BEA AquaLogic Enterprise Security™®

ALES Integration Guide

Version 3.0
Document Revised: January 2008

1. Introduction

2. Securing Web Servers

3. Securing WebLogic Servers

4. Securing Applications Developed Using BEA Workshop for WebLogic

5. Securing AquaLogic Data Services Platform

6. Securing WebLogic Portal Applications

7. Storing and Versioning ALES Policy with ALER

8. Securing AquaLogic Service Bus Runtime Resources

9. Securing ALES Components

A. ALES Adapter for Sun Identity Manager

CHAPTER 1

Introduction

This section includes the following topics:

Document Scope and Audience

This document primarily intended for administrators who are responsible for configuring the ALES components, integrating ALES into application environments, managing interaction between an applications and ALES, 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 ALES documentation and resources, see "Related Documentation" on page 1-2.

It is assumed that readers understand Web technologies and have a general understanding of the Microsoft Windows or UNIX operating system being used. Prior to using this document, you should have a general understanding of ALES's principal components and architecture. Read the *Introduction to BEA AquaLogic Enterprise Security* for conceptual information that is helpful in understanding how the product works. This document provides information about integrating ALES with application environments and defining policies to secure application resources.

Guide to this Document

This document is organized as follows:

Related Documentation

For information about other aspects of AquaLogic Enterprise Security, see the following documents:

- Getting Started Tutorials—This guide provides tutorials on installing ALES 3.0 and securing applications.
- Introduction to BEA AquaLogic Enterprise Security—This document provides overview, conceptual, and architectural information for AquaLogic Enterprise Security.
- Administration Server Installation Guide—This document describes installation of the ALES Administration Server.
- SSM Installation and Configuration Guide—This document describes installing and configuring Security Service Modules for AquaLogic Enterprise Security.
- Policy Managers Guide—This document defines the ALES policy model and describes how to generate, import and export policy data.
- Administration Reference—This document 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.
- Programming Security for Web Services—This document describes how to implement security in web servers. It includes descriptions of the Web Services Application Programming Interfaces.
- 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 ALES documentation's main page.

Contact Us!

Your feedback on BEA documentation is important to us. Send us e-mail at docsupport@bea.com if you have questions or comments. Your comments will be reviewed directly by the BEA professionals who create and update the documentation.

In your e-mail message, please indicate the software name and version you are using, as well as the title and date of your documentation. If you have any questions about this version of BEA AquaLogic Enterprise Security, or if you have problems installing and running BEA AquaLogic Enterprise Security products, contact BEA Customer Support through BEA WebSupport at http://www.bea.com. You can also contact Customer Support by using the contact information provided on the Customer Support Card, which is included in the product package.

When contacting Customer Support, be prepared to provide the following information:

- Your name, e-mail address, phone number, and fax number
- Your company name and company address
- Your machine type and authorization codes
- The name and version of the product you are using

A description of the problem and the content of pertinent error messages.

CHAPTER 2

Securing Web Servers

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

Overview

An ALES Web Server SSM is used to secure web server resources hosted on IIS and Apache Web Server. For a description of features and capabilities see Web Server Security Service Module.

Deploying the Web Server SSM also requires deployment of Web Services SSM on the same machine. The Web Server SSM communicates with the ALES 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 Configuring the Environmental Binding for IIS Server in 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) on the server using the **LoadModule** and **WLESConfigDir** directives. For instructions, see Configuring the Environmental Binding for the Apache Web Server in 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 Web Single Sign-On with ALES Identity Assertion" on page 2-10.

Note: For a description of use cases, also see Single Sign-On Use Cases.

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 ALES 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 2-4.
2. Define the web server resources in ALES as described in “Define Web Server Resources in ALES” 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 2-9.
5. For testing purposes, set up the web server as described in “Set Up and Test the Sample Application” on page 2-9.
6. (Optional) Set up SSO as described in “Implementing Web Single Sign-On with ALES Identity Assertion” on page 2-10.

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 set-by-step instructions, see *Configuring Security Providers* in the Administration Console's 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.
Log4 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 in ALES

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

- Resources are identified by canonical name. For example, for a web server with the names www.bea.com, www.beasys.com, www.web.internal.bea.com, and 204.236.43.12, the canonical name is www.bea.com.
- The Web Server SSM provides HTTP headers, cookies, query arguments, and form values to the ALES 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 ALES 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 ALES 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 ALES 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 Web Server Application Resources 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.

Table 2-2 Sample Web Server Application Resources Authorization Policies (Continued)

Policy	Description
`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 is started, 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\ales30-ssm\iis-ssm\instance\<instance>\templates`
   - NamePasswordForm.html—(Apache only) located in `BEA_HOME\ales30-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 IIS: `http://<hostmachine.cookiedomain>:8088/test/foo.html`.

5. When the browser is redirected to the logon form, enter the System username/password (the defaults are *system* and *weblogic*) and click **OK**. You are granted access to `foo.html`.

Implementing Web Single Sign-On with ALES Identity Assertion

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.

Web Server SSM to WebLogic Server SSM Single Sign-On

To implementing 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 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.

CHAPTER 3

Securing WebLogic Servers

In addition to providing links to other ALES 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

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 ALES documents, particularly the Policy Manager's Guide, Getting Started

Tutorials, and the help systems for the Administration Console and Entitlements Management Tool.

Securing Administrative Access to WebLogic Server

This chapter describes how to integrate ALES with WebLogic Server and define a policy for 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 ALES 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/10.0, are described in "WebLogic 9.x/10.0 Security Providers" on page 3-4.

2. Define the WebLogic administrative user in ALES 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 ALES 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.

Table 3-1 Portal Security Configuration (Continued)

Security Provider	Configuration Settings
WebLogic Credential Mapper	Clear the **Credential Mapping Deployment Enabled** checkbox.
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 ALES consoles.

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

ALES Security Providers Extension

To install the plugin:

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

   `BEA_HOME/ales30-ssm/wls9-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 ALES security providers using the WebLogic Server 9.x/10.0 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 ALES 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 ALES Console

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

1. Log into the ALES 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/<directory>` where <*directory*> is the directory name.
ASI Role Mapper	Set the **Identity Directory** to the directory created in step 1. Set the **Application Deployment Parent** to `//app/policy/<directory>` where <*directory*> is the directory name.

WebLogic Administrative User

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

1. Launch the ALES 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.

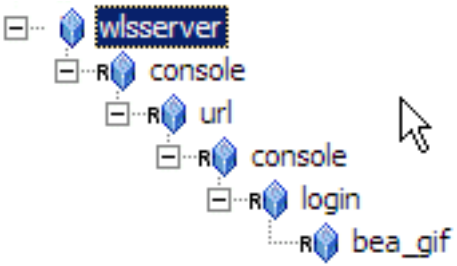
WebLogic Server Resources

WebLogic Server components must be defined in ALES as ALES resources. To create these resources using the ALES 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 ALES `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 ALES to protect a cluster of WebLogic Server 9.x/10.x domain instances.

Topology

This document assumes the following deployment of ALES components:

- The ALES 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 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 ALES 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/ales30-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 ALES 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 Applications Developed Using BEA Workshop for WebLogic

This section describes how to use ALES with BEA Workshop for WebLogic Platform and discusses how to use the ALES Annotations Plugin and the ALES Tag Library.

Overview

The ALES Annotations Plugin or ALES Tag Library for Workshop can be used to secure applications being deployed in WebLogic Workshop.

ALES Annotations Plugin

The ALES 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 ALES.
- 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 ALES annotations and then securing it with ALES policy. The integration tasks provided here are based on those instructions.

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

Set Up the ALES Annotations Plug-in

The ALES 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:

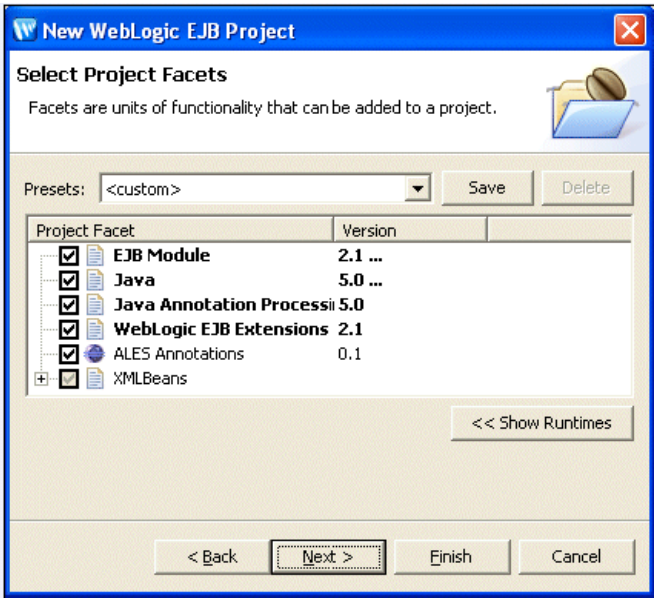
1. Copy the appropriate file from `BEA_HOME/ales30-admin/lib/eclipsePlugins` to the following directory:

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

2. Restart Workshop.

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

Figure 4-1 Project Facets



Using ALES Annotations in a WebLogic Bean Class

This section contains the following topics:

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 ALES 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:

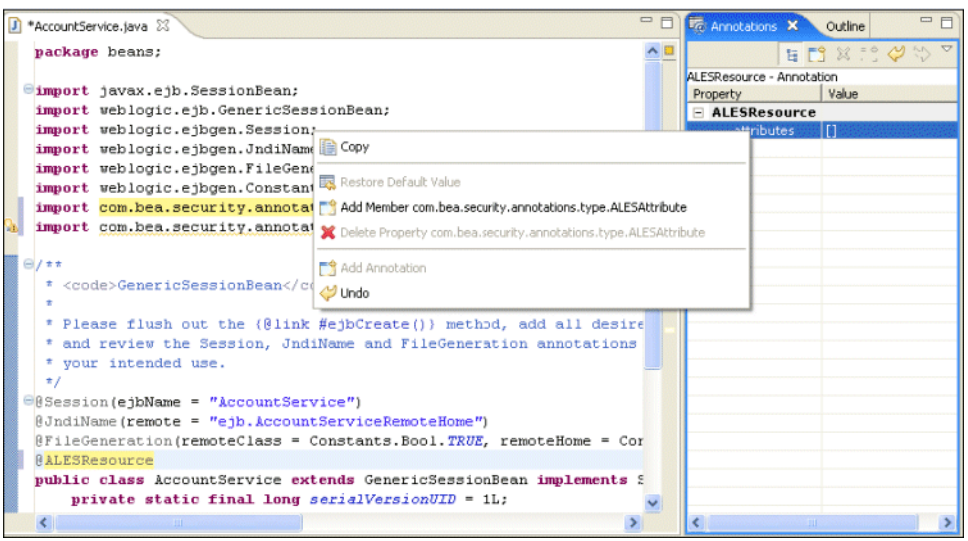
```
import com.bea.security.annotations.type.ALESResource;
import com.bea.security.annotations.type.ALESAttribute;
```

2. To define the Bean class `AccountService` as an ALES 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



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 ALES resources and assign ALES attributes to them using the ALES Annotations plug-in. For example, annotate the `ejbCreate()` method to define it as an ALES resource with the ALESAttribute `operation = create`.

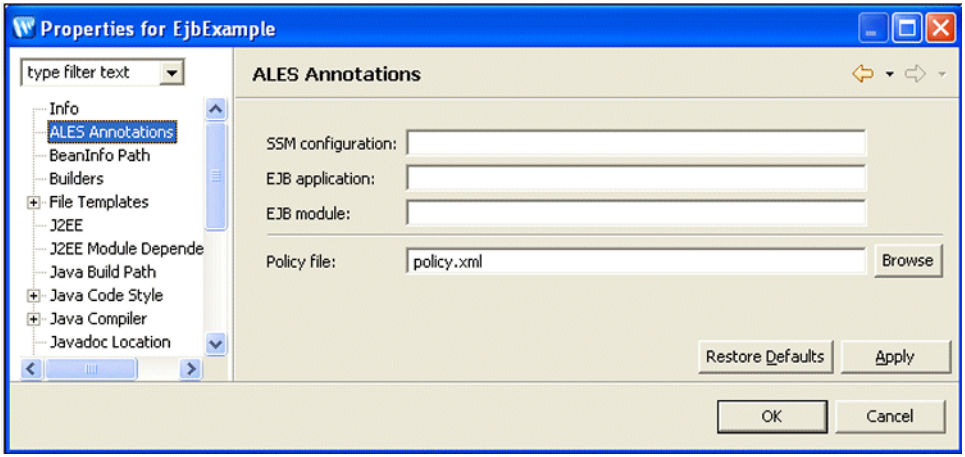
```
@ALESResource(attributes={@ALESAttribute(name = "operation", value =
"create")})
```

Add ALES Information to the Project

To define ALES-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 ALES Annotation Project Properties



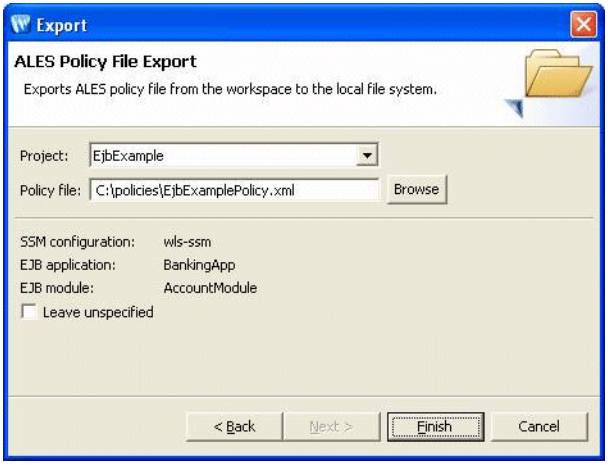
2. Select **ALES annotations** and configure the following four properties:
   - SSM configuration—SSM to bind to the EJB application with; for example,`wls-ssm`.
   - EJB application—EJB application to deploy; for example, `BankingApp`.
   - EJB module—EJB module to deploy; for example, `AccountModule`.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

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 ALES Policy File Export Window



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 into ALES

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

1. If you have not already done so, start the ALES 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 `ALES_ADMIN_HOME/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 `ALES_ADMIN_HOME/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:

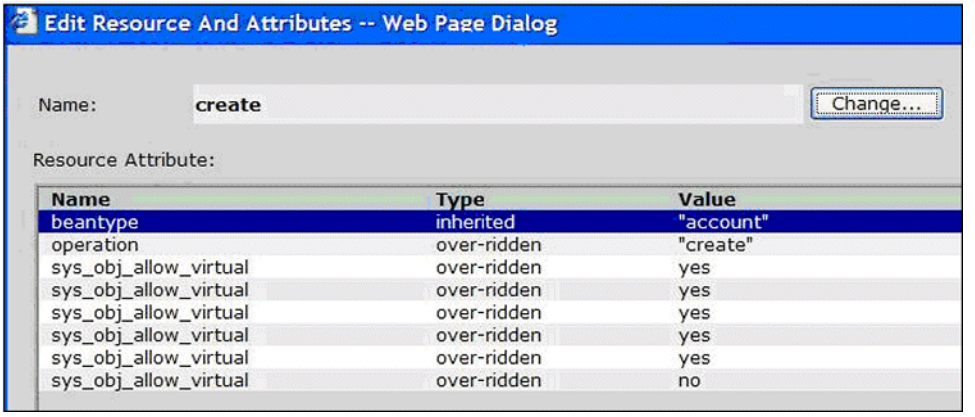
   ```
   C:\ALES_ADMIN_HOME/bin> policyIX -import config.xml EjbExamplePolicy.xml
   ```

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

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

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

Figure 4-5 Resource Attributes for the create Method



Define Policies for the Imported Policy File

After importing the policy file into ALES, you may create policies that define access rules for the resources defined with ALES 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"
```

ALES Tag Library for Workshop

The ALES Tag Library for Workshop can be used to add ALES 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 ALES Tag Library are:

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

ALES Tag Library Tags

The following tags are available in the ALES Tag Library for Workshop. See "ALES 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 ALES. This is an event tag that provides input to the ALES 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 ALES 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 ALES tags to a JSP page. The integration tasks provided here are based on those instructions.

1. Add the ALES Tag Library to Workshop as described in "Add the Tag Library to Workshop" on page 4-10
2. Add ALES tags to the JSP pages as described in "Using ALES Tags in JSP Pages" on page 4-11.
3. In ALES, 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 4-14.

Add the Tag Library to Workshop

A file named `alestags.jar` contains the ALES tag library and supporting classes. This file is packaged in one of the following WebLogic version-specific files located in the `BEA_HOME\ales30-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\ales30-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 **ALES 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 **ALES 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 ALES Tags.

Using ALES Tags in JSP Pages

The ALES 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 ALES tags in a JSP page.

Listing 4-1 Example JSP Page With ALES 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 ALES 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 ALES Administration Console or Entitlements Management Tool 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) ALES `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 ALES 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 ALES 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 ALES 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 ALES tag library does not implement its own set of iterators. The data returned by the ALES 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 ALES 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 ALES 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 ALES tags can be tested even if the SSM is not deployed. ALES tags will allow both ‘allow’ and ‘deny’ cases to display on the page and data retrieval methods return empty results.

ALES Tag Library Reference

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

isAccessAllowed

The `isAccessAllowed` tag calls the ALES 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

Table 4-1 isAccessAllowed

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”>
<%-- An attribute to pass to the ALES 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”>
<%-- An attribute to pass to the ALES application context for this call--%>
```

```
<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 ALES 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">
<%-- An attribute to pass to the ALES application context for all calls on
this page--%>
 <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>
<%-- An attribute to pass to the ALES application context for this call--%>
 <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 ALES 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
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 ALES 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 4-25, 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

Table 4-5 isUserInRole

Attribute	Return Type	Description	Required
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 ALES 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 ales 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 ALES. The body of this tag can also take an `attribute` tag, as described in "Attribute" on page 4-25. 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 ALES tags to pass extra attributes down in the ALES application context. Any of these variables can then be used to write constraints against in ALES policies.

Table 4-8 attribute

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

<%-- An attribute to pass to the ALES application context for this call--%>

<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>
```

CHAPTER 5

Securing AquaLogic Data Services Platform

This section provides information about securing ALDSP 2.5 and ALDSP 3.0 data services. It includes the following topics:

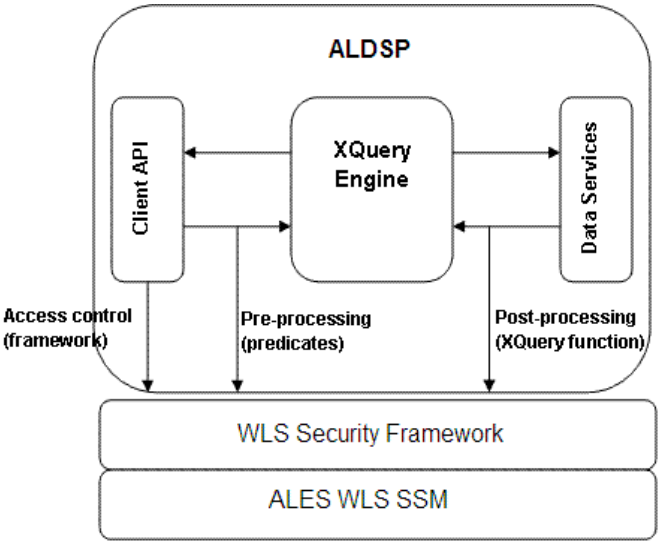
Overview

ALES can be used to manage access control to entire ALDSP data services or specific data service elements. In addition to simply returning an authorization decision that grants or prevents access, ALES policies can return information to ALDSP 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 5-17 and "Post-Processing Data Redaction" on page 5-22.

The following diagram illustrates how ALES components can be integrated with ALDSP.

Figure 5-1 ALDSP Integration Overview



Use-Case

Integration with ALDSP supports the following use-case scenario:

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

Prerequisites

The document assumes the following:

- Installation of WebLogic Server and ALDSP.

  NOTE: Depending on the ALDSP version, you may have to obtain a patch from BEA support so that ALDSP works correctly with ALES 2.6 and ALES 3.0:

  - If using ALDSP 2.5, obtain and apply ALDSP patch for CR318373.
  - If using ALDSP 3.0, obtain and apply ALDSP patch for CR354500.
  - If using ALDSP 3.2, no additional patches are required.

- Installation of the ALES Administration Server.
- Installation of the WLS SSM (for WebLogic Server 9.x, 10.0) or WLS 8.1 SSM (for WebLogic 8.1) on the ALDSP machine as described in the SSM Installation and Configuration Guide.

  NOTE: Be sure to make the necessary changes to `WLESarme.properties` as described in those chapters.

- Creation of the SSM instance.

Integration Tasks

The major integration tasks are:

Note: In addition to the steps below, additional tasks are required to define policies for data redaction. For pre-processing data redaction, see "Pre-Processing Data Redaction" on page 5-17. For post-processing data redaction, see "Post-Processing Data Redaction" on page 5-22.

1. Create the startWebLogic.cmd file for the ALDSP domain as described in the SSM Installation and Configuration Guide.
2. Create a file named `security.properties` that specifies the ALDSP 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. For information, see "Define Security Providers" on page 5-4.
4. In ALDSP, enable security on the data service elements to be secured as described in "Enable ALDSP Elements for Access Control" on page 5-4.
5. Define ALDSP identities in ALES. This chapter provides instructions for identities for the RTLApp sample application. See "Define ALDSP Identities in ALES" on page 5-5.
6. Define ALDSP resources in ALES. This chapter provides instructions for resources that represent RTLApp resources. See "Define ALDSP Resources in ALES" on page 5-5.

7. Define the Authorization and Role Mapping policies that secure RTLApp as described on "Define Policies for ALDSP" on page 5-9.
8. Distribute the policies to the SSM by following the instructions in "Distribute Policies" on page 5-13.

Define Security Providers

The following table provides information about ALES providers for securing ALDSP data services.

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 ALDSP Elements for Access Control

Access control must be set on the data service elements to be secured so that ALES 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 ALES:

1. Log in to the ALDSP console using an administrative account.
2. Browse to and select the RTApp data services elements to be secured. For example:
   a. Expand **RTLServices/OrderSummaryView** and select the **Security** tab.
   b. On the **Secured Elements** tab, expand **elements** and select **OrderSummary > OrderDate** as an element to be secured.

c. Repeat these steps for **CustomerView.ds > CUSTOMER > ORDERS> ORDER_SUMMARY > OrderDate**.

Define ALDSP Identities in ALES

To create the Identity directory and users:

1. In the ALES 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 ALDSP Resources in ALES

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

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, check the **Distribution Point** and **Allow Virtual Resources** checkboxes 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`

ALDSP 2.5 Resources

Figure 5-2 shows the ALDSP resource tree with all nodes expanded except the RTLApp node. For ALDSP 2.5, you must create the resources shown in both Figure 5-2 and Figure 5-3.

Figure 5-2 ALDSP 2.5 Resource Tree with RTLApp Node Collapsed

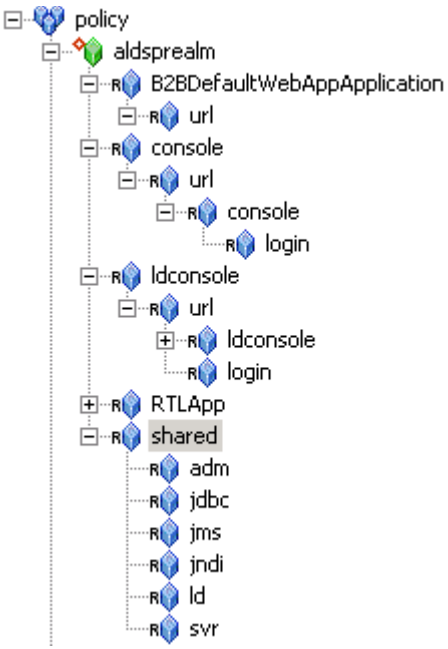
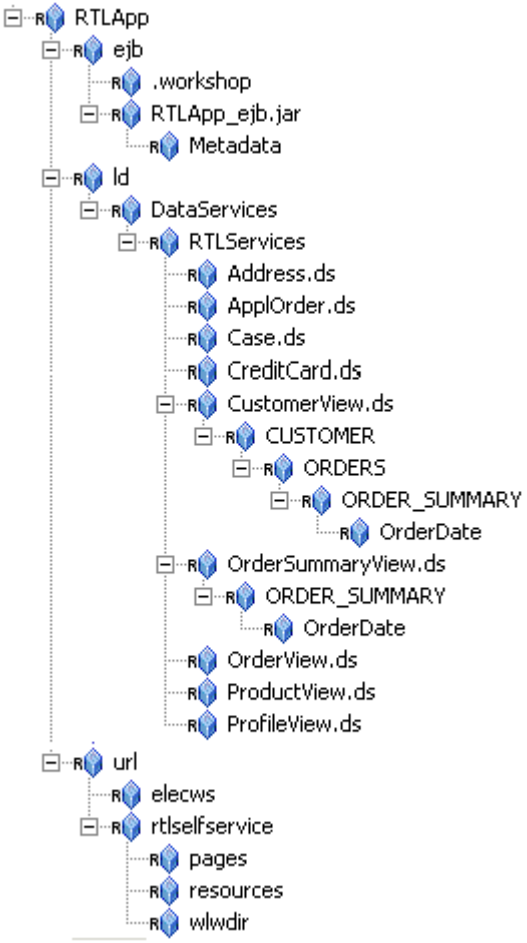


Figure 5-3 ALDSP 2.5 Resource Tree with RTLApp Node Expanded



ALDSP 3.0 Resources

Figure 5-4 shows the ALDSP resource tree with all nodes expanded except the RTLApp node. You must create the resources shown in both Figure 5-4 and Figure 5-5.

Figure 5-4 ALDSP Resource Tree with RTLApp Node Collapsed

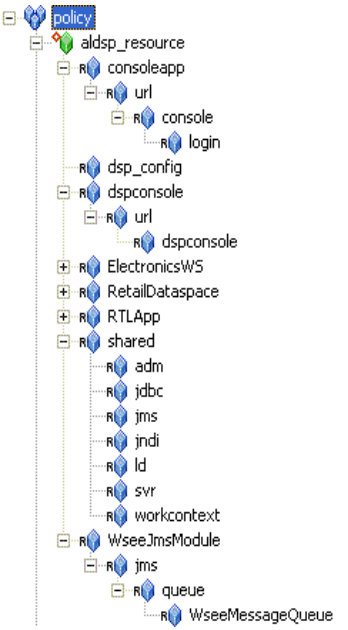
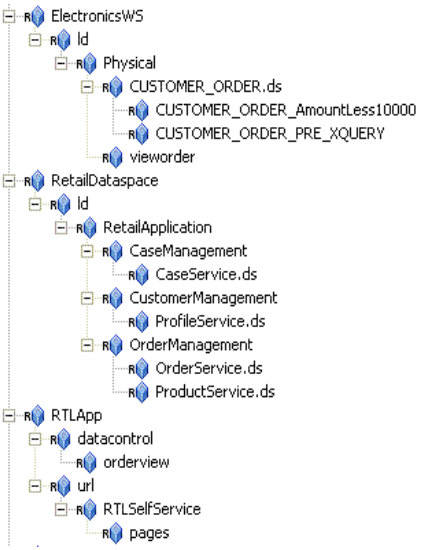


Figure 5-5 ALDSP Resource Tree with RTLApp Node Expanded



Define Policies for ALDSP

Because the sample applications provided with ALDSP 2.5 and ALDSP 3.0 are different, they require different policies. These are documented in the following sections:

- "Policies for ALDSP 2.5" on page 5-9
- "Policies for ALDSP 3.0" on page 5-11

Policies for ALDSP 2.5

The Authorization and Role Mapping policies needed to the secure the ALDSP 2.5 sample application are described below.

Authorization Policies

The following policies allow the `Admin` role to boot WebLogic Server and perform administrative tasks:

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

```
grant(any, [//app/policy/ aldsprealm
/RTLApp/ejb,//app/policy/aldsprealm/RTLApp/ld,//app/policy/aldsprealm/RTLA
pp/url/rtlselfservice/pages], [//role/Admin]) if true;

grant(any, [//app/policy aldsprealm
/RTLApp/ejb/RTLApp_ejb.jar/Metadata,//app/policy/aldsprealm/RTLApp/ejb/RTL
App_ejb.jar], [//role/Admin]) if true;

grant([any,//priv/create], //app/policy/ aldsprealm /RTLApp/ejb/.workshop,
//role/Admin) if true;

grant(any, [//app/policy/ aldsprealm
/console,//app/policy/aldsprealm/shared/svr,//app/policy/aldsprealm/shared
/adm], //role/Admin) if true;
```

The following policy gives `Admin` user full access to the LD console.

```
grant(//priv/any, //app/policy/aldsprealm/ldconsole/url/ldconsole,
//role/Admin) if true;
```

Perform the following steps to create the Authorization policies shown above.

1. Expand the **Policy** node in the left pane and select **Authorization Policies**. Then click **New** in the right pane.
2. On the **Create Authorization Policy** dialog, select the **Grant** radio button.
3. On the **Policy Subjects** tab, select **Admin** in the **Roles** list and click **Add**.
4. Repeat these steps for the remaining authorization policies.

   Note: If a policy specifies multiple resources for a single privilege and role, you may specify these resources in one policy.

Role Mapping Policies

The following policy assigns the `Everyone` role to all users in the `aldsprealm` directory.

```
grant(//role/Everyone, //app/policy/aldsprealm,
//sgrp/aldspusers/allusers/) if true;
```

The following policy assigns the `weblogic` user to the `Admin` role within `aldsprealm`.

```
grant(//role/Admin, //app/policy/aldsprealm, //user/aldspusers/weblogic/)
if true;
```

To create the Role Mapping policies above, perform the following steps.

1. Expand the **Policy** node in the left pane and select **Role Mapping Policies**. Then click **New** in the right pane.
2. On the **Create Role Mapping Policy** dialog, select the **Grant** radio button.
3. On the **Resources** tab, select `aldsprealm` and click **Add**.
4. On the **Policy Subjects** tab, select `allusers` in the **Roles** list and click **Add**.
5. Repeat these steps to define the other Role Mapping policy.

Policies for ALDSP 3.0

This section describes how to create the Authorization and Role Mapping policies to protect the ALDSP 3.0 sample application.

Authorization Policies

Define the Authorization policies shown in Table 5-1:

Table 5-1 Authorization Policies for ALDSP 3.0

Policies	Description

grant(any, //app/policy/aldsprealm/RTLApp/url/RTLSelfService, [//sgrp/aldspusers/LDSampleUsers/,//sgrp/aldspusers/aldsp_admins/,//sgrp/aldspusers/Administrators/]) if true;
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;
grant(any, //app/policy/aldsprealm/RetailDataspace/ld, [//sgrp/aldspusers/Administrators/,//sgrp/aldspusers/LDSampleUsers/]) if true;
	Grants Admin Role and/or weblogic user permission to boot the WLSand perform administrative tasks.

grant(any, //app/policy/aldsprealm/dspconsole, //sgrp/aldspusers/aldsp_admins/) 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.

Table 5-1 Authorization Policies for ALDSP 3.0

Policies	Description

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/RTLSelfService, [//sgrp/aldspusers/LDSampleUsers/,//sgrp/aldspusers/aldsp_admins/,//sgrp/aldspusers/Administrators/]) if true;	Allows all users to access the sample application.

grant(any, //app/policy/aldsprealm/shared/ld, //role/DataServiceAdmin) if true;
grant(any, [//app/policy/aldsprealm/shared/svr,//app/policy/aldsprealm/shared/adm,//app/policy/aldsprealm/consoleapp], [//role/Admin,//role/DataServiceAdmin]) if true;
grant(any, //app/policy/aldsprealm/ElectronicsWS/ld/Physical/CUSTOMER_ORDER.ds, //role/DataServiceAdmin) if true;
	Grants permission for data services.

Role Mapping Policies

Define the Authorization policies shown in Table 5-2:

Table 5-2 Role Mapping Policies for ALDSP 3.0 Sample Application

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.
grant(//role/DataServiceAdmin, //app/policy/aldsprealm/dspconsole, //sgrp/aldsp/aldsp_admins/) if true;	Grants the `aldsp_admins` group DataServiceAdmin role within the `dspconsole` application.

Distribute Policies

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

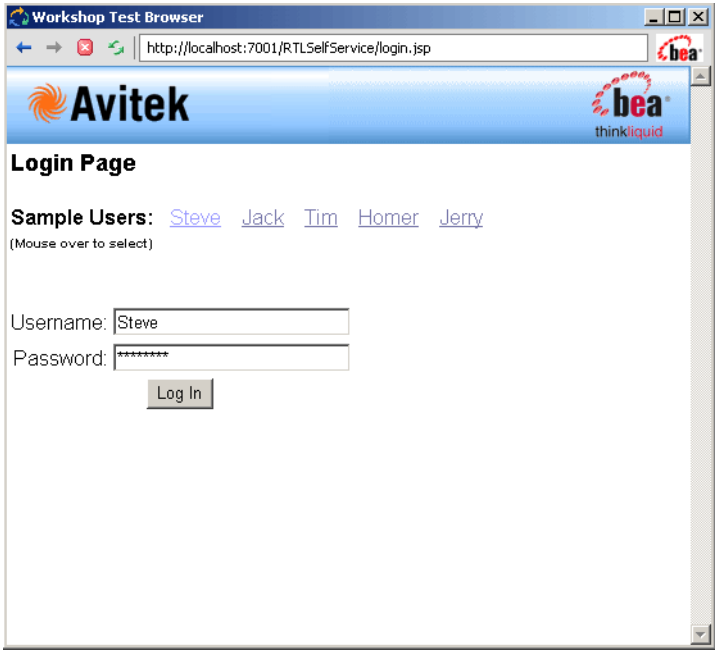
1. In the ALES Administration Console's left pane, select the **Deployment** node.
2. On the **Policy** tab in the right pane, select the checkbox before the name of the ALDSP resource parent and click Distribute Policy.
3. If you made any changes to the SSM configuration used to secure the ALDFSP domain, display the **Configuration** tab and select the checkbox for the SSM configuration. Then click click **Distribute Configuration Changes**.

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

Note: These steps may vary slightly depending on the ALDSP version being used.

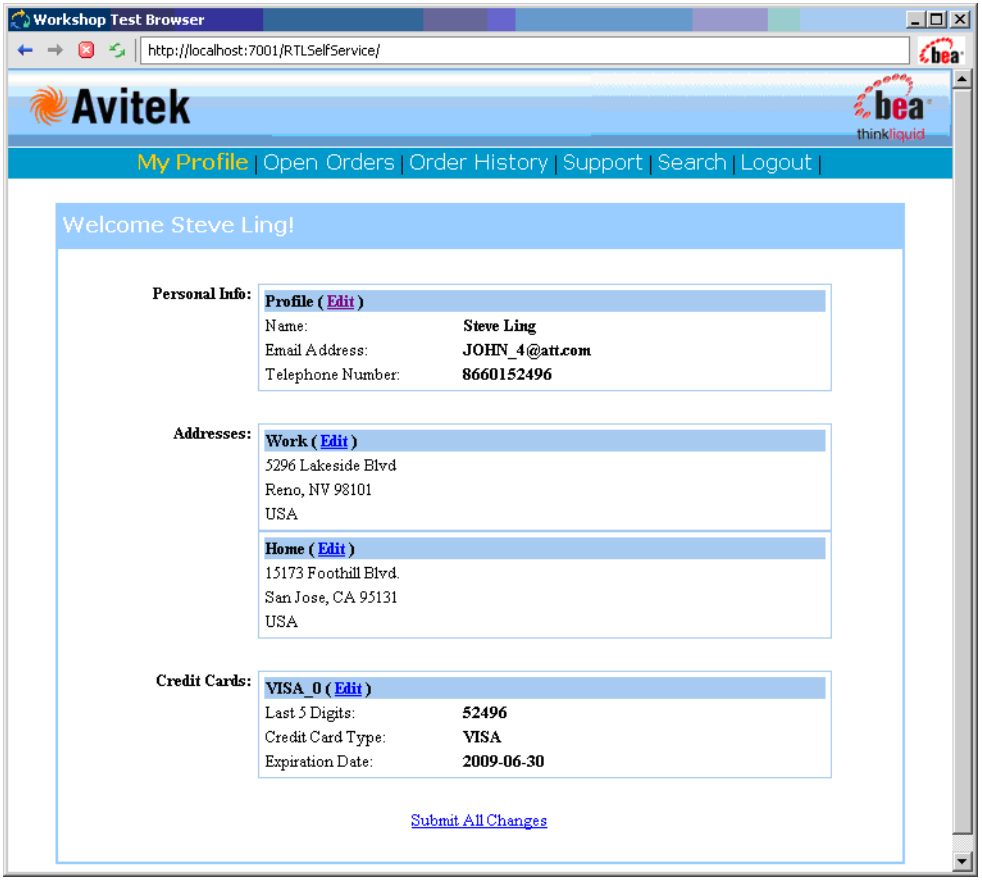
1. Start a WebLogic Server instance by changing to the `<bea_home>\<weblogic_home>\samples\domains\ldplatform` directory and running `startWebLogicALES.cmd` (Windows) or `startWeblogicALES.sh` (UNIX).
2. Access the RTLApp by pointing a browser to `http://<hostname>:<port>/RTLSelfService` where `<hostname>` is the machine on which RTL application is running. The browser is redirected to the authentication page (see Figure 5-6).

Figure 5-6 Authentication Page



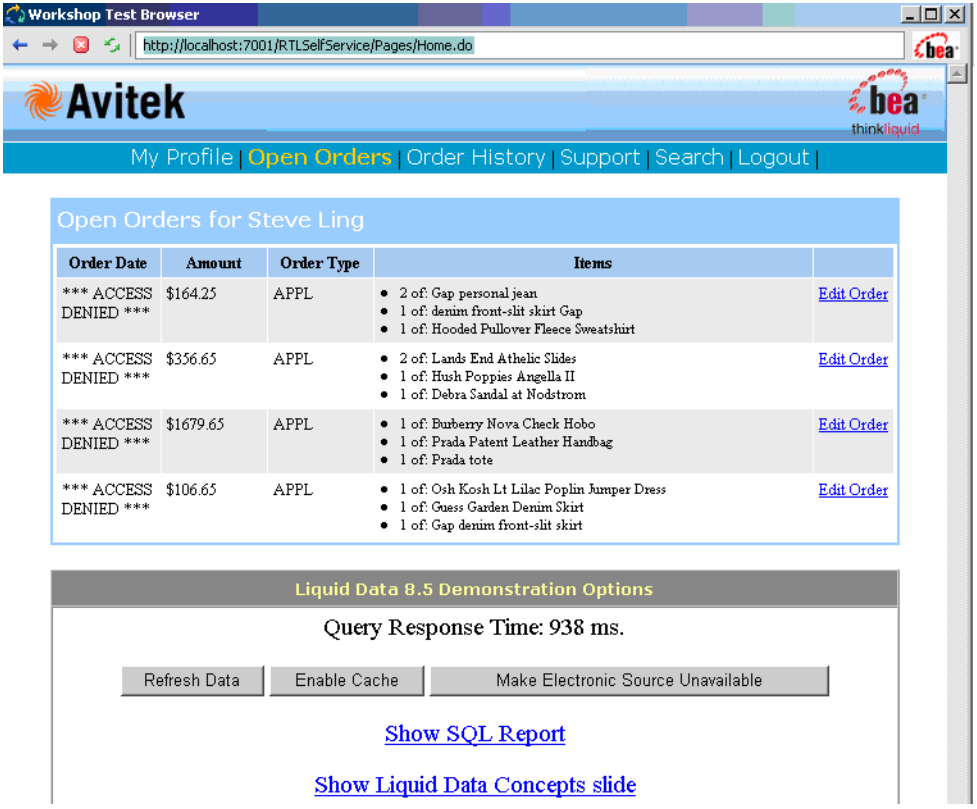
3. Log in as Steve and access should be granted to the Profile Page (see Figure 5-7).

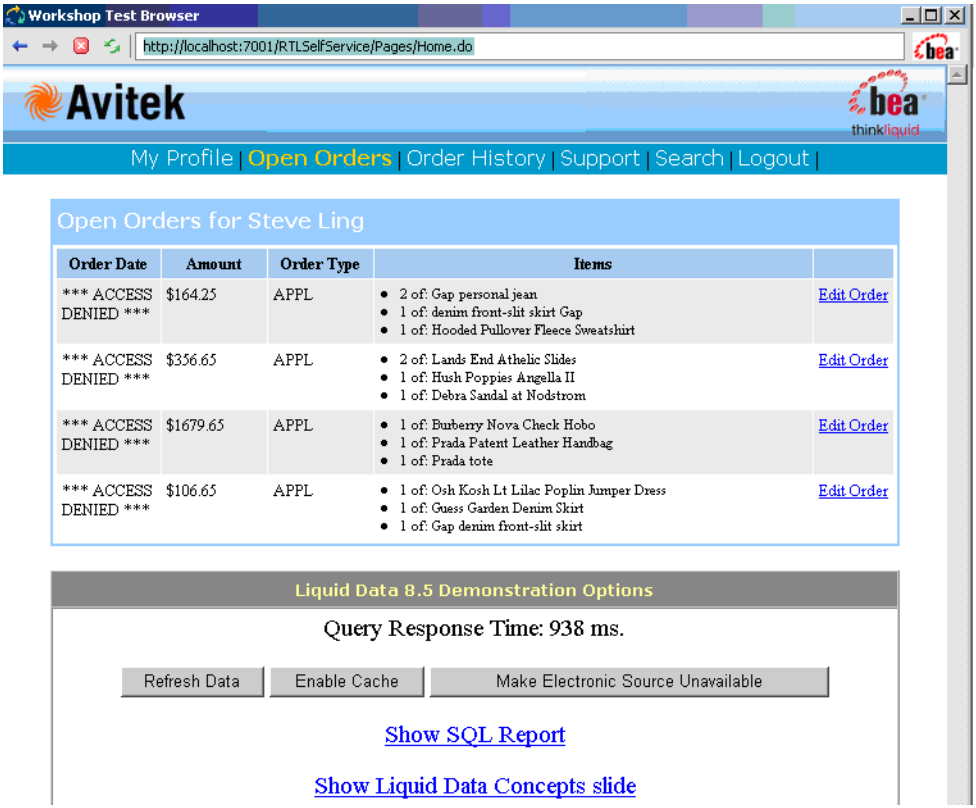
Figure 5-7 Profile Page



4. Select **Open Orders Page**. Open orders should be visible (see Figure 5-8), while access to Order Data should be denied.

Figure 5-8 Open Orders Page





Pre-Processing Data Redaction

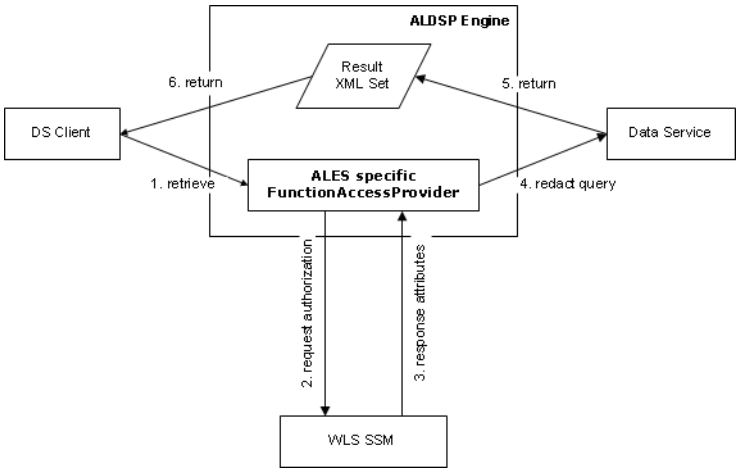
Pre-processing data redaction involves modifying the client request in some way before ALDSP forwards the request to the data service. This is achieved through the use of an ALES plug-in that calls the Java API to authorize the user request, gets the response attributes from the authorization response,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

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

1. ALDSP receives a data service client request and invokes the ALES plug-in.
2. ALES plug-in calls the ALES Java API for authorization. The authorization decision returns additional predicates as responses.
3. ALES plug-in returns the authorization decision and responses to ALDSP.

4. ALDSP adds the predicates and/or function name and calls the data service
5. ALDSP engine receives the results from the data service and returns it to the client.

Figure 5-9 Overview of Pre-Processing Solution



Pre-Processing Response Types

ALES supports two types of responses that return information to ALDSP 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 ALDSP 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, ALES 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 ALES plug-in (`com.bea.security.ales.aldsp.ALESFunctionAccessProvider`) calls the Java SSM to do authorization and return response attributes to ALDSP. For example, consider the following policy:

```
grant (
//priv/ALDSP_QUERY,
//app/policy/RTLApp/ld/DataServices/RTLServices/OrderView.ds/getOrders,
//user/asi/system/

) if report_as(“aldsp_replacement_function”,
“getOrdersThatAmountLessThan1000”)
```

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`. ALDSP then calls that replacement function (instead of the original). This function must the same signature; no additional verifications are performed at runtime.

Required ALES Response Attributes

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

ALES response attribute names are all lower case.

- **aldsp_replacement_function**

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

  ```
  report_as("aldsp_replacement_function",
  "ld:DataServices/RTLServices/OrderView:getOpenOrdersByCustID")
  ```

- **aldsp_xquery_expression**

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

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

  The attribute value can be a list. If so, ALDSP 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

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

1. Navigate to the WLS SSM instance directory and open `set-wls-env.bat` 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,//RTLApp/DataServices/RTLServices/OrderView.ds/
getOrders,//role/Admin/)if report_as(“aldsp_replacement_function”,
“getOrdersThatAmountLessThan1000”)
```

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”)
```

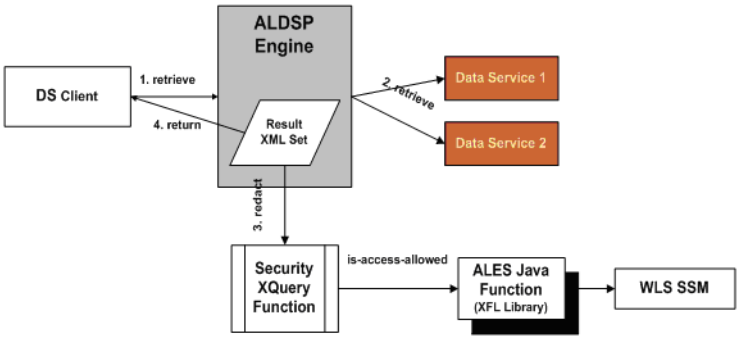
Post-Processing Data Redaction

Post-processing data redaction is achieved by invoking a security XQuery function after the ALDSP 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 ALDSP.
2. ALDSP engine retrieves the data from the data service(s).
3. ALDSP invokes the relevant security XQuery function for the data element.
4. Security XQuery function invokes an ALES Java method, passing in the resource name and one or more attribute names/values.
5. ALES Java method invokes the WLS SSM and gets the authorization result and optional responses defined in the queries.
6. ALES Java method returns the authorization decision and a set of responses to the security XQuery function.
7. XQuery function may use the ALES-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 5-10 Overview of the Post-Processing Solution



ALDSP Security XQuery Functions

The ALDSP 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.

ALES provides two ALES 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 ALES Java methods can be used instead of, or in addition to, the following ALDSP 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 ALES 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 ALDSP console, because the console compiler does not access the custom functions used in it.

ALES Java Methods

The two ALES 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 ALDSP data element. This resource name must be specified in the ALES 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 ALES policy.

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

In addition, ALES 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 ALDSP

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 5-11:

Figure 5-11 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 ALES Java method via an XFL library in the current Workshop application, as described in "Integrating the ALES Java Methods" on page 5-27.

   The ALES 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 ALES policies protecting the resource. Use two string arrays for names and values. A detailed example is provided in "ALES Security XQuery Function (ALDSP 2.5)" on page 5-29.

   If ALES 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 ALES Java function.

   The ALES Java method is able to determine the authenticated subject to use for authorization, and you do not need to supply it.
5. Invoke the ALES Java Function with the resource name, attributes, and values.

Integrating the ALES Java Methods

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

After you have done this, then:

1. Configure and distribute the policy in ALES:
   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 ALES Java method in Step 3.d.
   c. Distribute the policy change.
2. Import the ALES 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 ALDSP application's `APP-INF/lib` directory.
   b. In the ALDSP 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 ALDSP 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 ALES 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 ALES. 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 "ALES Security XQuery Function (ALDSP 3.0)" on page 5-34.

4. In the ALDSP configuration file, specify the data service element to protected:

   a. Open the *<ALDSP_APPLICATION_NAME>* `LDConfig.xml` in *<ALDSP_DOMAIN>*`/liquiddata`.

   b. Under the element whose id is the data service name to be protected, add the element *<con:AdminResources>*, such as the following:

```
<con:AdminResources>
  <con:AdminResource>
    <con:xpath>SecuredElementName</con:xpath>
    <con:useTag>false</con:useTag>
  </con:AdminResource>

  <con:AdminResource>
    <con:xpath> SecuredElementName</con:xpath>
    <con:QueryRef>SecurityXQueryFunctionName</con:QueryRef>
    <con:useTag>false</con:useTag>
```

```
    </con:AdminResource>
</con:AdminResources>
```

SecuredElementName is the XPath of the secured data element and *SecurityXQueryFunctionName* is the custom security XQuery function name.

5. Redeploy the application in the Weblogic Server Administration Console:
   a. Log in to the Weblogic Server Administration Console.
   b. Expand the node Deployments|Applications, 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.

ALES Security XQuery Function (ALDSP 2.5)

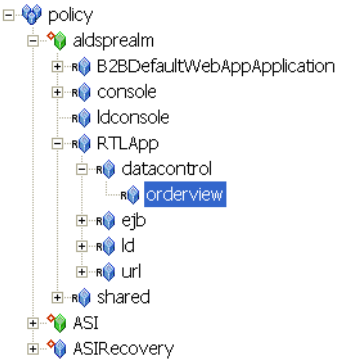
Note: An example for ALDSP 3.0 is provided in "ALES Security XQuery Function (ALDSP 3.0)" on page 5-34.

In this example, based on the RTLApp that is shipped by ALDSP 2.5, 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 ALDSP domain that is protected by the WLS SSM.

The integration example follows these steps:

1. Configure and distribute the policy in ALES:
   a. In the ALES Administration console, define resources named `datacontrol` and `orderview` under the `RTLApp` resource, as shown in Figure 5-12.

Figure 5-12 ALDSP Resource Tree



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

c. Define and distribute the following authorization policy:

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

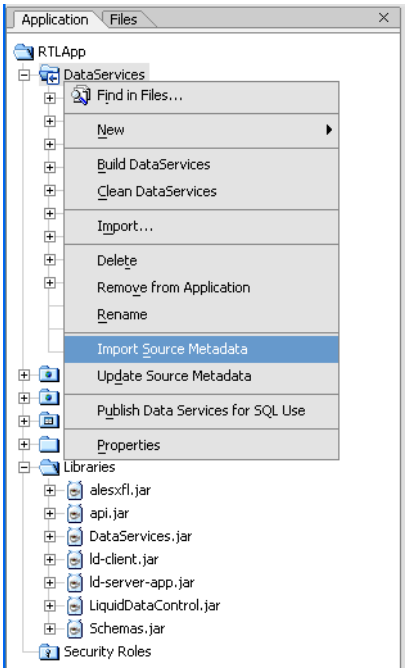
The privilege is `view`. The Subject is `LDSampleUsers`. The constraint is `totalorderamount < 1000`. Response attributes are returned via report_as("`class`", "A")”.

2. Import the ALES 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 ALDSP application’s `APP-INF/lib` directory.

b. In the ALDSP application, right-click the **DataServices** folder and select **Import Source Metadata** from the pop-up menu, as shown in Figure 5-13.

Figure 5-13 Import ALES Java Function



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, select the `is_access_allowed_with_response_attributes` method and finish the wizard.

3. Add a security XQuery function in the XFL file `library.xfl`, as shown in Figure 5-14. The following bullet points explain the function shown in the figure:

   – Line 22: Import the namespace of data service `OrderView`.

   – Line 24: Define a security XQuery function `secureOrders`.

   – Line 26: Invoke the ALES Java method. The first parameter is the resource name as defined in ALES. The second parameter is a string array that contains attribute names. In the example, there is only one attribute, named `totalorderamount`, which was defined in Step 1.c. The third parameter is a string array that contains attribute values.

– Line 28: The type of the element `TotalOrderAmount` is `xsd:decimal`. The function `fn:round()` converts the element into a integer. The function `fn:string()` converts the element into a string.

– Line 29: If the first element is `true`, it indicates that the current operation is permitted.

– Line 30: Find the response attribute `class`, which was defined in Step 1.d.

– Line 31, 31: If the response tabulate `class` is not found, return false.

– Line 33 to 46: Check if the response attribute `class` contains the value `A`.

Figure 5-14 Security XQuery Function

4. Specify the data service element to be protected in the ALDSP configuration file:

   a. Open `RTLAppLDConfig.xml` in *BEA_HOME*`/weblogic81/samples/domains/ldplatform/liquiddata`.

   b. Find the OrderView configuration item:

   ```
   <con:DSConfiguration id="ld:DataServices/RTLServices/OrderView.ds">
   ```

   c. Add the following XML element under the `<con:DSConfiguration>` element:

   ```
   <con:AdminResources>
       <con:AdminResource>
       <con:xpath>ORDER</con:xpath>
       <con:useTag>false</con:useTag>
   </con:AdminResource>
   ```

```
    <con:AdminResource>
       <con:xpath>ORDER</con:xpath>
       <con:QueryRef>{lib:DataServices/library}secureOrders
       </con:QueryRef>
       <con:useTag>false</con:useTag>
     </con:AdminResource>
  </con:AdminResources>
```

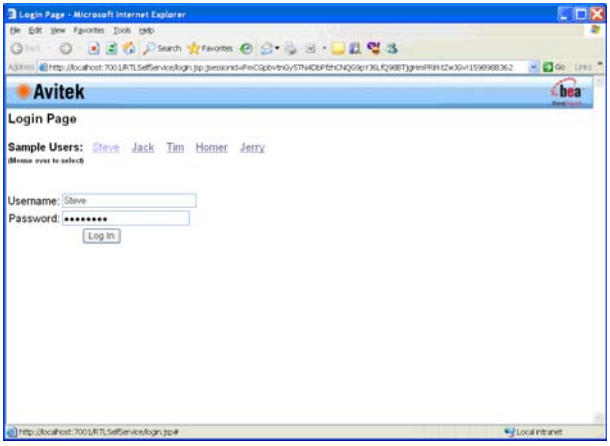
a. In the Weblogic Server Administration Console, expand the **Deployments|Applications** node and select the **RTLApp** application. In the right tab, select the **Deploy** tab and click the **Redeploy Application** button.

   When the Status of Last Action is `Success`, the application has been redeployed.

5. Restart the Weblogic Server.

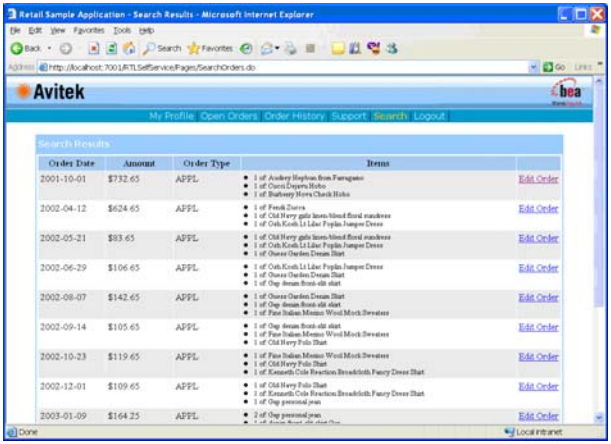
6. Open the RTLSelfService application (for example, http://localhost:7001/RTLSelfService) and select the user `Steve` to log in, as shown in Figure 5-15.

Figure 5-15 Avitek Login Page



7. Open the **Search** tab page and click the **Search Orders** button. Only those items whose attribute Amount is less than 1000 are displayed as shown in Figure 5-16.

Figure 5-16 Search Results with ALES Protection



ALES Security XQuery Function (ALDSP 3.0)

Note: An example for ALDSP 2.5 is provided in "ALES Security XQuery Function (ALDSP 2.5)" on page 5-29.

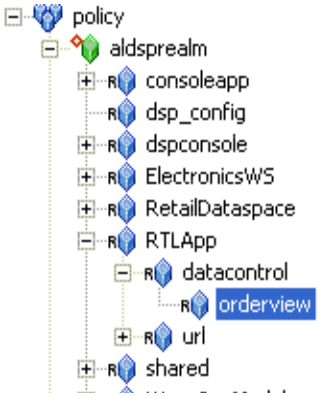
In this example, based on the RTLApp that is shipped by ALDSP 3.0, the data service `OrderView` is configured with a security XQuery function to protect its data elements. It is assumed that the application RTLApp has been deployed on an ALDSP domain that is protected by the WLS SSM.

The integration example follows these steps:

1. Configure and distribute the policy in ALES:

    a. In the ALES Administration console, define resources named `datacontrol` and `orderview` under the `RTLApp` resource, as shown in Figure 5-17.

Figure 5-17 ALDSP 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”);
```

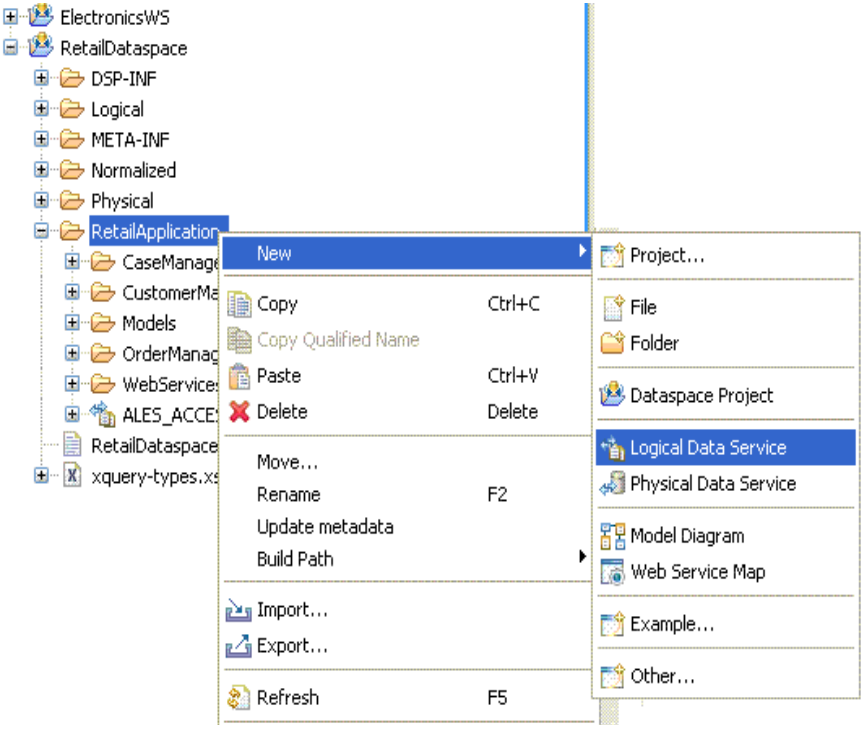
The privilege is view. The subject is `LDSampleUsers`. The constraint is `totalorderamount < 1000`. Response attributes are returned via report_as("class", "A")”.

2. Import the ALES 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 ALDSP application’s `APP-INF/lib` directory.

b. In the ALDSP 3.0 application, right-click **RetailApplication** folder and select **New>Logical Data Service**, as shown in Figure 5-18.

Figure 5-18 Import ALES Java Function



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, 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 5-19.

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

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

   – Line 37: invoke the ALES Java method. The first parameter is the resource name as defined in ALES. 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 5-28).

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

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

Figure 5-19 Security XQuery Function for ALDSP 3.0

ALES_ACCESS_CONTROL.ds

```
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 ALDSP 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 ALDSP console:

   a. Login ALDSP console and click **Lock & Edit**.

   b. Select **Security Configuration** on right bottom 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**.

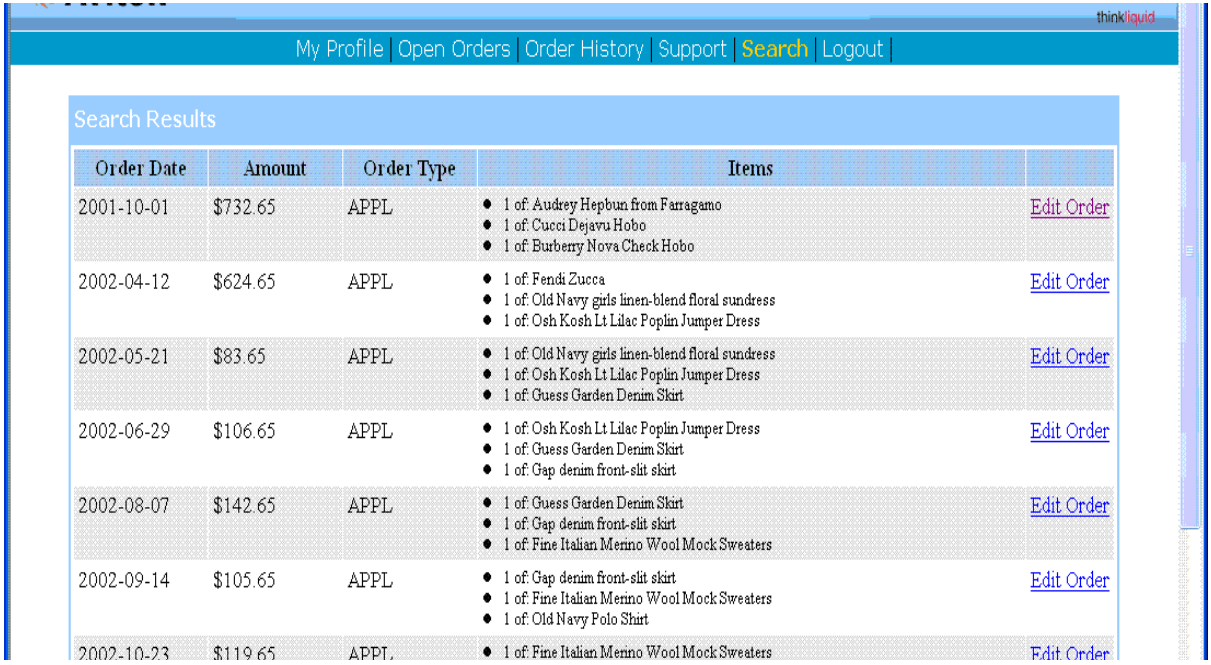
   e. On the **Secured Elements Configuration** tab, enter `ld:RetailApplication/ALES_ACCESS_CONTROL` for the namespace. Then enter `secureOrders` for local name and click **Add**.

   f. Click **Active 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 those items amounting to less than $1000.00 are displayed as shown in Figure 5-20.

Figure 5-20 Search Results with ALES Protection



CHAPTER 6

Securing WebLogic Portal Applications

This section covers the following topics:

Overview

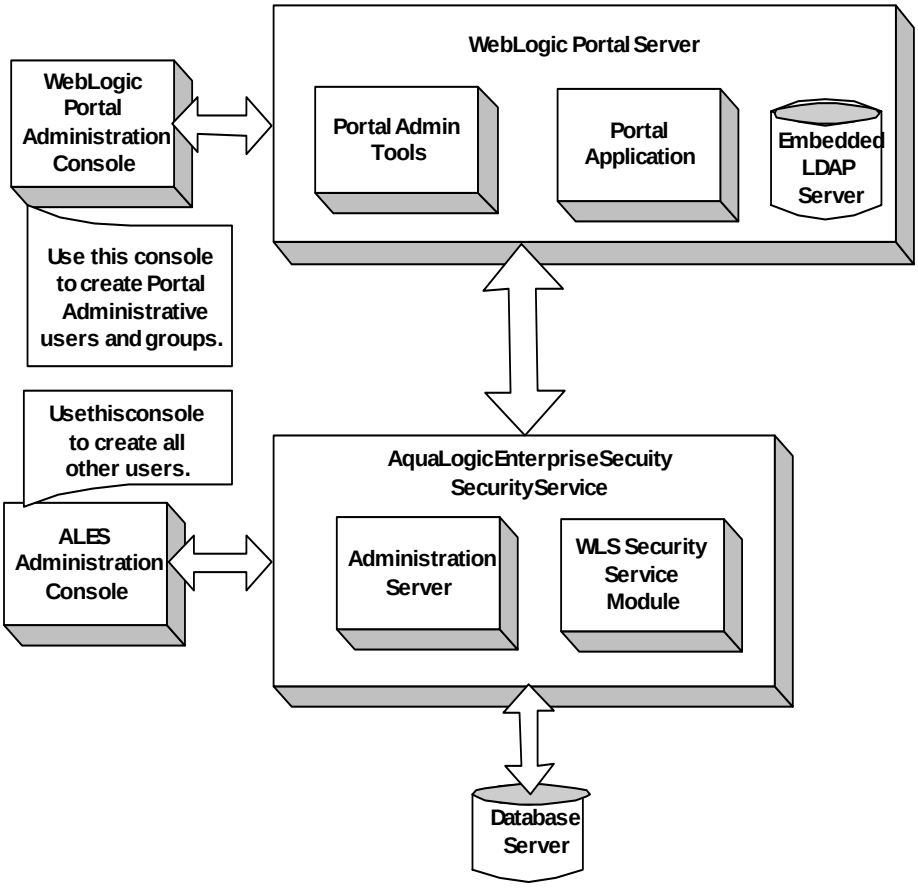
WebLogic Portal supports an open protection model and therefore requires a different policy model than the one used for other applications where access to a resource is denied unless a policy explicitly grants it. When integrated with ALES, the following access model is enforced:

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

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

ALES 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



Use-Case Scenario

The following use-case scenario is supported when you integrate ALES with WebLogic Portal:

- The ALES 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

When integrated with ALES, 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.

  ALES 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.
- ALES 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.

  Note: For information about installing and configuring the SSM, see the *SSM Installation and Configuration Guide*.

- Access to the ALES 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 ALES as described in "Define Portal Identities in ALES" on page 6-5.
3. Define the Portal resources in ALES as described in "Define Portal Resources in ALES" on page 6-6.
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, "Define Security Providers for WebLogic 9.x/10.0" on page 3-3.

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.
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 ALES

Note: To implement the use-case scenario described in "Use-Case Scenario" on page 6-3, you must use identities described in this section.

To create the Identity directory and users using the ALES 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 ALES

When defining Portal resources in ALES, keep in mind that ALES does not deny access to a portal resource if it is not defined in ALES *and* is accessible by anonymous access. In these cases, ALES returns *abstain*, instead of *true* or *false*. This is an important difference in the way that ALES 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 ALES 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 ALES.

Note: To implement the use-case scenario described in "Use-Case Scenario" on page 6-3, you must use the resources described in this section.

Realm Resource

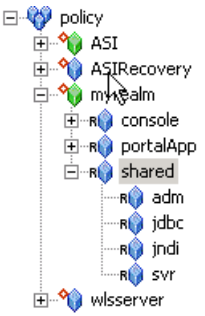
To create a realm resource, perform the following steps:

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



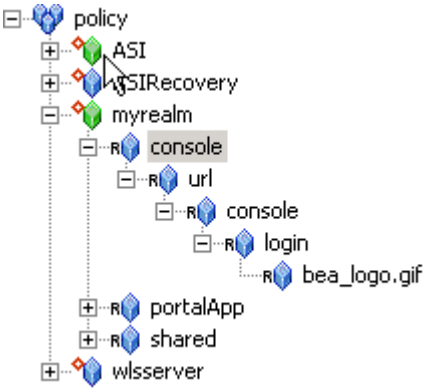
To create the `shared` resources, perform the following steps:

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**.

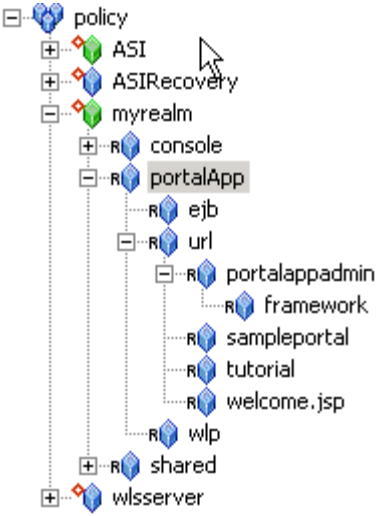
PortalApp Resources

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 6-3, 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. ALES 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 ALES in the following manner:

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

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 ALES 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 ALES 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 ALES 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 ALES 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 ALES, 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 ";`

CHAPTER 7

Storing and Versioning ALES Policy with ALER

This section describes how to integrate AquaLogic Enterprise Security with AquaLogic Enterprise Repository. It includes the following topics:

Overview

You can use AquaLogic Enterprise Repository to manage ALES policy data as ALER assets. By integrating ALER with ALES, you can:

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

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

Integration Tasks

To manage ALES policies with ALER:

1. Set ALER import/export properties as described in "Set ALER System Properties for Import and Export" on page 7-2.
2. Import the ALES Policy Asset Type into ALER, as described in "Import the ALES Policy Asset Type into ALER" on page 7-2.
3. Manage the ALES Policy Assets as described in "Manage ALES Policy Assets (ALER Console)" on page 7-4.

Set ALER System Properties for Import and Export

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

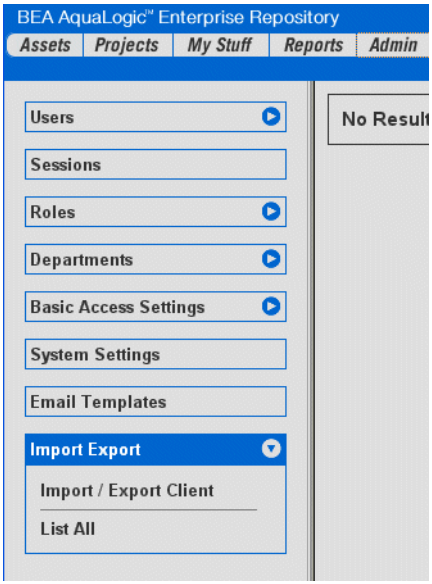
1. In the ALER console, open the **Admin** tab.
2. Select **System Settings** from the left column.
3. Under **Import / Export and Introspection** > **Import / Export**, set **Import/Export Client** `cmee.importexport.enabled` to **True**.
4. Under **Open API** > **Common**, set **Open API Enabled**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

Import the ALES Policy Asset Type into ALER

To manage ALES policies in ALER, the ALES Policy asset type must be imported in ALER. This asset type defines ALES metadata, such as privileges, policy, resources, and resource attributes. Follow these steps:

1. In the ALER console, open the **Admin** tab.
2. Select **Import Export** > **Import/Export Client**.

Figure 7-1 Starting the ALER Import/Export Client



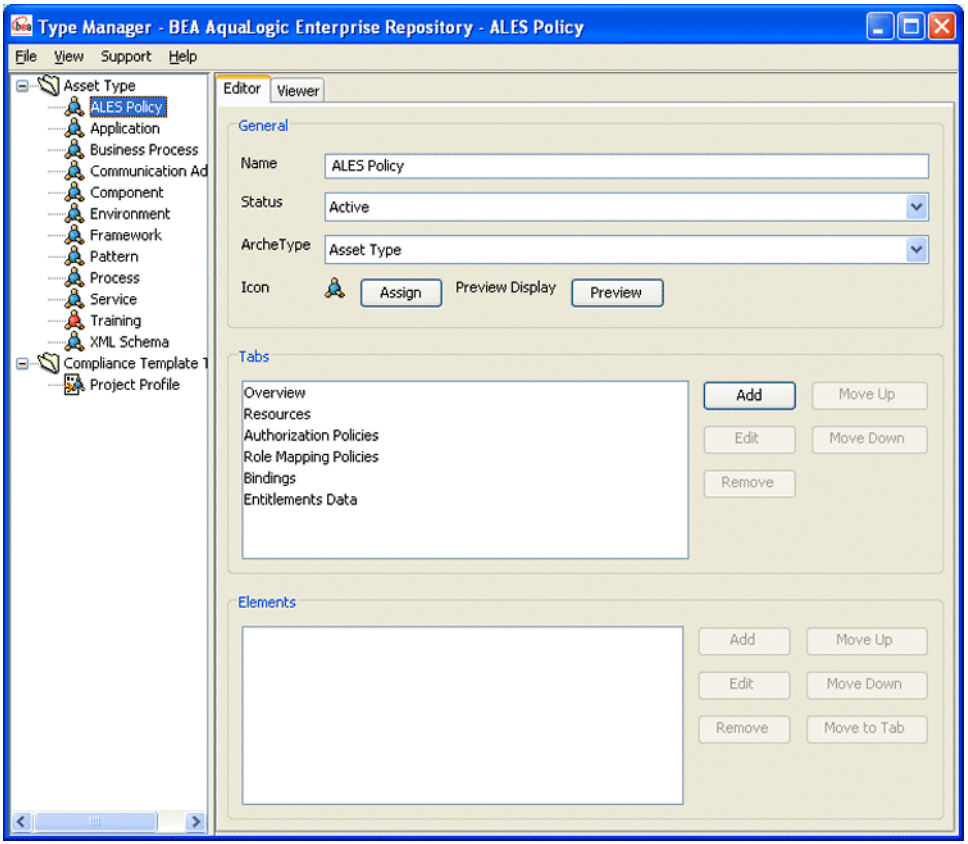
3. In the **Import** tab's **Select file to import** field, navigate to `ALES_ADMIN_HOME/data/aler` and select the appropriate ALES Policy Asset Type zip file. Then click **Next** twice.

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

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.

Figure 7-2 ALER Type Manager



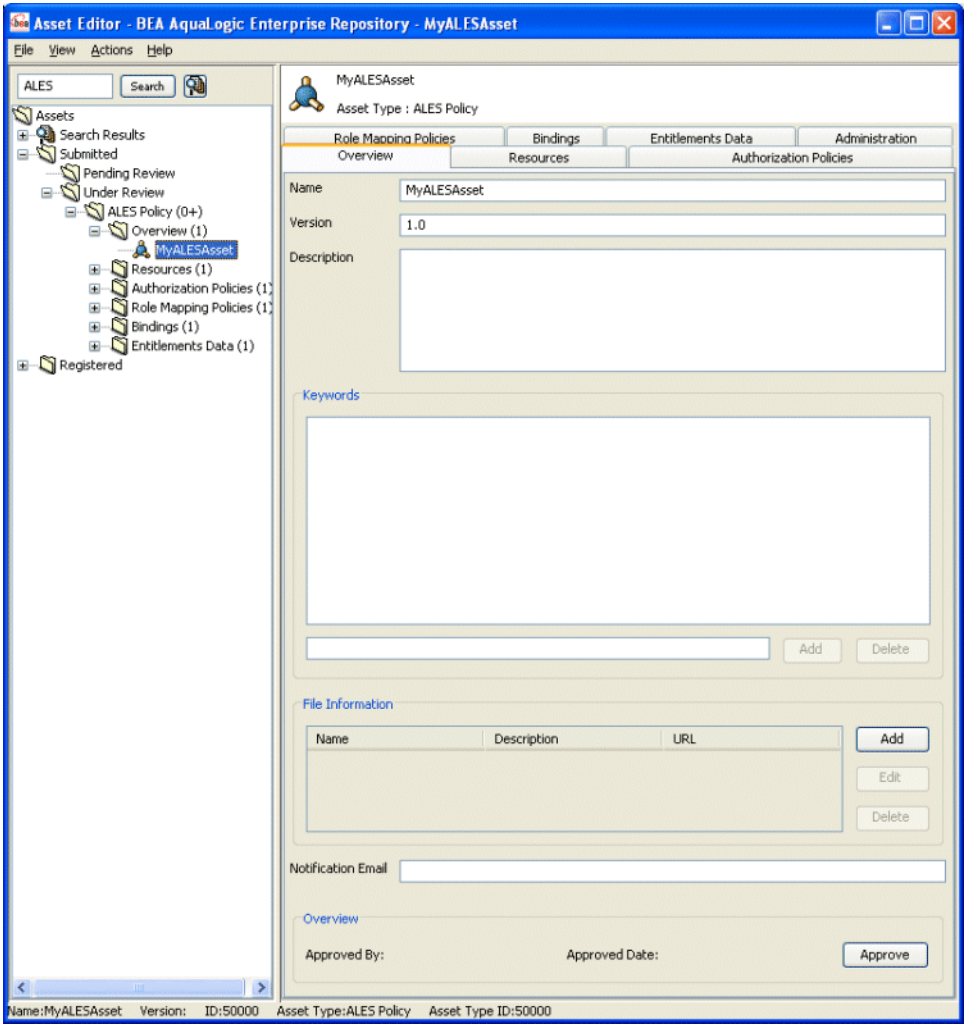
Manage ALES Policy Assets (ALER Console)

The ALER Asset Editor displays ALES Policy Assets in the following tabs (see Figure 7-3):

- 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, Privileges, and Privilege Groups.

- Role Mapping Policies—Used to manage role mapping 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—ALER administration information about the asset

Figure 7-3 ALES Policy Asset Editor Tabs



Versioning ALES Assets

ALER maintains version information for its assets. ALES Policy Assets use version numbers in the format N.N (1.0, for example). When importing ALES Policy Assets into ALER 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 ALER.

Importing/Exporting Policy Data Between ALES and ALER

The policyIX utility can perform direct import/export of policy asset data between ALES and ALER or it can generate files that can be used to import and export in separate steps.

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

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

Export from ALES to ALER

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

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

To export the data to ALER using a file:

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

Importing to ALES from ALER

To import directly into ALES from ALER, use the `-importFromALER` option:

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

To import using a file:

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

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

2. Import the data from the file:

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

Import/Export Configuration Files for ALER

This section describes the configuration files used for imports/exports between ALES and ALER.

Note: Further information about configuration files used with PolicyIX can be obtained by examining a sample file (*`ALES_ADMIN_HOME`*`/config/policyIX_config.xml`) or by consulting PolicyIX: config.xml in the *ALES Administration Reference* guide.

The configuration file uses XML syntax to specify required ALER information.

<aler_configuration>
A parent element containing all required <`aler_property`> elements.

<aler_property>

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

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

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

Listing 7-1 shows the contents of an example file:

Listing 7-1 Configuration for ALER Import and Export

```
<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 AquaLogic Service Bus Runtime Resources

This section covers the following topics:

Overview

AquaLogic Service Bus 2.5 (ALSB) is a configuration-based, policy-driven Enterprise Service Bus. It facilitates a loosely coupled architecture, facilitates enterprise-wide reuse of services, and centralizes management. AquaLogic Enterprise Security can be used to manage access control to ALSB's runtime resources, using the WLS SSM.

ALES secures only the runtime resources of ALSB, in general those resources that ALSB passes to `isAccessAllowed()`. It does not secure the resources used during ALSB configuration, such as the ALSB console.

Prerequisites

This document assumes the following:

- Installation of WebLogic Server 9.1/9.2 and AquaLogic Service Bus 2.5
- Installation of the ALES Administration Server.
- Installation and configuration of the WLS SSM on the ALSB machine.

Integration Tasks

The integration tasks are.

1. Define the security providers described in "Define the Security Providers" on page 8-2
2. Define ALSB resources as described in "Define ALSB Resources in ALES" on page 8-3
3. Define the Identities as described in "Define Identities" on page 8-6.
4. Define the policies as described in "Define Policies for ALSB" on page 8-6.
5. Distribute the policies as described in "Distribute Changes" on page 8-8.

Define the Security Providers

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

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

- ASI Authorizer
- ASI Role Mapping
- ASI Adjudicator

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`

Define ALSB Resources in ALES

Developing a set of policies typically begins by determining which resources you need to protect and your access control requirements. You then create the identity directory, resources, groups, users, and roles that you will use to write policies to protect those resources. Next you write a set of authorization and role mapping policies to define access control on those resources. Finally, you deploy the set of polices to the WebLogic Server Security Service Module that you use to control access to your data services.

Regular Resource

To create a regular resource named `abc`:

1. In the ALES 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 8-3.
2. Right-click the `xyz` resource and select **Configure Resource**.
3. Check the **Allow Virtual Resources** box and click **OK**.

ALSB Proxy Service Resources

Create resources in ALES corresponding to the ALSB Proxy Services. An ALSB 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`

ALES resource definitions for ALSB use this format:

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

Table 8-1 describes how ALSB Proxy Service reference elements map to ALES resource and privilege elements

Table 8-1 ALSB Proxy Service Elements Represented in ALES Resources and Privileges

Resource/Privilege Element	Description
binding app	The ALES binding node name.
Proxy Service app name	The default application name, `shared`.
ProxyService	The ALES resource type.
Folder name	The ALSB Proxy Service folder name.
`//priv/<operation>`	The operation field of the ALSB Proxy Service, representing one of the Web Services operations provided.

Here is an example of how to convert an ALSB transport level access control to an ALES policy. In ALSB:

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

is converted in ALES 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 ALSB 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 ALES 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 `ProxyServices` resources virtual.

Listing 8-1 Resource Tree

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

Discovering Services

When developing policies for use with a Security Service Module, 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*.

Define Identities

Follow these steps to create a user named `weblogic`.

1. In the ALES Administration Console, select **Identity** > **Users** and click **New**.
2. In the **Create User** window, enter the name `weblogic` and click **OK**.
3. Select the `weblogic` user and click **Set Password**. If this is a development environment, you can use the default password `weblogic`.

Define Policies for ALSB

The ALES Administration Server installation includes a set of sample polices for BEA AquaLogic Service Bus, located at `BEA_HOME/ales30-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 ALSB 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 8-6
- "Role Mapping Policies" on page 8-7

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 **Policy** > **Authorization Policies** and click **New**.

2. Check **grant** option the top of the window. Then select `any` from the list and click **Add**.
3. Click **Resources** tab and expand **myrealm** > **shared**.
4. Select `adm` and click **Add**, then select `svr` and click **Add**.
5. Click the **Policy Subjects** tab, click `Admin` and then click **Add**.
6. Make sure that the data is correct and click **OK**.
7. Repeat steps 2-6 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;
```

8. Repeat steps 2-6 to create a policy that grants all users access to the `ProxyServices` resource:

```
grant(access,
//app/policy/myrealm/shared/ProxyService/MortgageBroker/ProxyServices,
//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 **Policy** > **Authorization Policies** and click **New**.
2. In the **Available Roles** list, select **Admin** and click **Add**.
3. On the **Resources** tab, select `myrealm` in the A**vailable Resource** list and click **Add**.
4. On the **Policy Subjects** tab, select `Users` from the **Select Policy Subjects** dropdown menu. Then select `weblogic` and click **Add**.
5. Make sure that the data is correct and click **OK**.

6. Repeat steps 1-5 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;
   ```

7. Repeat steps 1-5 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;
   ```

Distribute Changes

After you have made changes to the configuration and policies in the ALES console, follow these steps to distribute the changes.

1. In the Administration Console's left pane, select **Click Deployment**.
2. Click **Configuration** in the right pane. Then select **Security Configurations** and click **Distribute Configuration Changes**. A message should indicate 100% distribution.
3. Select **Deployment** in the left pane.
4. Select **Policy** and click D**istribute Policy**. A message should indicate 100% distribution.

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

Verify SSM Configuration Using PerfDBAuditor

It is possible to use the ALES performance auditing provider to verify that the SSM has been properly configured to protect ALSB. This provider collects statistics about requests routed through ALES.

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 AquaLogic 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.

CHAPTER 9

Securing ALES Components

ALES is itself secured using the same policy model used to secure other applications. This chapter explains the default policies controlling administrative access to ALES. Information is provided in the following sections.

Note: Many tasks described in the document are performed using the ALES Administration Console. For more information about using the console, access the console's help system.

Default Objects

Installing ALES provides a number of objects that collectively define access to ALES components. This provides rudimentary security at startup; you can use the Administration Console to more completely define administrative access.

The default objects are listed below and are more fully described in sections that follow.

Table 9-1 Default Objects Defining Access to ALES

Object Type	Description
Resource	A representation of ALES components is defined in a separate tree under a root resource named ASI. Policies can be assigned to a resource representing an ALES component and thereby define access to that component. For more information, see "ALES Resources" on page 9-3.
Identity	A number of users, groups, and roles that reflect usage of ALES are provided. In particular, a user named `system` is set up as having complete administrative rights. For more information, see "ALES Identities" on page 9-16.
Role Mapping Policies	A number of role mapping policies are provided that assign some of the default roles to users/groups. For more information, see "Role Mapping Policies" on page 9-17.
Authorization Policies	A number of authorization policies are provided that assign privileges to roles/groups/users on specific resources in the ASI resource tree. For more information, see "Authorization Policies" on page 9-18.

Creating a New Admin User

By default, ALES provides a single administrative user identity named `system` having complete administrative rights. In a production environment, you should remove this administrative user and replace it with one or more other user identities. This section describes how to create a new administrative user named `myadmin`, replacing the `system` user:

1. In the ALES Administration Console, navigate to Identities > Users. Add a new user named `myadmin`.
2. Add the `myadmin` user to the Admin role.
3. Set a password for `myadmin` by selecting `myadmin` and clicking Edit > Set password.
4. Remove the `system` user from the Admin role.
5. Distribute policy
6. Stop the Administration Server.
7. Edit `BEA_HOME/ales30-admin/config/WLESWebLogic.conf` so that under "Java Additional Parameters" this line reads:

```
wrapper.java.additional.15=-Dwles.user.alias=myadmin
```

instead of

```
wrapper.java.additional.15=-Dwles.user.alias=system
```

8. Set the password for the `myadmin` user, using the `asipassword` utility. Execute:

   ```
   BEA_HOME/ales30-admin/bin/asipassword.bat myadmin ../ssl/password.xml
   ../ssl/password.key
   ```

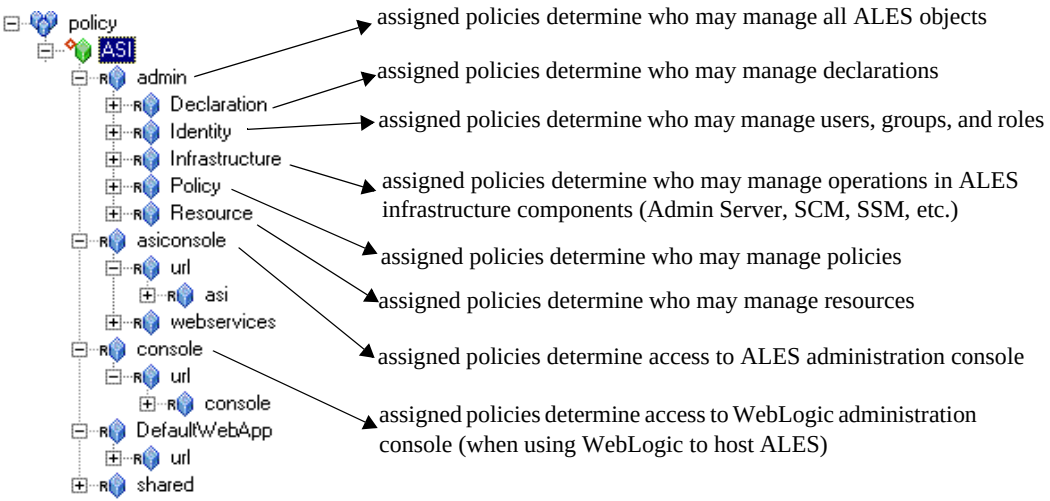
   and supply the password for `myadmin`.

9. Restart the ALES Administration Server with `WLESWebLogic.bat console`. You can now log in as `myadmin` with the password you set.

ALES Resources

ALES components are represented under the ASI resource tree, as shown in the figure below.

Figure 9-1 Representation of ALES Components



Administrative Operations

Table 9-2 describes resource objects that define the administrative operations that are performed using the Administration Console. By default, these resources are contained within `//app/policy/ASI/admin`.

Table 9-2 Resources Defining Administrative Operations

Resource Name	Protects operations on...
admin/Declaration/Attribute	attribute declarations
admin/Declaration/Constant	constant declarations
admin/Declaration/Enumeration	enumeration declarations
admin/Declaration/EvaluationFunction	evaluation function declarations
admin/Identity/Directory/Instance	identity directory instances
admin/Identity/Directory/AttributeMapping/Single	what scalar attributes may be assigned to users within a directory
admin/Identity/Directory/AttributeMapping/List	what vector attributes may be assigned to users within a directory
admin/Identity/Subject/User	users
admin/Identity/Subject/Group	groups
admin/Identity/Subject/Password	user passwords
admin/Identity/Subject/AttributeAssignment/Single	scalar subject attribute values
admin/Identity/Subject/AttributeAssignment/List	vector subject attribute values
admin/Resource/Instance	resources
admin/Resource/AttributeAssignment/Single	scalar resource attribute values
admin/Resource/AttributeAssignment/List	vector resource attribute values
admin/Resource/MetaData/LogicalName	setting the "logical name" resource metadata
admin/Resource/MetaData/IsApplication	setting the "is application" resource metadata
admin/Resource/MetaData/IsDistributionPoint	setting the "is distribution point" metadata
admin/Policy/Grant	grant policies
admin/Policy/Deny	deny policies
admin/Policy/Delegate	delegate policies
admin/Policy/Action/Role/Instance	roles (when used as actions)
admin/Policy/Action/Privilege/Instance	privileges
admin/Policy/Action/Privilege/Group	privilege groups
admin/Policy/Analysis/InquiryQuery	policy inquiries
admin/Policy/Analysis/VerificationQuery	policy verification
admin/Infrastructure/Engines/ARME	definitions of the ASI Authorizer, which is also called ARME
admin/Infrastructure/Engines/SCM	definitions of the Service Control Manager (SCM)
admin/Infrastructure/Management/BulkManager	the policy loader
admin/Policy/Repository	the policy repository

Privileges

Table 9-3 lists and describes the default privileges that may be assigned.

Table 9-3 Privileges

Privilege	Explanation
`create`	Create a policy element, including identities (identity directories, users, groups, attributes), resources and their attributes, configuration data and their bindings, and privileges and privilege groups.
`view`	View the contents of a policy element, including identities (identity directories, users, groups, identity attributes), resources and their attributes, configuration data and their bindings, privileges and privilege groups.
`delete`	Delete a policy element, including identities (identity directories, users, groups, identity attributes), resources and their attributes, configuration data and their bindings, and privileges and privilege groups.
`cascadeDelete`	Delete an element and its sub-elements (no permission check is made on sub-elements), including identities (identity directories, users, groups, identity attributes), resources and their attributes, configuration data and their bindings, and privileges and privilege groups.
`rename`	Rename a policy element, including identities (identity directories, users, groups, identity attributes), resources and their attributes, configuration data and their bindings, and privileges and privilege groups.
`modify`	Modify the contents of a policy element, including identities (identity directories, users, groups, identity attributes), resources and their attributes, configuration data and their bindings, and privileges and privilege groups.
`listAll`	Filter lists of instances based on a pattern specification.
`addMember`	Add a member to a group.
`removeMember`	Remove a member from a group.
`execute`	Execute a policy analysis query.
`deployUpdate`	Deploy a policy update.
`deployStructuralChange`	Deploy a structural change.
`bind`	Bind a resource to an ASI Authorization and ASI Role Mapping provider.
`unbind`	Unbind a resource from an ASI Authorization and ASI Role Mapping provider.

Table 9-3 Privileges (Continued)

Privilege	Explanation
`login`	Log on to the Administration Application, including the Administration Console, and the Policy Import and Export tools.
`copy`	Copy a policy element, including identities (identity directories, users, groups, identity attributes), resources and their attributes, configuration data and their bindings, and privileges and privilege groups.

Context Attributes

Context attributes can be used to provide fine-grained protection of policy operations. For example, when creating a privilege, the name of the privilege can be supplied as an attribute and used to control access to a single unique privilege.

Table 9-4 describes the default context attributes.

Table 9-4 Context Attributes

Attribute Name	Data Type	Description
declaration	string	Name of a declaration.
data_type	string	The name of a data type, for example, a string, integer, date.
attribute_usage_type	Enumeration (resource_attribute, subject_attribute, dynamic_attribute)	Specifies the type of policy element with which an attribute declaration is associated.
new_name	string	Generic attribute used when renaming elements.
new_attribute_usage_type	Enumeration (resource_attribute, subject_attribute, dynamic_attribute)	The new value for this item used to modify operations.
value	string	Generic attribute used to represent the value of an element.
values	list of strings	Generic attribute used to represent the value of an element as a list.
directory	string	The name of a directory.
attribute	string	The name of an attribute.
default_value	string	The default value of an attribute.
default_values	list of strings	The default value of a list attribute.

Table 9-4 Context Attributes (Continued)

Attribute Name	Data Type	Description
new_default_value	string	Used in modification operations to represent the new default value of an attribute value.
new_default_values	list of strings	Used in modification operations to represent the new default value of a list attribute.
subject_name	string	The name of a subject.
subjects	list of strings	A list of subjects.
groups	list of strings	The group membership of the subject.
subject_type	Enumeration (user_subject, group_subject, role_subject)	The type of subject.
member_subject_type	Enumeration (user_subject, group_subject, role_subject)	The type of the subject group member.
member_subject	string	Name of subject group member.
action	string	Name of the action.
action_type	Enumeration (privilege_action, role_action)	Type of the action.
resource	string	The name of the resource.
resources	list of strings	A list of resources.
constraint	string	The constraint of a policy; this is the portion between the ‘if’ and ‘;’ exclusive.
new_action	string	Name of new action in a modified policy.
new_action_type	string	New action type in a modified policy.
new_resource	string	New resource in a modified policy.
new_subject_name	string	New subject name.
new_constraint	string	New constraint in a modified policy.
delegator	string	The name of the delegator in a policy.
new_delegator	string	New delegator in a modified policy.
actions	list of strings	A set of actions.
action_groups	list of strings	A list of privilege group names.
action_group	string	The name of a privilege group.
parent_resource	string	The parent of the resource.
meta_data	string	The name of the metadata item.
logical_name	string	The logical name of a resource.
deleted_directories	list of strings	A list of deleted directories.
deleted_engines	list of strings	A list of deleted engines.[1]

Table 9-4 Context Attributes (Continued)

Attribute Name	Data Type	Description
deployed_engines	list of strings	A list of deployed engines.
deleted_bindings	list of strings	A list of deleted engine binding node pairs.
deleted_applications	list of strings	A list of deleted applications.
engine	string	The name of an ARME or SCM cluster.
engine_bindings	list of strings	A list of bindable resources bound to the ARME or SCM.
owner	string	The owner of analysis query.
effect_type	Enumeration (grant_effect, deny_effect, delegate_effect)	The type of role mapping and authorization policy effect.
title	string	The title of a analysis query.

1. The term engine refers to an ASI Authorization provider and ASI Role Mapper provider that are configured to operate in conjunction with one another, also referred to as the ARME. This combination of providers are configured to manage your authorization and role mapping policies.

Evaluation Functions

The evaluation functions listed in Table 9-5 are provided for writing custom administration policies. They may be used in the constraint portion of policies to limit the applicability of the policy based on contextual information.

Table 9-5 Evaluation Functions

Function Name	Description
resource_is_child(c,p,[d])	Check if c a child of p. d is a Boolean standing for direct. By default, d is true, meaning check if c is directly a child of p. If false, then c may be a descendant of p at any depth.
subject_in_directory(s,d)	Check if subject s is in directory d. This does not guarantee that either s or d exists, only that based on the name one would be in the other.
subject_is_group(s) subject_is_user(s) subject_is_role(s)	Check if the subject of a user group or role.
action_is_privilege(a) action_is_role(a)	Check if the action is a privilege or role

Authorization Queries

Table 9-6 describes when contextual data is used to define administrative access. This data that may be referenced when writing policies to protect the administration console.

Table 9-6 Context Attributes and Administrative Access

Admin Resource	Privilege	Context attributes	Description
Declaration/Attribute	create	declaration	Queried when user attempts to create a new attribute declaration.
	delete	declaration	Queried when user attempts to delete an attribute declaration.
	rename	declaration, new_name	Queried when user attempts to rename an attribute declaration.
	modify	declaration	Queried when user attempts to modify an attribute declaration.
Declaration/ Constant	create	declaration, value	Queried when user attempts to create a new constant.
	delete	declaration, value	Queried when user attempts to delete a constant.
	rename	declaration, value, new_name	Queried when user attempts to rename a constant.
	modify	declaration, value, new_value	Queried when user attempts to modify a constant.
Declaration/ Enumeration	create	declaration, value	Queried when user attempts to create a new enumeration.
	delete	declaration, value	Queried when user attempts to delete an enumeration.
	rename	declaration, value, new_name	Queried when user attempts to rename an enumeration.
	modify	declaration, value, new_value	Queried when user attempts to modify an enumeration.
Declaration/Evaluation Function	create	declaration	Queried when user attempts to create an evaluation function.
	delete	declaration	Queried when user attempts to delete an evaluation function.
	rename	declaration, new_name	Queried when user attempts to rename an evaluation function.

Table 9-6 Context Attributes and Administrative Access (Continued)

Admin Resource	Privilege	Context attributes	Description
Identity/Directory/Instance	create	directory	Queried when user attempts to create a directory.
	delete	directory	Queried when user attempts to delete a directory.
	cascade Delete	directory	Queried when user attempts to delete a directory and all its users.
	rename	directory, new_name	Queried when user attempts to rename a directory.
Identity/Directory/AttributeMapping/Single	create	attribute, default_value, directory	Queried when user attempts to add a scalar attribute to an attribute schema of a directory.
	delete	attribute, default_value, directory	Queried when user attempts to delete a scalar attribute from an attribute schema of a directory.
	modify	attribute, default_value, directory, new_default_value	Queried when user attempts to modify a scalar attribute in an attribute schema for a directory.
Identity/Directory/AttributeMapping/List	create	attribute, default_value, directory	Queried when user attempts to add a vector attribute to an attribute schema of a directory.
	delete	attribute, default_value directory	Queried when user attempts to delete a vector attribute from an attribute schema of a directory.
	modify	attribute, default_value, directory, new_default_value	Queried when user attempts to modify a vector attribute in an attribute schema of a directory.
Identity/Subject/User	create	subject_name	Queried when user attempts to create a new user.
	copy	subject_name, new_subject_name	Queried when user attempts to copy a user.
	delete	subject_name	Queried when user attempts to delete a user.
	cascade Delete	subject_name	Queried when user attempts to cascade a user and all policies associated with the user.
	rename	subject_name, new_subject_name	Queried when user attempts to rename a user.

Table 9-6 Context Attributes and Administrative Access (Continued)

Admin Resource	Privilege	Context attributes	Description
Identity/Subject/Group	create	subject_name	Queried when user attempts to create a new group.
	delete	subject_name	Queried when user attempts to delete a group.
	rename	subject_name, new_subject_name	Queried when user attempts to rename a group.
	addMember	subject_name, member_subject	Queried when user attempts to add a member to a group.
	remove Member	subject_name, member_subject	Queried when user attempts to remove a member from a group.
Identity/Subject/ AttributeAssignment/Single	create	attribute, value, subject_name	Queried when user attempts to set a value to a currently unset scalar subject attribute.
	delete	attribute, value, subject_name	Queried when user attempts to unset a currently set scalar subject attribute.
	modify	attribute, value, subject_name, new_value	Queried when user attempts to modify the value of a currently set scalar subject attribute.
Identity/Subject/ AttributeAssignment/List	create	attribute, value, subject_name	Queried when user attempts to set a value to a currently unset vector subject attribute.
	delete	attribute, value, subject_name	Queried when user attempts to unset a currently set vector subject attribute.
	modify	attribute, value, subject_name, new_value	Queried when user attempts to modify the value of a currently set vector subject attribute.
Identity/Subject/ Password	modify	subject_name	Queried when user attempts to modify the password for a user. The subject_name attribute contains the name of the user for which the password is associated.
Resource/Instance	create	resource, resource_type	Queried when user attempts to create a new resource.
	delete	resource	Queried when user attempts to delete a resource.
	cascade Delete	resource	Queried when user attempts to cascade delete a resource. This includes deletion of all child resources and associated policies.
	rename	resource, new_name	Queried when user attempts to rename a resource.

Table 9-6 Context Attributes and Administrative Access (Continued)

Admin Resource	Privilege	Context attributes	Description
Resource/Attribute Assignment/Single	create	attribute, resource, value	Queried when user attempts to set a value to a currently unset scalar resource attribute.
	delete	attribute, resource, value	Queried when user attempts to unset a currently set scalar resource attribute.
	modify	attribute, resource, value, new_value	Queried when user attempts to modify the value of a currently set scalar resource attribute.
Resource/Attribute Assignment/List	create	attribute, resource, value	Queried when user attempts to set a value to a currently unset vector resource attribute.
	delete	attribute, resource, value	Queried when user attempts to unset a currently set vector resource attribute.
	modify	attribute, resource, value, new_value	Queried when user attempts to modify the value of a currently set vector resource attribute.
Resource/MetaData/ IsApplication	modify	resource, value, new_value	Queried when user attempts to toggle the “is application” resource metadata.
Resource/MetaData/ IsDistributionPoint	modify	resource, value, new_value	Queried when user attempts to toggle the “is distribution point” resource metadata.
Resource/MetaData/ Logical Name	create	logical_name, resource	Queried when user attempts to create a logical name for a resource.
	delete	logical_name, resource	Queried when user attempts to delete a logical name for a resource.
	rename	logical_name, resource, new_name	Queried when user attempts to rename a logical name for a resource.
Policy/Grant	create	action, resource, subject_name, constraint	Queried when user attempts to create a new grant policy. “action”, “resource”, and “subject_name” attributes are lists.
	delete	action, resource, subject_name, constraint	Queried when user attempts to delete a grant policy. The “action”, “resource”, and “subject_name” attributes are lists.
	modify	action, resource, subject_name, constraint, new_action, new_resource, new_subject_name, new_constraint	Queried when user attempts to modify a grant policies “action”, “resource”, and “subject_name” attributes are lists.

Table 9-6 Context Attributes and Administrative Access (Continued)

Admin Resource	Privilege	Context attributes	Description
Policy/Deny	create	action, resource, subject_name, constraint	Queried when user attempts to create a new deny policy. “action”, “resource”, and “subject_name” attributes are lists.
	delete	action, resource, subject_name, constraint	Queried when user attempts to delete a deny policy. The “action”, “resource”, and “subject_name” attributes are lists.
	modify	action, action_type, resource, subject_name, subject_type, constraint, new_effect, new_action, new_action_type, new_resource, new_subject_name, new_subject_type, new_constraint	Queried when user attempts to modify a deny policy. The “action”, “resource”, and “subject_name” attributes are lists.
Policy/Delegate	create	action, resource, subject_name, delegator, constraint	Queried when user attempts to create a new delegate policy. “action”, “resource”, and “subject_name” attributes are lists.
	delete	action, resource, subject_name, delegator, constraint	Queried when user attempts to delete a delegate policy. The “action”, “resource”, and “subject_name” attributes are lists.
	modify	action, resource, subject_name, delegator, constraint, new_action, new_resource, new_subject_name, new_delegator, new_constraint	Queried when user attempts to modify a delegate policy. The “action”, “resource”, and “subject_name” attributes are lists.
Policy/Action/Role/ Instance	create	action	Queried when user attempts to create a new role.
	delete	action	Queried when user attempts to delete a role.
	rename	action, new_name	Queried when user attempts to rename a role.
Policy/Action/ Privilege/Instance	create	action	Queried when user attempts to create a privilege.
	delete	action	Queried when user attempts to delete a privilege.
	rename	action, new_name	Queried when user attempts to rename a privilege.

Table 9-6 Context Attributes and Administrative Access (Continued)

Admin Resource	Privilege	Context attributes	Description
Policy/Action/ Privilege/Group	create	action_group	Queried when user attempts to create a privilege group.
	delete	action_group	Queried when user attempts to delete a privilege group.
	rename	action_group, new_name	Queried when user attempts to rename a privilege group.
	addMember	action_group, action	Queried when user attempts to add a privilege to a privilege group.
	remove Member	action_group, action	Queried when user attempts to remove a privilege from a privilege group.
Policy/Analysis/ Inquiry Query	create	title, owner, effect_type, subjects, actions, resources, delegator	Queried when user attempts to create a new policy query.
	delete	title, owner	Queried when user attempts to delete a policy query.
	modify	title, owner, effect_type, subjects, actions, resources, delegator	Queried when user attempts to modify a policy query.
	execute	title, owner, effect_type, subjects, actions, resources, delegator	Queried when user attempts to execute a policy query. If this is an unsaved query “title” and “owner” will be set to an empty string.
Policy/Analysis/ Verification Query	create	title, owner, actions, resources	Queried when user attempts to create a new policy verification query.
	delete	title, owner	Queried when user attempts to delete a policy verification query.
	modify	title, owner, actions, resources	Queried when user attempts to modify a policy verification query.
	execute	title, owner, actions, resources	Queried when user attempts to execute a policy verification query. If this is an unsaved query “title” and “owner” will be set to an empty string.

Table 9-6 Context Attributes and Administrative Access (Continued)

Admin Resource	Privilege	Context attributes	Description
Policy/Repository	deploy Update	resource, directory	Queried when user attempts to deploy a policy update. "resource" is the distribution node and all nodes below it may be effected. This check is made for each chosen distribution point.
	deploy Structural Change	deleted_directories, deployed_engines, deleted_engines, deleted_bindings, deleted_applications	Queried when user attempts to deploy a structural change.
Infrastructure/Engines/ ARME	create	engine	Queried when user attempts to create a new Security Service Module.
	delete	engine	Queried when user attempts to delete a Security Service Module.
	rename	engine, new_name	Queried when user attempts to rename a Security Service Module.
	bind	engine, resource	Queried when user attempts to bind a resource to a Security Service Module.
	unbind	engine, resource	Queried when user attempts to unbind a resource from a Security Service Module.
Infrastructure/Engines/ SCM	create	engine	Queried when user attempts to create a Service Control Manager.
	delete	engine	Queried when user attempts to delete a Service Control Manager.
	rename	engine, new_name	Queried when user attempts to rename a Service Control Manager.
	bind	engine, resource	Queried when user attempts to bind a Security Service Module to a Service Control Manager. The "resource" contains the name of the Security Service Module.
	unbind	engine, resource	Queried when user attempts to unbind a Security Service Module from a Service Control Manager. The "resource" contains the name of the Security Service Module.

Table 9-6 Context Attributes and Administrative Access (Continued)

Admin Resource	Privilege	Context attributes	Description
Infrastructure/ Management/Console	login		Queried when user attempts to login to the Administration Console.
Infrastructure/ Management/BulkManager	login		Queried when user attempts to login to the Policy Import tool.

Enumerated Types

Table 9-7 lists the name of each enumerated type used in controlling administrative access.

Table 9-7 Enumerated Types

Name	Values	Description
attribute_usage_type_enum	(resource_attribute, subject_attribute, dynamic_attribute)	Specifies the valid usage for attributes.
subject_type_enum	(user_subject, group_subject, role_subject)	Specifies the valid subject types.
action_type_enum	(privilege_action, role_action)	Specifies the valid action types.
resource_type_enum	(organizational_node, binding_node, resource_node)	Specifies the valid resource types.
effect_type_enum	(grant_effect, deny_effect, delegate_effect)	Specifies the valid role mapping and authorization effect types.

ALES Identities

Table 9-8 shows the default ALES roles, users, and groups and some of their administrative rights as determined by existing policies.

Table 9-8 Default ProductNameShort Role Privileges and Identities

Role	Privileges / Resources	User/ Groups
Admin	Has all privileges, including creating and managing resources, identities, configurations, starting/stopping ALES servers, etc.	`System` (User)
Deployer	Privileges include modifying SCM/SSM configurations, deploying configuration and policy data, and running policy inquiries.	None

Table 9-8 Default ProductNameShort Role Privileges and Identities (Continued)

Role	Privileges / Resources	User/ Groups
Operator	Privileges include managing SCM/SSM configurations, starting /stopping Administration Server, and running policy inquiries.	None
Monitor	This role effectively provides read-only access to the Administration Console. Privileges include monitoring Administration Console activities and viewing SCM/SSM configurations.	None
Everyone	Change password, access the Console login page, access unprotected resources and operations	Allusers(Group)
Anonymous	No privileges. Does not allow access to ASI resources. This role is automatically assigned to all unauthenticated users.	Anonymous(User) Allusers(Group)

Role Mapping Policies

The default role mapping policies are described in Table 9-9 below. There are two ways they can be viewed in the Administration Console:

- To see role mapping policies assigned to a specific ALES resource, navigate to and select the resource in the ASI resource tree. Then click Role Mapping Policy Inquiry in the lower right page.
- To see role mapping policies assigned to a specific ALES role, expand the Identity node and select the Role node. Then select the role in the right page and click Role Mapping Policy Inquiry.
- To see all role mapping policies, expand the Policy node in the navigation tree and select Role Mapping Policies.

Of particular note, one of the role mapping policies assigns the Admin role to the user named System. This is the only administrative user provided when ALES is installed.

Table 9-9 Default Role Mapping Policies

Policy	Description
grant(//role/Everyone, //app/policy/ASI, //sgrp/asi/allusers/) if true;	Assigns Everyone role to allusers (group).
grant(//role/Admin, //app/policy/ASI, //user/asi/system/) if true;	Assigns Admin role to system (user).
grant(//role/Anonymous, //app/policy/ASI, //user/asi/anonymous/);	Assigns Anonymous role to anonymous (user)

Authorization Policies

A number of authorization policies are provided that define access to ALES components. Some of the more important default authorization policies are described in the table below.

Default Policy	Description
grant(//priv/delete, //app/policy/ASI/admin, //role/Admin) if true;	Allows Admin role to delete policies.
grant(//priv/cascadeDelete, //app/policy/ASI/admin, //role/Admin) if true;	Allows Admin role to perform cascadeDelete on children of ASI/admin.
grant(//priv/rename, //app/policy/ASI/admin, //role/Admin) if true;	Allows Admin role to rename children of ASI/admin.
grant(//priv/deployStructuralChange, //app/policy/ASI/admin/Policy/Repository, //role/Admin) if true;	Allows Admin role to deploy structural changes.
grant(//priv/login, //app/policy/ASI/admin/Infrastructure/Management/BulkManager, //role/Admin) if true;	Allows Admin role to use the policy loader tool.
grant(//priv/copy, //app/policy/ASI/admin/Identity/Subject/User, //role/Admin) if true;	Allows Admin role to copy users.
grant([//priv/bind,//priv/unbind], //app/policy/ASI/admin/Infrastructure/Engines, //role/Admin) if true;	Allows Admin role to bind/unbind resources, and configure authorization and role mapping provider combinations and SCMs.
grant(//priv/deployUpdate, //app/policy/ASI/admin/Policy/Repository, [//role/Admin,//role/Deployer]) if true;	Allows Admin and Deployer roles to deploy policy updates.
grant(//priv/modify, //app/policy/ASI/admin, [//role/Admin,//role/Deployer]) if true;	Allows Admin and Deployer roles to children of ASI/admin (resources, identities, policies, etc.)
grant(//priv/view, //app/policy/ASI/admin, [//role/Admin,//role/Monitor,//role/Operator,//role/Deployer]) if true;	Allows Admin, Monitor, Operator, and Deployer roles to view children of ASI/admin.
grant(//priv/listAll, //app/policy/ASI/admin, [//role/Admin,//role/Monitor,//role/Operator,//role/Deployer]) if true;	Allows Admin, Monitor, Operator, and Deployer roles to perform the listAll on children of ASI/admin.

Default Policy	Description
`grant(//priv/modify,` //app/policy/ASI/admin/Identity/Subject/ Password, //role/Everyone) if subject_name = sys_user_q;	Allows Everyone to modify their own password.

grant(//priv/create, [//app/policy/ASI/admin/Declaration, //app/policy/ASI/admin/Identity, //app/policy/ASI/admin/Infrastructure, //app/policy/ASI/admin/Resource], //role/Admin) if true;
grant(//priv/create, [//app/policy/ASI/admin/Policy/Action, //app/policy/ASI/admin/Policy/Analysis, //app/policy/ASI/admin/Policy/Rule/Delegate, //app/policy/ASI/admin/Policy/Rule/Grant], //role/Admin) if true;
	Allows Admin role to create policies.

grant([//priv/create,//priv/modify, //priv/view], //app/policy/ASI/admin/Policy/Analysis, [//role/Admin,//role/Monitor, //role/Operator,//role/Deployer]) if owner = sys_user_q;	Allows Admin, Monitor, Operator and Deployer roles to query ALES policies they own.
grant(//priv/execute, //app/policy/ASI/admin/Policy/Analysis, [//role/Admin,//role/Monitor,//role/Operator,//role/Deployer]) if owner = sys_user_q or owner = "";	Allows Admin, Monitor, Operator and Deployer roles to query both policies they own and policies with no owner.
grant([//priv/addMember,//priv/ removeMember], //app/policy/ASI/admin, [//role/Deployer]) if true;	Allows Deployer role to add and remove members to subject and privilege groups.

Setting Up Application Security Administrators

ALES allows you to set up application-level administrators who are responsible for managing the security for a specific application. An application-level administrator will be able to manage the policies protecting resources belonging to that application, but no others.

The basic procedure described here for setting up application-level administrators is to create a parent application resource that will contain the application resource to be secured and then define policies that will allow the administrators to manage those resources.

Establishing a Resource Parent for the Application

Using the Administration Console to create a resource that serves as the application resource parent involves the following steps:

1. Select the **Resource** node in the left pane.
2. In the right pane, right-click the `Policy` resource at the top of the tree and select select **Add Resource**.
3. Enter a resource name and select **Binding** in the **Type** field. Then click **OK**.
4. Right-click the new resource and select **Configure Resource**.
5. Select **Binding Application** in the **Type** field and click **OK**.

Policies for Application-Level Administration

Once the application parent resource is defined, you can define policies that apply to resources under the parent resource. Here are two examples:

Note: A comprehensive understanding of these policies can be obtained by examining the policies already in place for ALES components.

The following policy assigns the `Admin` role to `Joe` only for managing resources in the Petstore application.

```
grant(//role/Admin, //app/policy/ASI/admin/Resource, //user/asi/Joe/) if
resource_is_child(resource, //app/policy/Petstore, no);
```

Figure 9-2 Using the Resource_is_Child Constraint

Roles	Resources	Policy Subjects	Constraints
Admin	petstore	user/petstore/small	if resource_is_child (resource, //app/policy/petstore)

The following policy assigns `Bob` to the `Admin` role only for Petstore resources:

```
grant(//role/Admin, //app/policy/ASI/admin, //user/asi/Bob/) if
sys_defined(resource) and resource_is_child(resource,
//app/policy/Petstore, no);
```

APPENDIX A

ALES Adapter for Sun Identity Manager

The AquaLogic Enterprise Security 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 ALES.

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 ALES, and the user operations in the ALES Administration console will be synced into Sun Identity Manager. The sync interval from ALES to Sun Identity Manager is configurable.

Set Up ALES 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 `ales30-admin` to `idm/WEB-INF/lib`:
   - `ales30-admin/lib/asi_classes.jar`
   - `ales30-admin/lib/asitools.jar`
   - `ales30-admin/lib/jsafeJCE.jar (WLS 8.x) or jsafeJCEFIPS.jar (WLS 9.x)`
   - `ales30-admin/lib/log4j.jar`

- `ales30-admin/lib/ssladapter.jar`
- `ales30-admin/lib/sslplus.jar`
- `ales30-admin/lib/webservice.jar`
- `ales30-admin/lib/webserviceclient.jar`
- `ales30-admin/lib/providers/ojdbc14_g.jar`
- `ales30-admin/lib/providers/jconn2.jar`
- `ales30-admin/lib/providers/jconn3.jar`
- `ales30-admin/data/SunIMAdapter/lib/ALESResourceAdapter.jar`

3. Copy `ales30-admin/data/SunIMAdapter/forms/*` to `idm/sample/forms`.
4. Copy `ales30-admin/data/SunIMAdapter/images/ALES.gif` to `idm/applet/image`.
5. Add execute permission for the following scripts on UNIX platforms:
   - `ales30-admin/bin/install_user_change_schema_oracle.sh`
   - `ales30-admin/bin/install_user_change_schema_sybase.sh`
6. Run the following scripts to set up table space for the ALES `UserChangeDBAuditor`, which is configured in a subsequent step.

   For Oracle, run:

   `ales30-admin/bin/install_user_change_schema_oracle.bat|sh`

   For Sybase, run:

   `ales30-admin/bin/install_user_change_schema_sybase.bat|sh`

   You need to supply your ALES 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 ALES 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 ALES resource parameters as follows, and then click Test Configuration. Make sure that the ALES Administration servers are currently running.

       - Host: The host name or IP address of ALES admin server
       - TCP port: The port number for BLM server (default=7011)
       - Username: The user who has privilege to manager users in ALES, e.g. “system”
       - Password: The password of user manager of ALES admin
       - Directory of Keystore: The full path to the ssl dir in the ALES admin. If the IDM is not located on the same machine as ALES 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 ALES Resource parameters and make sure that the ALES 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 ALES Resource

An ALES Audit provider is used to record user-related operations in the ALES 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 ALES resource depends on whether you are using the WebLogic 9.x or WebLogic 8.1 SSM. When you use the WLS 9.x SSM, you configure security providers and other aspects of the SSM in the WebLogic Administration Console, rather than the ALES Administration Console.

Using the WebLogic 9.x SSM

1. Start the ALES 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 ALES 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 ALES configuration.
   i. Click Save.
5. Click Release Configuration on left top of page.
6. Restart the ALES servers to make the UserChangeDBAuditor take effect.

Using the Weblogic 8.1 SSM

1. Start the ALES Administration Server.
2. Log in to the ALES 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 ALES 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 ALES 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 ALES Resource:
   a. Click Resource at the top of the menu.
   b. Select the ALES 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 ALES 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 ALES resource to Target Resources.

i. On the Target Attribute Mappings page, you can use add and remove to set up the mapping between ALES attributes and Identity Manager attributes. After you have finished the attribute-mapping settings, click Save to finish.