlp/someportal/com_bea_p13n,
  //sgrp/Everyone/
) IF sys_obj_q =
//app/policy/someRoot/portalear/wlp/someportal/com_bea_p13n;
```

Policies for Look and Feel

A Look and Feel is a selectable combination of skins and skeletons that determine the physical appearance of a portal desktop. It is referenced as a resource in OES in the following manner:

```
//app/policy/myrealm/portalapp/wlp/sampleportal/com_bea_p13n/LookAndFeel/<
label>
```

where `<label>` is the label definition of the Look and Feel.

If you define an authorization policy at the *`textLookAndFeel`* level, you can control access at the *`textLookAndFeel`* level. Table 6-9 shows an authorization policy that allows the `SampleVisitor` role to view *`textLookAndFeel`* resource.

Table 6-9 textLookAndFeel Look and Feel Authorization Policy

Effect	Privilege	Resource	Subject
Grant	view	`/myrealm/portalapp/wlp/sampleportal/com_bea_p13n /LookAndFeel/textLookAndFeel`	SampleVisitor role

Policies for Portlets using Instance ID

Portlets have a unique instance ID that allows for granular authorization policy definition outside the standard hierarchy of the Desktop>Book>Page>Portlet. To use this in OES, add a condition statement in the portlet rule that adds the portlet instance ID. For example:

```
grant( [//priv/maximized,//priv/minimized,//priv/view],
//app/policy/myrealm/portalapp/wlp/sampleportal/com_bea_p13n/Portlet
/portlet_login, //role/Operator) if instanceid = "portlet_login";
```

Table 6-10 shows an authorization policy that allows visitors in the Operator role to view the *portlet_login_1* portlet.

Table 6-10 Portlet_login_1 Authorization Policy Using Instance ID

Effect	Privilege	Resource	Subject	Condition
Grant	view	`/myrealm/portalapp/wlp/sampleportal /com_bea_p13n/Portlet/portlet_login`	Operator role	`if instanceid = "portlet_login ";`

Storing and Versioning Policy with Oracle Enterprise Repository

This section describes how to integrate Oracle Entitlements Server with Oracle Enterprise Repository. It includes the following topics:

Overview

You can use Oracle Enterprise Repository to manage OES policy data as OER assets. By integrating OER with OES, you can:

- Share policy information between implementers and designers.
- Use the OER workflow to manage approval of changes made to OES assets.
- Maintain versioning of OES policy. An OES asset can have its version number updated in OER when a change occurs in the policy definition contained within the asset.
- Use OER's advanced categorization, reports and querying of OES assets.

Note: While the OER console allows direct modification of data in an OES policy asset, it is recommended that policy changes first be made in OES and then imported to OER. See "Importing/Exporting Policy Asset Data" on page 7-3.

Integration Tasks

To manage OES policies with OER:

1. Set OER import/export properties as described in "Set OER Import and Export Properties" on page 7-2.
2. Import the ALES Policy Asset Type into OER, as described in "Import Policy Asset Type into OER" on page 7-2.
3. Manage the ALES Policy Assets as described in "Manage Policy Assets" on page 7-3.

Set OER Import and Export Properties

Follow these steps to set the required import/export properties in OER:

1. In the OER console, open the **Admin** tab.
2. Select **System Settings** from the left column.
3. Under **Import / Export and Introspection** > **Import / Export**, set **Import/Export Client** `cmee.importexport.enabled` to **True**.
4. Under **Open API** > **Common**, set **Open API Enabled** `cmee.extframework.enabled` to **True**.
5. Click **Save**.

Import Policy Asset Type into OER

The OES Policy asset type must be imported in OER. This asset type defines OES metadata, such as privileges, policy, resources, and resource attributes. Follow these steps:

1. In the OER console, open the **Admin** tab.
2. Select **Import Export** > **Import/Export Client**.
3. In the **Import** tab's **Select file to import** field, navigate to `ales32-admin/data/aler` and select the appropriate Policy Asset Type zip file. Then click **Next** twice.

ALES 2.6—`ales_policy-asset-type.zip`
ALES 3.0—`ales_policy-asset-type-3.0.0.zip`
OES—`ales_policy-asset-type-3.2.0.zip`

4. Open the **Assets** tab and click **Edit/Manage Assets**.
5. In the Asset Editor, select **Actions** > **Manage Types** and verify that `ALES Policy Asset Type` appears in the Type Manager.

Manage Policy Assets

The Asset Editor displays OES Policy Assets in the following tabs:

- Overview—Asset name, location, and other general information.
- Resources—Name, type, description, attributes, and indicates if it is a virtual resource and/or distribution point.
- Authorization Policies—Used to manage authorization policies, actions and action groups.
- Role Mapping Policies—Used to manage role policies.
- Bindings—SCM name, SSM name, and binding resources of the asset.
- Entitlements Data—Information related to identities, identity schema, groups, group attributes, and group memberships.
- Administration—OER administration information about the asset.

Versioning OES Assets

OER maintains version information for its assets. OES Policy Assets use version numbers in the format N.N (1.0, for example). When importing Policy Assets into OER for the first time, the version number is set to 1.0. When you subsequently import the same assets, the version number increments by 1. You can also modify the version number of an asset within OER.

Importing/Exporting Policy Asset Data

The policyIX utility can perform imports/exports of policy asset data between OES and OER.

PolicyIX makes use of configuration files for imports/exports with OER. For details, see "Import/Export Configuration Files" on page 7-4.

For more information about policyIX itself, see PolicyIX in the *Administration Reference*

Export to OER

To export policy assets directly to OER, run policyIX with the `-exportToALER` option:

```
policyIX -exportToALER <config_file>
```

To export the data to OER using a policy data file:

```
policyIX -exportToALER <config_file> <file_name>
```

The structure of policy data files has changed in this release. If <`file_name`> was generated by an earlier version of policyIX, perform the following steps to add the necessary information to the file before exporting the data to OER:

1. Use ALES 2.6/3.0 version of policyIX to export data from ALES to a policy data file.
2. Use the current version of policyIX to import the policy data file into the OES database.
3. Import the OES Policy Asset Type into OER.
4. Use policyIX to export the OER data to a policy data file.

Importing to OES

To import directly into OES, use the `-importFromALER` option:

```
policyIX -importFromALER <config_file>
```

To import using a file:

1. Generate a file that obtains the data from OER:

   ```
   policyIX -importFromALER <config_file> <file_name>
   ```

2. Import the data from the file:

   ```
   policyIX -import <config_file> <file_name>
   ```

   Note: For imports. this version of policyIX supports earlier versions of policy data files.

Import/Export Configuration Files

This section describes the configuration file used for imports/exports between OER and OES. This file uses XML to specify required OER information.

Note: Further information about the configuration file used with policyIX, see Import/Export Utilities in the *Administration Reference* guide.

<aler_configuration>
Parent element containing all required <`aler_property`> elements.

<aler_property>

Specifies the name and value of an OER property using the format:

```
<aler_property name="<property_name>" value="<value>"/>
```

`server_version`—(Required) Server version (2.6 or 3.0)
`server_url`—(Required) connection URL
`username`—(Required) user name for connecting to OER
`userPassword`—(Required) user password
`assetName`—(Required) name of the asset
`assetDescription`—(Optional) Description of the asset
`importAssetVersion`—(Required) Asset version to import; valid only if the `-importFromALER` option is used in the policyIX command.

Listing 7-1 provides an example of the OER-related elements in the configuration file:

Listing 7-1 Import/Export Configuration

```
<aler_configuration>
    <aler_property name="server_version" value="3.0"/>
    <aler_property name="server_url"
        value=http://123.43.32.3546:7101/aler/services/FlashlineRegistry/>
    <aler_property name="userName" value="admin"/>
    <aler_property name="userPassword" value="tan66kds9"/>
    <aler_property name="assetName" value="MyALESPolicy"/>
    <aler_property name="assetDescription" value="An ALES Policy asset"/>
    <aler_property name="importAssetVersion" value="2"/>
</aler_configuration>
```

CHAPTER 8

Securing Oracle Service Bus Runtime Resources

This section covers the following topics:

Overview

The WLS SSM can be used to manage access control to Oracle Service Bus runtime resources. Oracle Service Bus is a configuration-based, policy-driven Enterprise Service Bus. It allows a loosely coupled architecture, facilitates enterprise-wide reuse of services, and centralizes management.

Only the runtime service bus resources are secured, meaning those resources that are passed to `isAccessAllowed()`. It does not secure the resources used during Oracle Service Bus configuration, such as the OSB console.

Prerequisites

This document assumes the following:

- Installation of WebLogic Server 10.0 MP1 and Oracle Service Bus 3.0
- An Oracle Service Bus-enabled domain (this domain supports WebLogic Server and Oracle Service Bus products)
- Installation of the Administration Server.
- Installation and configuration of the WLS SSM on the Oracle Service Bus machine.

Initial Configuration

Perform the following tasks to provide an SSM configuration and define an initial policy set for securing Oracle Service Bus resources. At the conclusion of these steps, you can refine this information as described in the remaining sections of this document.

1. Stop any running servers.
2. Start the Administration Server.
3. Configure the WLS SSM to protect an Oracle Service Bus domain as follows:
   a. Open a command window in `BEA_HOME/ales32-ssm/wls-ssm/adm`.
   b. Make a copy of `myssm_config.properties` and name it `alsb_ssm_config.properties`.
   c. Open `alsb_ssm_config.properties` in an editor. Set `ssm.type=wls-alsb-ssm` and specify other entries as needed.
   d. Execute `ConfigTool -process alsb_ssm_config.properties`.
4. Start Oracle Service Bus domain server and access its console (typically, `http://host:port/sbconsole`).

   You can now use the facilities, including creating/managing projects.

Sample Properties File

This is a sample of the properties files used to establish the initial configuration for securing Oracle Service Bus resources.

For instructions on completing this file, see the SSM Installation and Configuration Guide.

```
### This file lists properties for the SSM configuration tool
### ConfigTool will interactively prompt for values which
### are commented out

### This is the weblogic domain directory
### Use / (and not \ ) for the path
wls.domain.dir =
C:/BEAProducts/alsb300_wls100/user_projects/domains/alsb_domain_3

### SSM's config-id
### You can use the name of your application for this value
ssm.conf.id = SimpleApp2

### Database password
db.password = password

### OES Admin password
ales.admin.password = password

### SSM Username and password
### Note : This is the admin user's username/password of the domain being
### protected. In this case the target domain is the ALSB domain
ssm.admin.name = system
ssm.admin.password = weblogic

### The type of SSM defined by the type of domain against which it
### is configured. The tool will load policies and configuration from
### BEAHOME/ales*-ssm/wls-ssm/config/<ssm.type> where <ssm.type> is one of:
###       wls-ssm (for WebLogic Server domain)
###       wls-portal-ssm (for WebLogic Portal domain)
###       wls-alsb-ssm   (for AquaLogic Service Bus domain)
### Note : For ALSB domain this is'wls-alsb-ssm'.
```

```
ssm.type = wls-alsb-ssm

#############################################################
### If you have not installed Admin and SSM in the same BEA-HOME,
### specify the values below. The ConfigTool will interactively prompt for
### values that are commented out
#############################################################

### Database user name
# db.login = db_user

### OES Admin username
# ales.admin.name = admin

### name of the SSM instance directory
# ssm.instance.name = MySsm

### the OES application node name
### This is like the root resource for the SSM
# ales.resource.root = //app/policy/MyApp

### OES identity directory name
# ales.identity.dir = ALSBdir

### Database JDBC URL:
### Oracle -> jdbc:oracle:thin:@<server>:<port>:<sid>
### Sybase -> jdbc:sybase:Tds:<server>:<port>
### Sql Server -> jdbc:sqlserver://<server>:<port>
### Pointbase -> jdbc:pointbase:server://<server>/ales
###
### values:
### <server>: name or IP address of database machine
### <port>: port where the database listener is running
### <sid>: SID for oracle database
# db.jdbc.url = jdbc:oracle:thin:@db_server:1521:db_sid

### Database JDBC Driver:
### Oracle: oracle.jdbc.driver.OracleDriver
### Sybase: com.sybase.jdbc3.jdbc.SybDriver
### Sql: com.microsoft.sqlserver.jdbc.SQLServerDriver
### Pointbase: com.pointbase.jdbc.jdbcUniversalDriver
### DB2: com.ibm.db2.jcc.DB2Driver
# db.jdbc.driver = oracle.jdbc.driver.OracleDriver

### ARME's port number, by default this is 8000
# arme.port = 8000
```

Security Providers

Note: Providers for WebLogic 9.x/10.0 are defined using the WebLogic console. For details, see "WebLogic 9.x/10.0 Security Providers" on page 3-4.

To secure Service Bus resources, create a security realm and define the following provider types:

- ASI Authorizer
- ASI Role Mapping
- XACML Authorizer
- OES Adjudicator

Notes: User could specify resource types that only use the decision from ASI Authorizer or XACML Authorizer. Multiple resource types are separated by a comma. By default, a resource type of 'wlsb-console' only uses the decision from XACML Authorizer, and a resource type of 'alsb-proxy-service' only uses the decision from ASI Authorizer.

When creating the realm, use the following settings:

- **Security Model Default**—`Advanced`
- **Combined Role Mapping Enabled**—clear this checkbox
- **Check Role and Policies**—`All Web applications and EJBs`

OSB Resources

Policy definitions include the Oracle Service Bus resources to which the policy applies. These resources must be defined in Oracle Entitlements Server.

Regular Resource

To create a regular resource named `abc`:

1. In the Administration Console, open the resource tree.
2. Right-click the parent of `abc` and select **Add Resource**.
3. In the **Name** field, enter `abc` and click **OK.**

Virtual Resource

To create a virtual resource named `xyz`:

1. Create a resource as described in "Regular Resource" on page 8-5.
2. Right-click the `xyz` resource and select **Configure Resource**.
3. Check the **Allow Virtual Resources** box and click **OK**.

OSB Proxy Service Resources

Create resources in OES corresponding to the OSB Proxy Services. An OSB Proxy Service has up to four key/value properties:

- **path**—Full name of the proxy service, for example: `path=project/folder1/folder2`
- **proxy**—Name of the proxy service, for example `proxy=myProxy`
- **action**—One of two values, `invoke` or `wss-invoke`
- **operation**—The name of the operation to invoke, used only where `action=wss-invoke`, for example `operation=processPO`

Resource definitions for Oracle Service Bus use this format:

```
//app/policy/<binding app>/<Proxy Service App name>/ProxyService/<Project
Name>/[Folder name]/<Proxy Service Name>
```

Table 8-1 describes how Oracle Service Bus Proxy Service reference elements map to OES resource and privilege elements

Table 8-1 OSB Proxy Service Elements Represented in OES Resources and Privileges

Resource/Privilege Element	Description
binding app	The OES binding node name.
Proxy Service app name	The default application name, `shared`.

Table 8-1 OSB Proxy Service Elements Represented in OES Resources and Privileges

Resource/Privilege Element	Description
ProxyService	The OES resource type.
Folder name	The OSB Proxy Service folder name.
`//priv/<operation>`	The operation field of the OSB Proxy Service, representing one of the Web Services operations provided.

Here is an example of how to convert an Oracle Service Bus transport level access control to a policy. In OSB:

```
type=type=<alsb-proxy-service>, path=project/folder, proxy=myProxy,
action=invoke
```

is converted to:

```
//app/policy/<binding app node>/shared/ProxyService/project/folder/myProxy
```

with a default privilege of `//priv/access`, since with `action=invoke`, there is no operation defined.

Here is an example of how to convert OSB access control during inbound web-service-security request processing:

```
type=<alsb-proxy-service>, path=project/folder, proxy=myProxy,
action=wss-invoke, operation=ProcessPO
```

is converted in OES to:

```
//app/policy/<binding app node>/shared/ProxyService/project/folder/myProxy
```

with a privilege of `//priv/ProcessPO`.

Resource Binding Application and Distribution Point

To make a resource binding application and distribution point named `def`:

1. Right click the mouse on parent of `def`, and select **Add Resource**.
2. In the **Name** field, enter `def`.
3. From the Type dropdown field, select **Binding and** check the **Distribution Point** box.

4. After the resource is created, right-click the resource and select **Configure Resource**.
5. Select **Binding** application from the pull-down menu and click **OK**.

Resource Tree

Select Resources on the left pane and create a resource tree as shown in Listing 8-1:

1. Make `myrealm` a resource binding application and distribution point.
2. Make the `consoleapp` and `ProxyService` resources virtual.

Listing 8-1 Resource Tree

```
myrealm
   |---- consoleapp
   |---- shared
         |----- adm
         |----- eis
         |----- ejb
         |----- jdbc
         |----- jms
         |----- jndi
         |----- ProxyService
         |       |----- MortgageBroker
         |               |----- ProxyService
         |                      |---- loanGateway1
         |                      |---- loanGateway2
         |                      |---- loanGateway3
         |----- svr
         |----- url
         |----- webservices
         |----- workcontext
```

Discovering Services

When developing policies, you can use the Discovery mode feature to help define your policy components. Instructions for using Discovery mode are provided in the Resource Discovery section in the *Policy Managers Guide*.

Service Bus Identities

The ConfigTool will create an OES Identity directory and the OSB administrative user. This user's password is used to start the OSB application. Assuming the OSB Identity directory name is `ALSBdir` and the administrative user name is `weblogic`, follow these steps to maintain the password:

1. In the Entitlements Administration Application, select the organization containing the OSB identity directory and select the **Identities** tab in the right pane.
2. In the **Identity Directories** list, select the `ALSBdir` directory.
3. On the **Users** tab, weblogic user and click **Modify** at the bottom of the pane.
4. Specify the password as required and click **OK**.

Additional users and groups may be required. For background information, see Identities in the *Policy Managers Guide*.

Policies for OSB

The ConfigTool will create an initial set of policies using the files located in `BEA_HOME/ales32-admin/examples/policy/alsb_sample_policy`. You may import and use them as a starting point for developing a full set of policies to secure OSB resources. For information about how to import the sample policies, see the README file in the sample directory and see also Importing Policy Data in the *Policy Managers Guide*.

This section includes examples of policy creation:

- "Authorization Policies" on page 8-9
- "Role Mapping Policies" on page 8-10

Authorization Policies

The following policy grants any user with the role `Admin` all privileges over the resources `adm` and `svr` resources:

```
grant(any, //app/policy/myrealm/shared/adm, //role/Admin)if true;
grant(any, //app/policy/myrealm/shared/svr, //role/Admin) if true;
```

To add this policy:

1. Select the application in the left pane. Then and click the **Policies** tab in the right pane and select **Authorization Policies**.
2. Click **New** at the bottom of the pane and complete the policy definition as follows:

Effect — grant
Actions — any
Resources — `adm`, `svr`
Subjects — `Admin`

3. Repeat these steps to create a policy that grants all users all privileges over the `eis`, `ejb`, `jdbc`, `jms`, `jndi`, `url`, `webservices` and `workcontext` resources:

```
grant(any, //app/policy/myrealm/shared/eis, //role/Everyone) if true;
grant(any, //app/policy/myrealm/shared/ejb, //role/Everyone) if true;
grant(any, //app/policy/myrealm/shared/jdbc, //role/Everyone) if true;
grant(any, //app/policy/myrealm/shared/jms, //role/Everyone) if true;
grant(any, //app/policy/myrealm/shared/jndi, //role/Everyone) if true;
grant(any, //app/policy/myrealm/shared/url, //role/Everyone) if true;
grant(any, //app/policy/myrealm/shared/webservices, //role/Everyone) if
true;
grant(any, //app/policy/myrealm/shared/workcontext, //role/Everyone) if
true;
```

4. Repeat these steps to create a policy that grants all users access to the `ProxyService` resource:

```
grant(access,
//app/policy/myrealm/shared/ProxyService/MortgageBroker/ProxyService,
//role/Everyone)if true;
```

Role Mapping Policies

The following policy grants the user `weblogic` the role `Admin` over the resource `myrealm`:

```
grant(//role/Admin, //app/policy/myrealm, //user/asi/weblogic/) if true;
```

To add this policy:

1. Select the application in the left pane. Then and click the **Policies** tab in the right pane and select **Role Policies**.

2. Click **New** at the bottom of the pane and complete the policy definition as follows:

   Effect — grant
   Roles — `Admin`
   Resources — `myrealm`
   Subjects — `weblogic` user

3. Repeat these steps to create a policy that grants the user `anonymous` the role `Anonymous` over the resource `myrealm`:

```
grant(//role/Anonymous, //app/policy/myrealm, //user/asi/anonymous/) if
true;
```

4. Repeat these steps to create a policy that grants the group of all users the role `Everyone` over the resource `myrealm`:

   ```
   grant(//role/Everyone, //app/policy/myrealm, //sgrp/asi/allusers/) if
   true;
   ```

Distributing Changes

After you have made changes to the configuration and policies in the Entitlements Administration Application, distribute the policies to the SSM by clicking the Distribute tab at the top of the console.

After the policies are distributed, start both the `myrealm` ARME instance used to protect the Oracle Service Bus domain and the domain itself.

Verifying SSM Configuration Using PerfDBAuditor

It is possible to use the performance auditing provider to verify that the SSM has been properly configured to protect Oracle Service Bus.

To use the PerfDBAuditor to verify the SSM configuration, follow these steps:

1. In the WebLogic Server Administration Console, select **Security Realms** > **myrealm** > **Providers** > **Auditing** and click **New**.
2. In the **Name** field, enter `PerfDBAuditor`. Then select `PerfDBAuditor` from the **Type** field and click **OK**.
3. On the **Configuration: Provider-Specific** page for the PerfDBAuditor security provider, enter the JDBC connection information. For Oracle databases, the JDBCDriver Class Name is `oracle.jdbc.driver.OracleDriver` and the JDBC Connection URL is `jdbc:oracle:thin:@oracle-host:1521:listener-name`, where `oracle-host` is the name or IP address of the system running the Oracle database and `listener-name` is the name of the database listener.

   Optionally, set the Performance Statistics Interval attribute to 1 to collect data at 1 minute intervals (instead of the default 5 minutes).
4. Click on **Save** and then activate changes.
5. Stop and restart the domain.

6. Generate some data by:

   a. Opening (`http://localhost:7021/examplesWebApp/index.jsp`) and reloading the application.

   b. Under **Run the Service Bus Examples**, click **Run the Example**.

   c. Click **Submit Loan Application**.

7. After a few minutes, check the PERF_ATZ_STAT database table. You should see a non-zero value under TOTALREQ. This indicates that the SSM is configured correctly to secure the application.

Integrating Oracle Access Manager as an Authentication Provider

This section contains the procedures to integrate Oracle Access Manager as an authentication provider with Oracle Entitlements Server.

- "Configuring the Administration Server" on page 9-1
- "Configuring the Security Service Module" on page 9-2

Configuring the Administration Server

The following procedure configures Oracle Entitlements Server to use Oracle Access Manager as an authentication provider.

1. Build the provider JAR using the WebLogic MBeanMaker.

   See Using the WebLogic MBeanMaker to Generate the MBean Type at http://download.oracle.com/docs/cd/E12890_01/ales/docs32/dvspisec/progrmng.html#wp1074928.

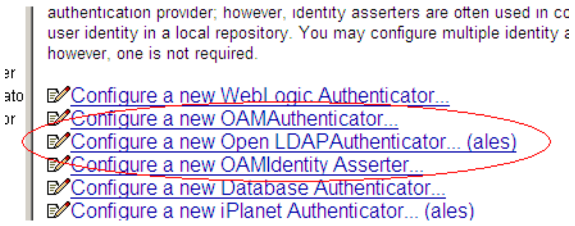
2. Copy the `oamAuthnProvider.jar` to `ales32-admin/lib/providers/ales`.

3. Restart the Administration Server.

4. Login to the ASI console at `https://`*`hostname`*`:`*`sslport`*`/asi/`.

5. Open the `Administration Console/Security Configuration/Service Control Managers/`*`SCM Name`*`/`*`App Name`*`/Authentication` folder located in the left frame.

6. Click **Configure a New OAMAuthenticator** or **Configure a new OAMIdentity Asserter** to create the respective server definitions.



authentication provider; however, identity asserters are often used in cc
user identity in a local repository. You may configure multiple identity a
however, one is not required.

Configure a new WebLogic Authenticator...
Configure a new OAMAuthenticator...
Configure a new Open LDAPAuthenticator... (ales)
Configure a new OAMIdentity Asserter...
Configure a new Database Authenticator...
Configure a new iPlanet Authenticator... (ales)

Configuring the Security Service Module

The following procedure configures an Oracle Entitlements Server Security Service Module to use Oracle Access Manager as an authentication provider.

1. Upgrade the provider using the following command .

```
ales32-ssm/xxx-ssm/instances/ssm-instance-name/bin/upgrade_providers.sh
 -srcDir dir-contains-provider-jar-to-be-processed -destDir dest-dir
```

2. Stop the instance of the Security Service Module.

3. Copy `oamAuthnProvider_Update.jar` to the `ales32-ssm/`*`xxx-ssm`*`/lib/providers/ales` directory.

4. Add the following Java property: “”.

```
-Djava.library.path=OAM-SDK-Home/oblix/lib
```

5. Add the JAR to the class path.

```
OAM-SDK-Home/oblix/lib/jobaccess.jar
```

6. Start the instance of the Security Service Module.

Securing Microsoft Office SharePoint Server (MOSS) Resources

The Oracle Entitlements Server integrates with Microsoft Office SharePoint Server (MOSS) to provide protection of pages hosted on the SharePoint portal. This integration solution provides a fine grained entitlements solution for SharePoint.

This document describes software requirements, installation and configuration procedures, and setup steps.

Overview

Integration with SharePoint is provided through plugins which intercept calls within the SharePoint server and send the same to a Web Service SSM that acts as the Policy Decision Point (PDP).

With this integration, protection of the following SharePoint components will be externalized using OES:

- Web Sites
- Web Pages within SharePoint Web Sites
- Web Parts
- List Items
- Custom Page Content

Software Requirements

The following software must be installed for this integration to work properly.

Table 10-1 Required Software and Version

Product	Version
Operating System	Microsoft Windows Server 2003 SP1/SP2
Microsoft .NET	Microsoft .NET Framework v3.5
Application Server	IIS 6 (For SharePoint 2007)
SharePoint Server	Microsoft Office SharePoint Server 2007*
Oracle Entitlements Server	OES 10gR3 or ALES 3.0 CP2 or Higher

Install the SharePoint SSM

The SharePoint SSM is included in the IIS SSM. To install it, launch the SSM installation program (OES10gR3_ssm_win32.exe) and select the IIS SSM when prompted.

The SSM is installed in `<BEA_HOME>/ales32-ssm/iis-ssm/sharepoint-ssm`. This directory contains the directories/files listed in Table 10-2.

Table 10-2 SharePoint SSM Directories and Files

Directory or File	Description
[.]	The base directory containing the `adm` and `lib` directories and the Word *readme* of this document.
[adm]	Directory containing files used for configuration and error page display.
[lib]	Directory containing shared libraries (.DLLs), executables (.EXE), Java packages (.JARs) which serve as functional modules. It also contains [`ALESAuthorizationFeature`] directory which contains XML files used for deployment on the SharePoint server.
[lib] BEA.SharePoint.dll	Assembly (DLL) containing the SharePoint integration classes. The components packaged within this assembly include: Authorization Web Control Custom tag library HTTP Module ALES Authorizer (with supporting classes)
[lib] Feature.xml	This XML file is the deployment descriptor of the feature used to deploy the authorization web control.
[lib] Elements.xml	This XML file contains various elements deployed within the feature which is essentially the control in this case.
[Runtime] Log4net.xml	This XML file contains the logging pattern, log file location, etc. for the output logs.
[lib] MOSSResourceDiscovery.exe	This executable is the administration script used to extract the resources from SharePoint and dump them into policy files that may be imported into OES.
[Discovery] AdmUrls.txt	This file contains the set of administrative URL’s which are replicated by default for each new web created in SharePoint.
[Pages] custError.aspx	The customize error page which is displayed when a user is not authorized to view a page.
[lib] ALES_MOSS_Installer.exe	This executable is the installation script used to install the solution.
Config.txt	This configuration file is used by the installation script.
[lib] SampleIdentityAsseter.jar	This is the Sample Identity asserter used to test the solution with the default Windows based authentication used at SharePoint.
[lib] ssmwsCustomAssertion.jar	This is the custom credential holder used at WS-SSM to test the solution.

Configure the SharePoint SSM

This section describes how to configure the SharePoint SSM. It assumes the deployer has administrative privileges on the Windows server where SharePoint is installed.

1. **Assembly Deployment**

   Open `C:\WINDOWS\assembly` in a separate Windows Explorer window. Then drag and drop the following files into this folder.

   ```
   <BEA_HOME>\ales32-ssm\iis-ssm\sharepoint-ssm\lib\BEA.SharePoint.dll
   <BEA_HOME>\ales32-ssm\iis-ssm\sharepoint-ssm\lib\log4net.dll
   ```

   This will register these assemblies in the windows GAC (Global Assembly Cache) and make them available to all .NET applications on the host.

2. **Update Master Page**

   The SharePoint default.master page template (which is used by the various sites to create master pages of their own) has to be updated with the declaration of the delegate control. The location of the default.master page template is `C:\Program Files\Common Files\Microsoft Shared\web server extensions\12\TEMPLATE\GLOBAL`. The declaration for specifying the delegate control which is specified in the HTML HEAD section of this master page is as follows:

   ```
   <SharePoint:DelegateControl runat="server" ControlId="PageHeader"/>
   ```

   In addition to this any custom master pages used for the SharePoint sites will have to be updated with the delegate control declaration. This can be easily done via the Microsoft Office SharePoint Designer.

3. **Custom Error Page**

   Copy `<BEA_HOME>\ales32-ssm\iis-ssm\sharepoint-ssm\adm\Pages\custError.aspx` to the following SharePoint server directory:

   ```
   C:\Program Files\Common Files\Microsoft Shared\web server
   extensions\12\TEMPLATE\LAYOUTS
   ```

4. **Restart IIS**

   For detailed instructions on restarting IIS, see: http://www.microsoft.com/technet/prodtechnol/WindowsServer2003/Library/IIS/003ed2fe-6339-4919-b577-6aa965994a9b.mspx?mfr=true

5. **SharePoint Resource List**

To identify the SharePoint resources to be defined in OES and included in policy definitions, run `MOSSResourceDiscovery.exe` that is provided in the SharePoint SSM. This will generate three files (object, objattr and decl) in a policy format that can then be used as input to the policy loader.

a. Open a command line window on the SharePoint Server and run `<BEA_HOME>\ales32-ssm\iis-ssm\sharepoint-ssm\lib\MOSSResourceDiscovery.exe`.

b. When prompted, enter the following:

- Path to an *existing* folder where the policy files will be created which the tool will create the policy files, for example `C:\moss-ales-policy`.
- Path to `<BEA_HOME>\ales32-ssm\iis-ssm\sharepoint-ssm\adm\Discovery\AdmUrls.txt`, which is used to extract the admin URLs.
- URL of the top level web (also known as home web) in SharePoint, e.g. `http://<sharepoint_server_name>`.
- Root resource under which the rest of the SharePoint resources will be defined in OES, e.g. `//app/policy/MOSS`.

Note that MOSSResourceDiscovery.exe takes a while to complete.The following is a sample successful execution of this process.

```
C:\bea\ales32-ssm\iis-ssm\sharepoint-ssm\lib>MOSSResourceDiscovery.exe
----------------------------------------------------------
         Welcome to the MOSS Resource Discovery
----------------------------------------------------------
Enter the folder path where you want to create object,objAttr and decl file
c:\moss-ales-policy
Enter Path where Admin Url file is located
C:\bea\ales32-ssm\iis-ssm\sharepoint-ssm\adm\Discovery\AdmUrls.txt
Enter SharePoint site URL and DONOT append url with /. e.g.
http://sharepoint01
http://eagle

Enter Resource Base and DONOT append resource base with /. e.g.
//app/policy/MicrosoftSharePoint
//app/policy/MOSS

Resource Discovery starts....
Resource Discovery completed.
```

Configure the Web Service SSM

The SharePoint SSM uses the Web Service SSM to make calls for policy authorization. Therefore, the Web Service SSM instance must be correctly configured as described in *Configuring SSMs Using the ConfigTool* in the *SSM Installation and Configuration Guide*.

After configuring the Web Service SSM, ensure that the ConfigTool created the root resource and bound it to the Web Service SSM. To verify this, log in to the Entitlements Administration Application and select the **DefaultApp** application in the left pane. Then click the **Resources** tab in the right pane. The root resource should appear directly under the **Resources** node. Select it and click **Modify** to make sure the **SSM Bound** field is set to the Web Services SSM.

If the application root resource is not present, it can be created it as follows:

1. Under **DefaultOrg**, select **DefaultApp** in the left pane and click the **Resources** tab in the right pane.
2. Select the **Resource** node and click **New** at the bottom of the pane.
3. On the **New Resource** dialog, complete the fields as described below and click **OK**:
   - **Name** — the root resource name, e.g., `MOSS`.
   - **Type** — select binding
   - **Allow Virtual Resource** — select the checkbox
   - **Distribution Point** — select the checkbox
   - **SSM Bound** — select the SSM created when the Web Service SSM was configured.
4. Click **Save Changes** at the top of the main window.

Import SharePoint Resources

After generating the SharePoint resource list, the resources must be imported into the OES.

To import the resources, do the following:

1. In the Web Service SSM's `config\webservice-ssm\ales-policies` directory, make a copy of `load.conf` file and save it in

`<BEA_HOME>\ales32-ssm\iis-ssm\sharepoint-ssm\adm`. Then modify the file as described in Setting Configuration Parameters in the *Policy Managers Guide*.

2. Enter the following:

```
Run policyloader.bat
<BEA_HOME>\ales32-ssm\iis-ssm\sharepoint-ssm\adm\load.conf
```

After the resources are imported, use the Entitlements Administration Application to define the the authorization policies to control access to these resources.

Configure the SharePoint Server

This section contains instructions for configuring the SharePoint server to connect to the Web Service SSM:

- Automated Configuration describes how to use the interactive installer utility.
- Manual Installations describes how to perform the steps manually.

Automated Configuration

The installation of SharePoint code can be accomplished as follows:

1. Execute the following script:

   ```
   BEA_HOME\ales32-ssm\iis-ssm\sharepoint-ssm\lib\ALES_MOSS_Installer.exe
   ```

2. Complete the prompts as described in Table 10-3.

Table 10-3 ALES_MOSS_Installer.exe

Prompt	Description
Web Service URL	The URL of the Web Service SSM. Example: `http://<hostname>:<port>/ServiceRegistry`
Web service SSM ID	The Web Service SSM's configuration ID.
Name of the token in the user's HTTP session used to assert the identity of the user.	The name of the token key used to assert the identity of the user. This depends on the type of identity assertion being used. **Note:** The default in Sharepoint is LOGON_USER . If SharePoint is configured to use a different authentication model, this token will be different.

Table 10-3 ALES_MOSS_Installer.exe

Prompt	Description
Token name used for identity assertion	The name of the identity assertion type of the assertion token used in the ALES Identity Asserter that is configured for the Web Service SSM. **Note:** If using `BEA_HOME\ales32-ssm\iis-ssm\sharepoint-ssm\lib\SampleIdentityAsserter.jar` as the identity asserter on the Web Service SSM.
Fully-qualified path to the web configuration file (web.config) of the MOSS web application	Fully-qualified path to `web.config`, for example `C:\Inetpub\wwwroot\wss\VirtualDirectories\80\web.config`
Please enter the WS - SSM resource base	Name of the root SharePoint resource. This resource was created as described in "Configure the Web Service SSM" on page 10-6.
Fully-qualified path to the log4Net XML configuration file (log4Net.xml)	Fully-qualified path to the log4net configuration file. A default log4Net.xml is present in the `<BEA_HOME>\ales32-ssm\iis-ssm\sharepoint-ssm\adm\Runtime` directory. It needs to be updated based on required log level, log file location and logging format. Give grant permission to "Everyone" to the folder in which the log file will be generated. This is because the logs are written from each SharePoint user's security context.
Please enter the Top level SharePoint site URL	Enter the URL in the format: `http://<sharepoint_server_name>/` **Note:** Be sure to include the '/' at the end of the URL.

3. When prompted, confirm you want to continue the installation. When you do so, the web configuration file is updated with the necessary information and the IIS server is restarted.

4. When prompted to activate the authorization feature on the SharePoint sites and sub sites, enter the URL of the site or sub site.

   Note: This can also be accomplished by selecting **Site Settings** > **Modify All Site Settings> Site Features** page for the web.

   The installer then echoes entries you made and prompts you to verify the entries. Upon verification, the following messages appear:

```
Please wait while the installation proceeds ......

Changes are being made in config file...
Config file has been updated
```

```
Feature directory has been deployed

Server is being restarted....
Attempting stop...
Internet services successfully stopped
Attempting start...
Internet services successfully restarted


Feature is being installed
Operation completed successfully.
Feature has been installed
```

5. When prompted, verify that you want to activate the authorization feature on individual sites.
6. When prompted, enter the SharePoint site, enter the address in the following format:

   `http://<sharepoint_server_name>:<port>/<site_name>`

7. When prompted to exit, enter `0`.

Manual Installations

This section describes how to manually configure the SharePoint server (without using ALES_MOSS_Installer.exe).

Modify SharePoint Web Configuration

The SharePoint web configuration file must be updated to include information about the assembly deployed above. Following is a list of changes that need to be performed:

1. Open the following file in an editor:

   `C:\Inetpub\wwwroot\wss\VirtualDirectories\80\web.config`

   WARNING: On Windows, use Notepad. Using Wordpad will add '?' characters to the file.

2. In the **appSettings** section, specify the properties as described in Table 10-4.

Table 10-4 AppSettings

Entry	Description
SsmUrl	Registry URL of Web Service SSM (e.g. http://192.168.120.38:9000/ServiceRegistry)
SsmId	Configuration ID of Web Service SSM

Table 10-4 AppSettings

Entry	Description
token	Name of the token used for getting the users assertion from the session
IdentityAsserterName	Name of the identity assertion type defined in identity asserter which is configured in Web Service SSM
resourceBase	Distribution point of SharePoint resource tree bound to Web Service SSM
log4NetXmlfile	Fully qualified path to the log4Net XML configuration file.
sharepointSite	Top level SharePoint site, e.g. http://spsvr2/ (Here the trailing '/' is important)
DisableALES	Flag to disable the ALES integration **Note:** When set to **true**, the Oracle SharePoint modules do load and no policies are evaluated. Therefore, no runtime authorization is performed against OES. For Web Parts, authorization is performed at runtime. If an authorization policy is set to a DENY view on a web part and DisableALES=true, the runtime authorization is not performed. Hence, web part will still be presented (since the policy is ignored). However in the case of List items, the permission will already be applied on SharePoint and DisableALES=true will affect this. Hence, the List item will be hidden.

3. Add the following entries to `<SafeControls>`:

```
<SafeControl Assembly="BEA.SharePoint, Version=1.0.0.0,
Culture=neutral, PublicKeyToken=68b08a2fa869dfdc"
Namespace="BEA.SharePoint.Controls" TypeName="*" Safe="True" />
```

```
<SafeControl Assembly="BEA.SharePoint, Version=1.0.0.0,
Culture=neutral, PublicKeyToken=68b08a2fa869dfdc"
Namespace="BEA.SharePoint.Modules" TypeName="*" Safe="True" />
```

4. Add the following entry to the **httpModules** section.

```
<add name="CustHTTPModule" type="BEA.SharePoint.Modules.CustHTTPModule,
BEA.SharePoint, Version=1.0.0.0, Culture=neutral,
PublicKeyToken=68b08a2fa869dfdc " />
```

5. (Required only if custom content is published) The SafeMode section's PageParserPaths may be updated with the Virtual path in which custom content is required to be published. The custom content may be authorized via the tag library provided with the solution. An example of such a change is shown below:

```
<PageParserPaths>|
<PageParserPath VirtualPath="/Pages/*" CompilationMode="Always"
AllowServerSideScript="true" IncludeSubFolders="true"/>
</PageParserPaths>
```

6. Restart IIS server.

Deploy OES Authorization in SharePoint

Perform the following steps to the deploy the OES authorization feature in SharePoint:

1. Copy the `BEA_HOME\ales32-ssm\iis-ssm\sharepoint-ssm\lib\ALESAuthorizationFeature` directory to the following directory on the SharePoint server:

   ```
   C:\Program Files\Common Files\Microsoft Shared\web server
   extensions\12\TEMPLATE\FEATURES
   ```

2. To install the feature, open a command prompt and execute the following:

   ```
   C:\Program Files\Common Files\Microsoft Shared\web server
   extensions\12\BIN\STSADM.EXE” –o installfeature –name
   ALESAuthorizationFeature
   ```

3. Once installed, the feature may be activated for each web and sub-web (activated separately for webs and sub-webs) using **Site Settings>Modify All Site Settings>Site Features** page or by the command line as shown:

   ```
   “C:\Program Files\Common Files\Microsoft Shared\web server
   extensions\12\BIN\stsadm.exe" -o activatefeature -name
   ALESAuthorizationFeature -url http://sharepoint01/News
   ```

When authorization is activated against a sub-site, access to all the web parts in the sub-site’s web pages is secured. For example, if the authorization feature is deployed on `http://eagle/Reports`, access is controlled on all web parts under this sub-site.

If an authorization policy denies access to the `//app/policy/MOSS/Reports/Pages/Default.aspx/Announcements` resource, the Announcements web part will not appear when `http://eagle/Reports/Pages/Default.aspx` is opened.

Modify SSM Configuration

Perform the following steps:

1. Shut down Web Service SSM and copy `BEA_HOME\ales32-ssm\iis-ssm\sharepoint-ssm\lib\SampleIdentityAsseter.jar` to the `BEA_HOME\ales32-ssm\webservice-ssm\lib\providers\css` directory.
2. Configure the new identity assertion type ("sampletoken") at the Web Service SSM as described in "Sample Identity Asserter Configuration" on page 10-15.

   Note: The credential holder class (`com.bea.security.ssmws.credentials.TestCredHolderImpl`) is packaged in `BEA_HOME\ales32-ssm\iis-ssm\sharepoint-ssm\lib\ssmwsCustomAssertion.jar`.
3. Restart Web Service SSM so that it can pick up the latest configuration and deploy the provider.

Create SharePoint Resources in OES

If resources (like web's, lists etc.) are created in SharePoint after resource discovery is performed, these resources must be defined in OES.

The following section describes this manual creation of resources corresponding to webs, lists, items and web parts. As the resource model is based on URL's, the web page URL's would be used to create resources in OES. These resources must be defined under the root resource that was created by the config tool.

Webs

A SharePoint web is defined in OES using the web's URL. If the web URL is `http://<SharePoint_Server_Name>/web1`, a child resource named **web1** under the root resource.

The administrative URL's corresponding to the web are listed in `BEA_HOME\ales32-ssm\iis-ssm\sharepoint-ssm\adm\Discovery\AdmUrls.txt`. Because these URL's are invoked on administrative actions performed on the web and its children, they should be created as child resources of the **web1** resource.

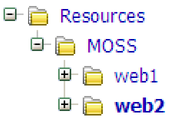
Lists & Items

Web creation will also create a set of lists depending upon the template used. These lists are incorporated into the resource tree. How they are defined depends on whether they are document or non-document lists.

For Document Lists...

For a List view URL of `http://<SharePoint_Server_Name>/web1/TestDocLib/Forms/AllItems.aspx`, create the resource hierarchy shown in Figure 10-1.

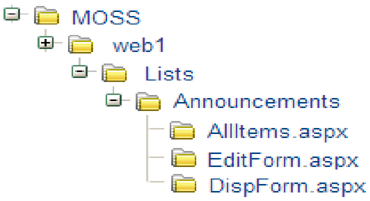
Figure 10-1 Document Lists Resource Hierarchy



For Non-Document Lists...

1. For a List view page of `http://<SharePoint_Server_Name>/web1/Lists/Announcements/AllItems.aspx`, create the resource tree shown in Figure 10-2.

Figure 10-2 Non-Document List Resource Hierarchy



2. Click on any item in a list and note the **ID** parameter in the URL, for example:

   ```
   http://<SharePoint_Server_Name>/web1/Lists/Announcements/DispForm.aspx?
   ID=2&Source=http%3A%2F%2Fsharepoint01%2Fweb1%2FLists%2FAnnouncements%2F
   AllItems%2Easpx
   ```

   The ID is used as a name of the non-document item and must be defined as a child resource of both `EditForm.aspx` and `DispForm.aspx`. This must be performed for all items within a non-document list.

   Note: An easy way to finding the ID's of an item is to hover the mouse over the item link and note the ID from the URL displayed on the browser's status bar.

Pages

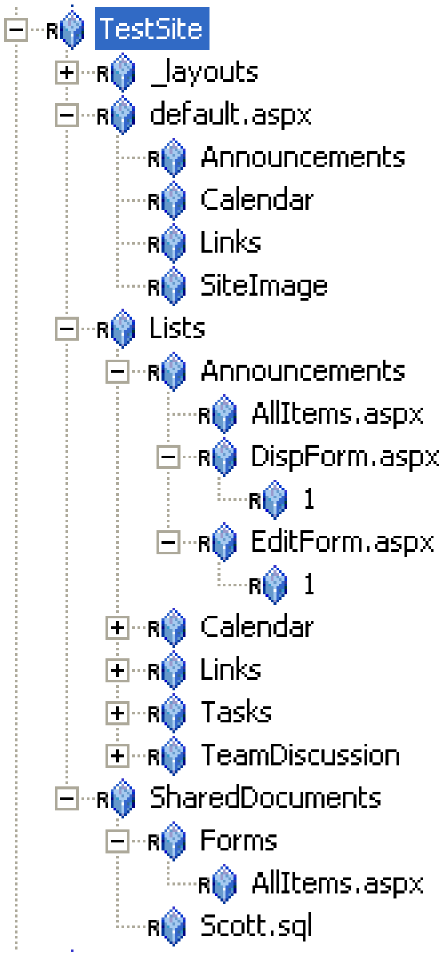
Pages in SharePoint exist either in document libraries or in the web base. The page may be defined in OES by creating a resource for each section of the URL.

Web Parts

Corresponding to a web, there may be a set of pages created for publishing content. A web part is one of the easiest and best way in which content is published in SharePoint. For authorization on web parts, the web parts may be created as resources in OES. Web Parts are created as sub resources of the page resource and the name of the resource is the display name of the web part.

An example as it would be displayed in the Administration Console is shown in Figure 10-3.

Figure 10-3 Web Parts Resource Hierarchy



Sample Identity Asserter Configuration

This section provides instructions for testing the solution with SharePoint's out-of-box windows-based authentication.

Note: The solution employs simple username assertion, which does not provide sufficient security for production environments.

Policy Updates

The users present in the directory used for authentication by SharePoint (AD by default) should be created in OES. The user name should be in lower-case. Policies should be created for these users and distributed to the Web Service SSM.

Example:

1. In the Entitlements Administration Application select the SharePoint organization in the left pane and click the Identities tab in the right pane.
2. If necessary, create a SharePoint identity directory. Under this directory create a user using the format `<computer_name>\<user_name>`, for example, `eagle\administrator`.

   Note: This is a user containing a backslash (\) in the name. Make sure to use all lowercase letters.

3. Define an authorization policy that denies access on a web part in a web page.

   For example, the following policy denies the `eagle\administrator` user access to `//app/policy/MOSS/Reports/Pages/default.aspx/Announcements`. As a result of this policy, the user will not see the 'Announcements' web part when accessing `http://eagle/Reports/Pages/Default.aspx`.

   ```
   Deny (view, //policy/MOSS/Reports/PAges/default.aspx/Announcements,
   //user/asi/eage\\administrator/)
   ```

SSM Configuration Updates

Perform the following steps:

1. On the SSM machine, copy `BEA_HOME\ales32-ssm\iis-ssm\sharepoint-ssm\lib\SampleIdentityAsserter.jar` to the following directory on the Administration Server machine:

   ```
   BEA_HOME\ales32-admin\lib\providers\css
   ```

2. Restart the Administration Server and log into the Administration Console.
3. In the left pane, expand the **Security Configuration** node and select the Web Service SSM used for securing SharePoint.
4. In the right pane, select **Java SSM 3.0, WS SSM 3.0** in the **Configuration Version** dropdown list and click **Apply**.
5. Remaining in the right pane, select the **Provider** tab and then open the **Authentication** tab.

6. On the Authentication tab, choose **Configure a new Sample Identity Asserter2**. For this identity asserter, choose the *sampletoken* as an **Active** type and make sure the "**Base64Decoding required** checkbox is not selected. Then click **Create** and **Apply**.

7. Return to the Authentication tab and click on `Reorder the Configured Authentication Providers`. Then make sure the `SampleIdentityAsserter2` is at the top of list and click **Apply**.

8. If any other authentication provider has been configured for the Web Service SSM, display the provider's **General** tab and select OPTIONAL in the **Control Flag field**. Then click **Apply**.

Adding New Identity Assertion Types

To add support for new assertion types "sampletoken" to the Web Services SSM:

1. To add the JAR file containing the holder class to the Web Service SSM's classpath, open `BEA_HOME/ales32-ssm/webservice-ssm/instance-name/config/WLESws.wrapper.conf` in an editor and add a line like the following:

   ```
   wrapper.java.classpath.40=
   BEA_HOME/ales32-ssm/iis-ssm/sharepoint-ssm/lib/
   ssmwsCustomAssertion.jar
   ```

   Note: The wrapper.java.classpath lines must increment sequentially.

2. To modify the mapping file for incoming messages, open `BEA_HOME/ales32-ssm/webservice-ssm/lib/com/bea/security/ssmws/soap/castor.xml` in an editor and add a line like the following:

   ```
     <class name="com.bea.security.ssmws.credentials.TestCredHolderImpl">
            <map-to cst:xml="sampletoken" />
            <field name="cookie" type="java.lang.String" >
                <bind-xml node="text"/>
            </field>
    </class>
   ```

3. To modify the mapping file for outgoing messages, open `BEA_HOME/ales32-ssm/webservice-ssm/lib/com/bea/security/ssmws/credentials/castor.xml` in an editor and add a line like the following in the <**mapping**> element:

   ```
   <class name="com.bea.security.ssmws.credentials.TestCredHolderImpl">
     <map-to cst:xml="sampletoken"
      cst:ns-uri="http://security.bea.com/ssmws/ssm-soap-types-1.0.xsd"
      />
           <field name="cookie" type="java.lang.String" >
   ```

```
                <bind-xml node="text"/>
            </field>
    </class>
```

4. Restart Web Service SSM.

   When the Web Services SSM is started, it will use the new holder implementation and the mapping entries to convert back and forth between the token's XML and Java representations.

SharePoint Configuration Updates

For using the identity asserter configured above, the following updates are required in the **appSettings** section of the SharePoint web configuration file (`C:\Inetpub\wwwroot\wss\VirtualDirectories\80\web.config` directory of the SharePoint deployment).

The value of the **token** key should be set to `LOGON_USER`. This is a header set by SharePoint that has the user id of the currently logged-in user (in the form of `<MachineName\<UserName>`, for example, `EAGLE\Administrator`). The value of this header is passed in the call to the Web Service SSM for asserting the identity of users coming to SharePoint.

The value of the **IdentityAsserterName** key should be set to `sampletoken`. This is the active token type for the identity asserter `*BEA_HOME*\ales32-ssm\iis-ssm\sharepoint-ssm\lib\SampleIdentityAsseter.jar` configured above.

After updating the web configuration, restart IIS.

Uninstall OES-SharePoint Integration

This section details the steps to be performed to uninstall OES from SharePoint.

1. Remove any entries that were added to `C:\Inetpub\wwwroot\wss\VirtualDirectories\80 directory`\web.config as described in "Modify SharePoint Web Configuration" on page 10-9.

2. Delete the following file:

   ```
   C:\Inetpub\wwwroot\wss\VirtualDirectories\80\web.config.preALESMOSSInte
   g
   ```

3. To deactivate the authorization feature, open a command line and enter:

```
C:\Program Files\Common Files\Microsoft Shared\web server
extensions\12\BIN\STSADM.EXE" -o deactivatefeature -name
ALESAuthorizationFeature -url http://sharepoint01/News -force
```

4. Repeat the previous step against all the sub-sites where the authorization feature was activated.
5. To uninstall the authorization feature, open a command line and enter:

```
C:\Program Files\Common Files\Microsoft Shared\web server
extensions\12\BIN\STSADM.EXE” –o uninstallfeature –name
ALESAuthorizationFeature -force
```

6. Delete the `ALESAuthorizationFeature` directory from `C:\Program Files\Common Files\Microsoft Shared\web server extensions\12\TEMPLATE\FEATURES`.
7. Delete the custom error page (`custError.aspx`) from the `C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\12\TEMPLATE\LAYOUTS` directory.
8. If the tag library is used in SharePoint pages, use the SharePoint designer to remove them.
9. Use a text editor (for the default template) or the SharePoint Designer (for custom master pages) to remove the delegate control entry made in the master pages.
10. In the `C:\WINDOWS\assembly` directory, right-click `BEA.SharePoint.dll` and select **uninstall**.

OES Adapter for Sun Identity Manager

The OES Adapter is a plug-in to the Sun Identity Manager that enables the bi-directional propagation of users and user attributes between Sun Identity Manager and OES.

This document contains detailed, step-by-step instructions on how to configure the adapter in Sun Identity Manager, and how to set up active sync from the adapter.

After completing these tasks, the user operations in Sun Identity Manager will take effect in OES, and the user operations in the Administration Console will be synced into Sun Identity Manager. The sync interval from OES to Sun Identity Manager is configurable.

Set Up OES Resource in Sun Identity Manager

Perform the following steps to set up the adapter as a resource in Sun Identity Manager:

1. Stop the Sun Identity Manager container.
2. Copy the following files from `ales32-admin` to `idm/WEB-INF/lib`:

```
ales32-admin/lib/asi_classes.jar
ales32-admin/lib/asitools.jar
ales32-admin/lib/jsafeJCE.jar (WLS 8.x) or jsafeJCEFIPS.jar (WLS 9.x)
ales32-admin/lib/log4j.jar
ales32-admin/lib/ssladapter.jar
ales32-admin/lib/sslplus.jar
ales32-admin/lib/webservice.jar
ales32-admin/lib/webserviceclient.jar
ales32-admin/lib/providers/ojdbc14_g.jar
```

```
ales32-admin/lib/providers/jconn2.jar
ales32-admin/lib/providers/jconn3.jar
ales32-admin/data/SunIMAdapter/lib/ALESResourceAdapter.jar
```

3. Copy `ales32-admin/data/SunIMAdapter/forms/*` to `idm/sample/forms`.
4. Copy `ales32-admin/data/SunIMAdapter/images/ALES.gif` to `idm/applet/image`.
5. Add execute permission for the following scripts on UNIX platforms:

   ```
   ales32-admin/bin/install_user_change_schema_oracle.sh
   ales32-admin/bin/install_user_change_schema_sybase.sh
   ```

6. Run the following scripts to set up table space for the `UserChangeDBAuditor`, which is configured in a subsequent step.

   For Oracle, run:

   ```
   ales32-admin/bin/install_user_change_schema_oracle.bat|sh
   ```

   For Sybase, run:

   ```
   ales32-admin/bin/install_user_change_schema_sybase.bat|sh
   ```

   You need to supply your OES credentials in order for the scripts to make the necessary changes.

7. Start the Sun Identity Manager container.
8. Log in to the Sun Identity Manager console with the Configurator id. The default password is *configurator.*
9. Configure the resource type:
   a. Click Configure at the top of the menu.
   b. Click Managed Resource in the sub-menu.
   c. Click the Add Custom Resource button. Enter `com.bea.adapter.ALESResourceAdapter` as the Resource Class Path under Custom Resource, and click Save.
10. Configure the OES resource:
    a. Click Resource at the top of the menu.
    b. Select New Resource in Resource Type Action from the dropdown list.
    c. Select ALES from the dropdown list of Resource Type, and click New.

d. In Welcome Create ALES Resource Wizard, click Next.

e. Enter the resource parameters as follows, and then click Test Configuration. Make sure that the OES Administration servers are currently running.

   - Host: The host name or IP address of Administration Server
   - TCP port: The port number for BLM server (default=7011)
   - Username: The user who has privilege to manager users in OES, e.g. "admin"
   - Password: The password of user manager of OES admin
   - Directory of Keystore: The full path to the ssl dir in the OES admin. If the IDM is not located on the same machine as OES admin then the ssl dir should be copied to the IDM machine

f. If the test configuration is successful, status is displayed as `Test connection succeeded for resource(s): ALES`. Click Next.

   If the test configuration is not successful, an error message is displayed. You need to check the Resource parameters and make sure that the OES Administration servers started. After you have done this, test again.

g. Configure user attributes, and click Next.

h. Accept Identity Template settings, and click Next.

i. Enter your Resource Name in Identity System Parameters, accept the other default settings, and then click Save.

Enable Active Sync for OES Resource

An OES Audit provider is used to record user-related operations in the OES system. This is done so that the adapter for Sun Identity Manager can sync these changes automatically.

The procedure you follow to enable active sync for the OES resource depends on whether you are using the WebLogic 9.x/10.x or WebLogic 8.1 SSM. When you use the WLS 9.x/10.x SSM, configure security providers using the WebLogic Administration Console, rather than the Administration Console.

Using the WebLogic 9.x SSM

1. Start the Administration Servers.

2. Log in to the WebLogic Server Administration Console on the system on which the WebLogic 9.x SSM is installed, `https://hostname:port/console`.
3. Click Lock and Edit on the left top of the page.
4. Create an instance of UserChangeDBAuditor. There should be no more than one User Change DB Auditor in one OES domain.
   a. Click on Security Realms in the left panel.
   b. Click on your configured security realm in the middle of the right main panel.
   c. Click Providers on the top menu of realm.
   d. Click Auditing in the sub menu.
   e. Click New to configure a new Audit provider.
   f. Enter a name and select `UserChangeDBAuditor` as type, and click OK.
   g. Click the name you entered and go to the provider setting page.
   h. Click the Provider Specific top menu, and enter the JDBC parameters. The values should equal those of the OES configuration.
   i. Click Save.
5. Click Release Configuration on left top of page.
6. Restart the Administration servers to make the UserChangeDBAuditor take effect.

Using the Weblogic 8.1 SSM

1. Start the Administration Server.
2. Log in to the OES Administration Console by entering the following in a browser:

   `https://<host>:<port>/asi`

   where <*`host`*> is the server host and <*`port`*> is port (default = 7010)
3. Create a UserChangeDBAuditor as follows:
   a. In the left pane, select the **asiadmin** SSM under the **adminconfig** SCM.
   b. Click **Providers** in the right pane and then select the **Auditors** tab.

   c. On the **Auditors** tab, click on **Configure a new User Change DBAuditor**. Then accept the default name and click **Create**. Finally, open the **Details** tab, enter the JDBC parameters, and click **Apply**.

      Note: The JDBC parameter values should equal those of the OES database configuration.

4. Return to the left pane and select the **Deployment** node at the bottom of the tree. Then select the **Configuration** tab in the right pane.
5. On the **Configuration** tab, select the **Security Configuration** checkbox and then click **Distribute Configuration Changes**.
6. Click **Refresh** until the distribution is 100% complete.
7. Restart the Administration Server.

Set Up Active Sync in Identity Manager

1. Log in to the Identity Manager console with the Configurator id. The default password is `configurator`.
2. Configure Active Sync for the OES Resource:
   a. Click Resource at the top of the menu.
   b. Select the OES Resource in Resource List by clicking on the checkbox. Then, select Active Sync Wizard in the -- Resource Actions -- dropdown list.
   c. Select the Use Wizard Generated Input Form ratio button for Input Form Usage. Then, select Advanced for Configuration Mode and click Next.
   d. Configure Active Sync Running Settings on demand.
   e. Configure General Active Sync Settings. Enter JDBC values to match those of the OES database configuration. Click Next.
   f. On the Event Types page, accept the default values and click Next.
   g. On the Process Selection page, accept the default values and click Next.
   h. On the Target Resources page, add the Identity Manager resources that need to sync with OES resource to Target Resources.

i. On the Target Attribute Mappings page, you can use add and remove to set up the mapping between OES attributes and Identity Manager attributes. After you have finished the attribute-mapping settings, click Save to finish.