BEA AquaLogic Enterprise Security™®

Integrating ALES with Application Environments

Version 2.6
Document Revised: April 2007

Copyright

Restricted Rights Legend

Trademarks and Service Marks

1. Introduction

2. Securing ALES Components

3. Setting Up Application Security Administrators

4. Integrating ALES with Applications

5. Configuring the Web Server SSM

6. Configuring the Web Services SSM

7. Configuring the WebLogic Server 8.1 SSM

8. Configuring the WebLogic Server 9.x SSM

9. Post-Installation Considerations for WLS 8.1 SSM and WLS 9.x SSM

10. Integrating with BEA Workshop for WebLogic Platform

11.Integrating with WebLogic Portal

12.Integrating with AquaLogic Data Services Platform

15.Enabling SAML-based Single Sign-On

16.Enabling SPNEGO-based Single Sign-on

CHAPTER 1

Introduction

This section describes the contents and organization of this guide—*Integrating ALES with Application Environments*. It includes the following topics:

Document Scope and Audience

This document is a resource for system administrators and database administrators who administer and deploy BEA AquaLogic Enterprise Security™. It is primarily intended for Application Security Administrators who are responsible for configuring the ALES components, integrating ALES into application environments, managing interaction between an applications and ALES, and setting up application-level security administrators.

The topics in this document are relevant during the staging, production deployment, and production use phases of a software project. For links to other AquaLogic Enterprise Security documentation and resources, see "Related Documentation" on page 1-3.

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 the principal components and architecture of BEA AquaLogic Enterprise Security. Read the *Introduction to BEA AquaLogic Enterprise Security* for

conceptual information that is helpful in understanding how the product works. This document provides information for post-installation configuration and operation of AquaLogic Enterprise Security; read *Installing the Administration Server* and *Installing Security Service Modules* for information about installation procedures that you need to perform first.

Additionally, BEA AquaLogic Enterprise Security includes many terms and concepts that you need to understand. These terms and concepts, which you will encounter throughout the documentation, are defined in the *Glossary*.

Guide to this Document

This document describes tasks associated with configuring and deploying AquaLogic Enterprise Security. After you have installed the ALES Administration Server (as described in *Installing the Administration Server*) and any ALES SSMs (as described in *Installing Security Service Modules*), you need to configure the SSMs to integrate them with the applications they secure. This document is organized as follows:

- Chapter 2, "Securing ALES Components," describes how to control access to ALES using the Administration Console.
- Chapter 3, "Setting Up Application Security Administrators," describes how to establish application-security administrator users using the Administration Console.
- Chapter 4, "Integrating ALES with Applications," describes how to configure SSMs and bind them to application servers.
- Chapter 5, "Configuring the Web Server SSM," describes how to configure the Web Server SSM.
- Chapter 6, "Configuring the Web Services SSM,"describes how to configure the Web Services SSM.
- Chapter 7, "Configuring the WebLogic Server 8.1 SSM,"
- Chapter 8, "Configuring the WebLogic Server 9.x SSM,"
- Chapter 9, "Post-Installation Considerations for WLS 8.1 SSM and WLS 9.x SSM,"
- Chapter 10, "Integrating with BEA Workshop for WebLogic Platform," describes how to integrate AquaLogic Enterprise Security with BEA Workshop for WebLogic by adding security-related annotations to your Java code and using the ALES Tag Library in your Java Servlet Pages (JSPs).

- Chapter 11, “Integrating with WebLogic Portal,”
- Chapter 12, “Integrating with AquaLogic Data Services Platform,”
- Chapter 13, “Integrating with AquaLogic Enterprise Repository,”
- Chapter 14, “Integrating with AquaLogic Service Bus,”
- Chapter 15, “Enabling SAML-based Single Sign-On,” describes how to configure SSMs to provide SSO using the SAML 1.1 Browser POST Profile.
- Chapter 16, “Enabling SPNEGO-based Single Sign-on,” describes the setup steps necessary to achieve Single Sign-On (SSO) integration with .NET based web services clients, as well as Internet Explorer browser clients.
- Chapter 17, “Authorization Caching,” describes how to configure and manage authorization caching to enhance performance when implementing authorization policies.

Related Documentation

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

- *Introduction to BEA AquaLogic Enterprise Security*—This document provides overview, conceptual, and architectural information for AquaLogic Enterprise Security.
- *Installing the Administration Server*—This document describes installing and configuring the AquaLogic Enterprise Security Administration Application.
- *Installing Security Service Modules*—This document describes installing and configuring Security Service Modules for AquaLogic Enterprise Security.
- *Administration and Deployment Guide*—This document provides an architectural overview of the product and includes step-by-step instructions on how to perform various post-installation administrative tasks.
- *Policy Managers Guide*—This document defines the policy model used by BEA AquaLogic Enterprise Security, and describes how to generate, import and export policy data.
- *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 for BEA AquaLogic Enterprise Security* —This document provides security vendors and security and application developers with the information needed to develop custom security providers.
- *Javadocs for Java API*—This document provides reference documentation for the Java Application Programming Interfaces that are provided with and supported by this release of BEA AquaLogic Enterprise Security.
- *Wsdldocs for Web Services API*—This document provides reference documentation for the Web Services Application Programming Interfaces that are provided with and supported by this release of BEA AquaLogic Enterprise Security.
- *Javadocs for Security Service Provider Interfaces*—This document provides reference documentation for the Security Service Provider Interfaces that are provided with and supported by this release of BEA AquaLogic Enterprise Security.
- *Javadocs for BLM API*—This document provides reference documentation for the Business Logic Manager (BLM) Application Programming Interfaces that are provided with and supported by this release of BEA AquaLogic Enterprise Security.

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 ALES Components

AquaLogic Enterprise Security is itself secured using the same policy model used to secure any other application. This chapter explains the default policies controlling administrative access to ALES.

Information is provided in the following sections.

Using the Administration Console

Many of the tasks described in the document are performed using the ALES Administration Console. For more information about using the Administration Console, see Using the Administration Console in the *Administration and Deployment Guide* and also consult the ALES Administration Console online help system.

Default Database Objects

Installing ALES provides a number of database 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 database objects are listed below and are more fully described in sections that follow.

Table 2-1 Default Database 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 2-3.
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 to the database. For more information, see "ALES Identities" on page 2-17.
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 "Default Role Mapping Policies" on page 2-17.
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 "Default Authorization Policies" on page 2-18.

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/ales26-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/ales26-admin/bin/asipassword.bat myadmin ../ssl/password.xml
   ../ssl/password.key
   ```

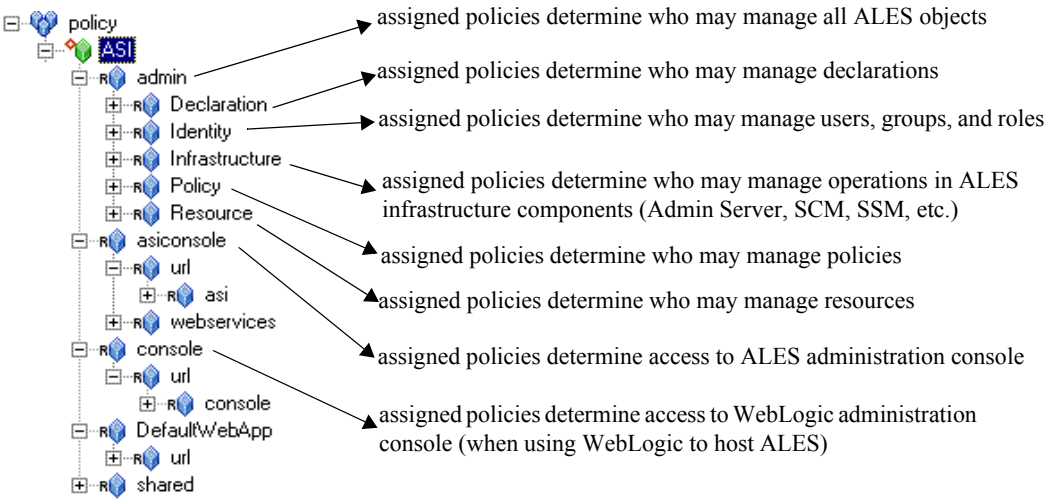
   and supply the password for `myadmin`.

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 2-1 Representation of ALES Components



Administrative Operations

Table 2-2 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 2-2 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

Table 2-2 Resources Defining Administrative Operations

Resource Name	Protects operations on...
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 2-3 lists and describes the default privileges that may be assigned.

Table 2-3 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.

Table 2-3 Privileges (Continued)

Privilege	Explanation
`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.
`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 2-4 describes the default context attributes.

Table 2-4 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.

Table 2-4 Context Attributes (Continued)

Attribute Name	Data Type	Description
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.
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.

Table 2-4 Context Attributes (Continued)

Attribute Name	Data Type	Description
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]
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 2-5 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 2-5 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.

Table 2-5 Evaluation Functions

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 2-6 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 2-6 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.

Table 2-6 Context Attributes and Administrative Access (Continued)

Admin Resource	Privilege	Context attributes	Description
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.
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.

Table 2-6 Context Attributes and Administrative Access (Continued)

Admin Resource	Privilege	Context attributes	Description
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.
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.
	addMembe r	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/S ingle	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/L ist	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.

Table 2-6 Context Attributes and Administrative Access (Continued)

Admin Resource	Privilege	Context attributes	Description
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.
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.

Table 2-6 Context Attributes and Administrative Access (Continued)

Admin Resource	Privilege	Context attributes	Description
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.
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.

Table 2-6 Context Attributes and Administrative Access (Continued)

Admin Resource	Privilege	Context attributes	Description
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.
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.
	addMembe r	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.

Table 2-6 Context Attributes and Administrative Access (Continued)

Admin Resource	Privilege	Context attributes	Description
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.
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.

Table 2-6 Context Attributes and Administrative Access (Continued)

Admin Resource	Privilege	Context attributes	Description
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.
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 2-7 lists the name of each enumerated type used in controlling administrative access.

Table 2-7 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 2-8 shows the default ALES roles, users, and groups and some of their administrative rights as determined by existing policies.

Table 2-8 Default ALES 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
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)

Default Role Mapping Policies

The default role mapping policies are described in Table 2-9 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 2-9 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)

Default 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 Table 2-10 below.

Table 2-10 Default Authorization Policies

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.

Table 2-10 Default Authorization Policies (Continued)

Default Policy	Description
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.
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.

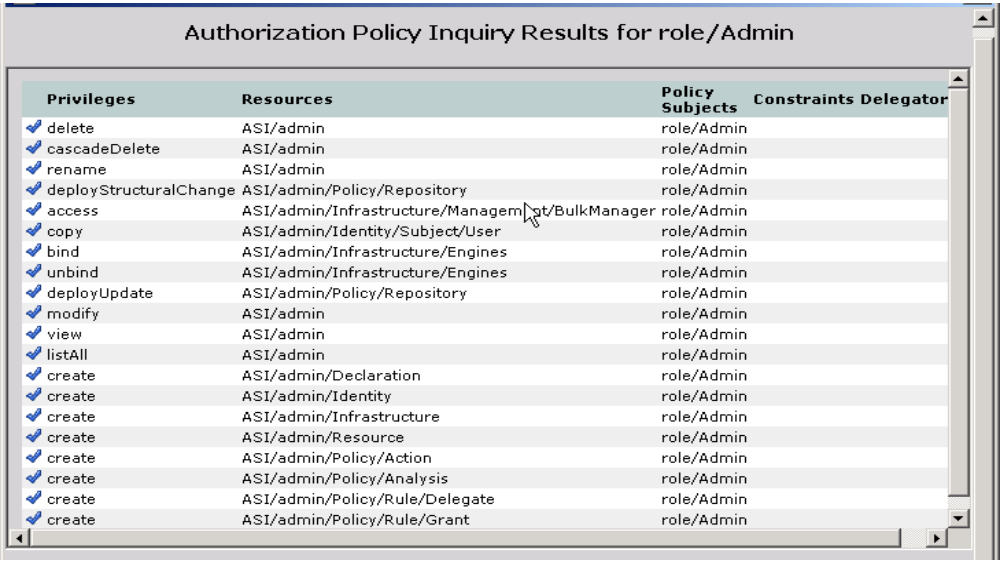
Viewing Authorization Policies

There several ways to view authorization policies in the Administration Console:

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

Figure 2-2 below shows the results of an authorization policies query on the Admin role.

Figure 2-2 Authorization Policy Inquiry Results Dialog



CHAPTER 3

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. The application-level administrator will be able to manage the policies protecting resources belonging to that application, but no others. This chapter describes some basic steps for establishing an application-level security administrators and provisioning them with an initial framework for protecting applications. This section provides information on the following topics:

Overview

Although the design of the administrative model will vary by use, it is presumed that the task of defining policies to secure an application will be assigned to application-level administrator who has complete rights only for the specific application.

The basic procedure described here for setting up application-level administrators is to create a parent application resource that will contain a representation of the application in the resource tree, create administrator user accounts and groups as needed, and then use policies that will allow the administrators to manage the application's security.

Establishing a Resource Parent for the Application

To represent an application in ALES, create a binding application resource to serve as the application parent. Then give the application security administrator the right to build resources under this parent.

To create a binding application resource for an application:

1. Select the Resource node in the navigation tree to display the current resource tree in the right panel of the Administration Console.
2. Right-click the top parent resource that will contain the application and 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.

Create Administrative Users

User accounts are needed for the application security administrators. If you want, you may create application-specific directories containing users and groups for the application.

Note: An implicit group named `allusers` is automatically added to all directories.

Identity Directories

To create a separate directory for an application's users and groups:

1. Select the Identity node in the navigation tree to display the current directories in the right panel of the Administration Console. After ALES is installed, there is one directory named ASI.
2. Click New in the lower right page.
3. On the Create Directory dialog, enter the directory name and click OK.

Users and Groups

To add a user or group to a directory:

1. Select the Identity node in the navigation tree to display the current directories in the right panel of the Administration Console.

2. Click the directory where you want to add the user or group, then select Edit Users or Edit Groups at the bottom of the page. This displays the directory's groups or users depending on your selection.
3. Select New at the bottom of the page.
4. On the dialog that displays, enter the user or group name and select OK.

Policies for Application-Level Administration

Once the application parent is defined in the resource tree and the necessary identities have been created, you can use policies to determine administrative access to the application. Here are two examples:

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

- Using policy constraints allows you to limit administrative rights. For example, the following policy assigns the Admin role to Joe only for managing resources for the Petstore application.

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

Figure 3-1 Using the Resource_is_Child Constraint

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

- Assign Admin role to Bob (user) for the purpose of performing inquiries on policies set on the Petstore application.

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

CHAPTER 4

Integrating ALES with Applications

This chapter provides information about ALES built-in support for integration with specific environments.

Overview

ALES provides a number of built-in solutions for integration with the following environments:

- Microsoft Internet Information Server
- Apache HTTP Server
- Web Services clients
- BEA WebLogic Server
- BEA WebLogic Portal
- AquaLogic Data Services Platform
- AquaLogic Service Bus

Each of these integrations is based on an ALES Security Service Module.

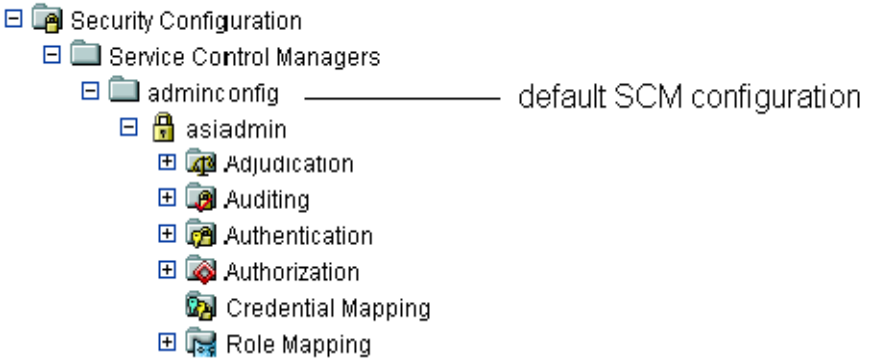
Security Service Modules

Before a SSM can be integrated with a server, a SSM configuration that specifies the security providers must be created and the configuration must be bound to the SCM running on the same machine.

As shown in Figure 4-1, installation of ALES creates a default SCM configuration named `adminconfig` that contains a SSM configuration and security providers used by the Administration Server itself.

If the SSM instance will be located on the same machine, you can use the SCM and create a SSM configuration under it. If on a separate machine, you must create a new SCM. For step-by-step instructions on managing SCM and SSM configurations, see the Administration Console help.

Figure 4-1 Default SCM



To create a SSM configuration:

1. Open the Security Configuration folder.
2. Select Unbound Configurations in the navigation tree and click on `Create a new Security Service Module Configuration` in the right page.
3. On the General tab, complete the following fields and click Create.

Table 4-1 SSM Configuration ID

Field	Description
Configuration ID	This entry must match the SSM configuration ID that is specified when the SSM instance is created on the server machine. The configuration ID is the means by which the SSM receives it configuration from the SCM.
Description	(Optional) A brief description of the SSM.

SSM Security Providers

The security providers needed depend on the requirements of the application. Installing a SSM deploys a JAR file that contains all ALES security providers. However, before any of the security providers can be used, you must use the Administration Console to configure them. You have the option of configuring either the security providers that ship with the product or custom security providers, which you may develop yourself or purchase from third-party security vendors. For more information on how to develop custom security providers, see *Developing Security Providers for BEA AquaLogic Enterprise Security*. For step-by-step instructions on managing providers, see the Administration Console help.

Note that the process of configuring security providers for the WebLogic Server 9.x SSM is different from that for other SSMs. For more information, see Chapter 8, "Configuring the WebLogic Server 9.x SSM."

Table 4-2 Authentication Providers

Provider	Description
WebLogic Authenticator	Authenticate users with WebLogic's embedded LDAP directory.
ALES Identity Asserter	Supports web server authentication and single sign-on between web server SSMs. Use this provider in conjunction with the ALES Credential Mapper.
Database Authenticator	Authenticates users using the ALES relational database provider.
Single Pass Negotiate Identity Asserter	Supports identity assertion using HTTP authentication tokens from the SPNEGO protocol. For more information, see Chapter 16, "Enabling SPNEGO-based Single Sign-on."
SAML Identity Asserter	Accepts SAML assertions sent using the Browser POST Profile and returns the corresponding user. For more information, see Chapter 15, "Enabling SAML-based Single Sign-On."

Table 4-2 Authentication Providers (Continued)

Provider	Description
Open LDAPAuthenticator	Authenticates users using an Open LDAP directory.
Active Directory Authenticator	Authenticates users using Active Directory.
NTAuthenticator	Authenticates users using Windows NT authentication.
iPlanet Authenticator	Authenticates users using an iPlanet LDAP directory.
Novell Authenticator	Authenticates users using a Novell LDAP directory.
X509 Identity Asserter	Supports identity assertion through an X.509 digital certificate, supporting ASN.1 encoding and decoding

Table 4-3 describes Authorization providers.

Table 4-3 Authorization Providers

Provider	Description
WebLogic Authorizer	Authorizes access to resources based on WebLogic security policy.
ASI Authorization Provider	Authorizes access to resources based on ALES security policy.

Table 4-4 describes Credential Mapping providers.

Table 4-4 Credential Mapping Providers

Provider	Description
Database Credential Mapper	Returns authentication credentials for a user (username and password) from a database.
SAML Credential Mapper	Returns a SAML assertion for an authenticated user. For more information, see Chapter 15, "Enabling SAML-based Single Sign-On."
ALES Identity Credential Mapper	Supports web server authentication and single sign-on between web server SSMs. Returns a ALES assertion for an authenticated user.
Weblogic Credential Mapper	Returns authentication credentials for a user (username and password) from the Weblogic LDAP directory.

Table 4-5 describes Role Mapping providers.

Table 4-5 Role Mapping Providers

Provider	Description
ASI Role Mapper	Returns a set or roles granted to a user on a protected resource based on ALES security policies.
Weblogic Role Mapper	Returns a set or roles granted to a user on a protected resource based on WebLogic security policies.

Integrating ALES with Other BEA Applications

ALES includes two SSMs for integrating with WebLogic Server and other BEA applications:

- WebLogic Server 8.1 SSM
- WebLogic Server 9.x SSM

The WebLogic Server SSMs integrate ALES with WebLogic Server and with BEA WebLogic Portal, AquaLogic Data Services Platform, and AquaLogic Service Bus. See the following chapters for more information about configuring ALES to work with those products:

- Chapter 7, “Configuring the WebLogic Server 8.1 SSM”
- Chapter 8, “Configuring the WebLogic Server 9.x SSM”
- Chapter 11, “Integrating with WebLogic Portal”
- Chapter 12, “Integrating with AquaLogic Data Services Platform”
- Chapter 13, “Integrating with AquaLogic Enterprise Repository”
- Chapter 14, “Integrating with AquaLogic Service Bus”

CHAPTER 5

Configuring the Web Server SSM

This section describes the Web Server Security Service Module, including tasks that you must perform after installing and completing the post-installation tasks. See *Installing Security Service Modules* for information about how to install the Web Server Security Service Module for Microsoft IIS or the Apache Web Server. The following topics are covered in this section:

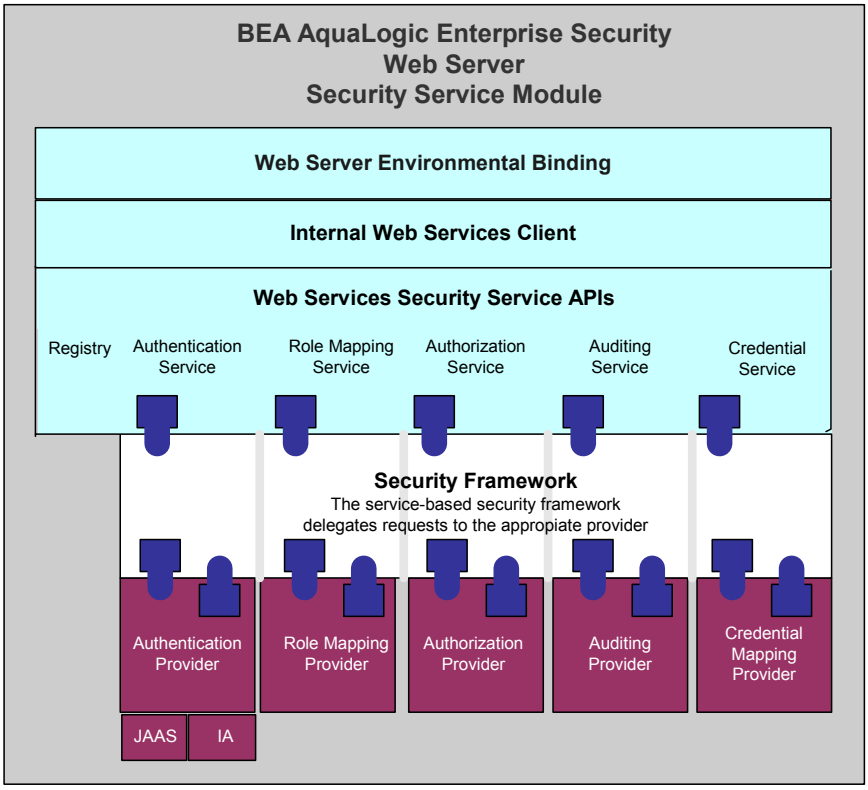
Understanding the Web Server SSMs

This section includes the following topics describing the Web Server SSMs:

Web Server SSM Overview

An ALES Web Server SSM provides the environmental bindings between the ALES and a web server. It can provide six distinct services: Registry, Authentication, Authorization, Auditing, Role Mapping, and Credential Mapping.

Figure 5-1 Web Server SSM Components



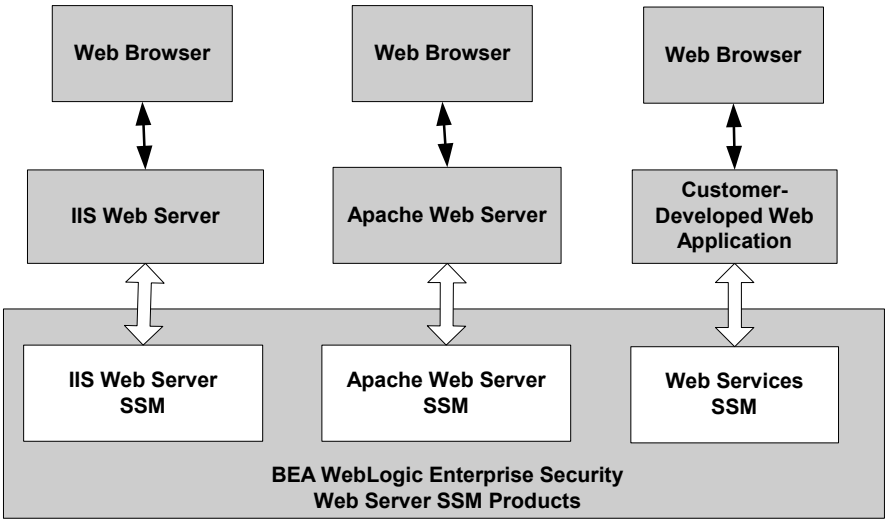
A Web Server SSM makes access decisions for the web server to which it is bound. The security configuration on which the access control decisions are based is defined and deployed by the Administration Server via the Security Control Module.

A Web Server SSM can be tailored to specific needs. Using templates provided as part of the product, security developers can customize the look and feel of authentication pages and

configure parameters that allow fine tuning for a particular installation. Web applications can have information added to the HTTP request by the security framework, such as roles and response attributes.

ALES provides three Web Server SSMs: IIS Web Server SSM (SSM), Apache Web Server SSM, and Web Services SSM (see Figure 5-2).

Figure 5-2 Web Server SSM Components



Web Server Environmental Binding

The environmental binding is used to bind to and interact with web servers. Binding a Web Server SSM to the server projects the ALES subsystem into the web server environment. The SSM accepts HTTPS requests from the web server and presents them to the ALES security framework.

Bindings are provided for two types of web servers: ASF Apache and Microsoft IIS. The second function is ultimately for enforcing access control and providing a means of implementing the SAML Browser/POST profile. Additionally, the Web Server SSM implements the server-side includes (SSIs) that process SAML Browser/POST profile.

Web Server SSM Features

This section describes the Web Server SSM features in the following sections

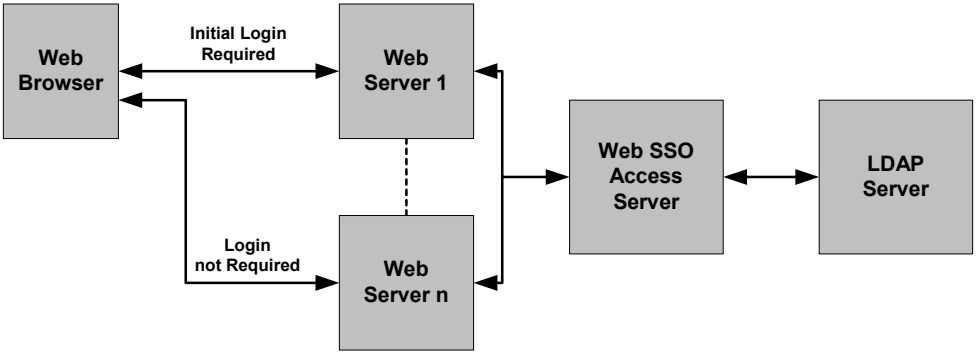
Web Single Sign-on Capabilities

This section covers the following topics:

What is Web 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. Figure 5-3 shows the basic components of a web single sign-on service.

Figure 5-3 Web Single Sign-on



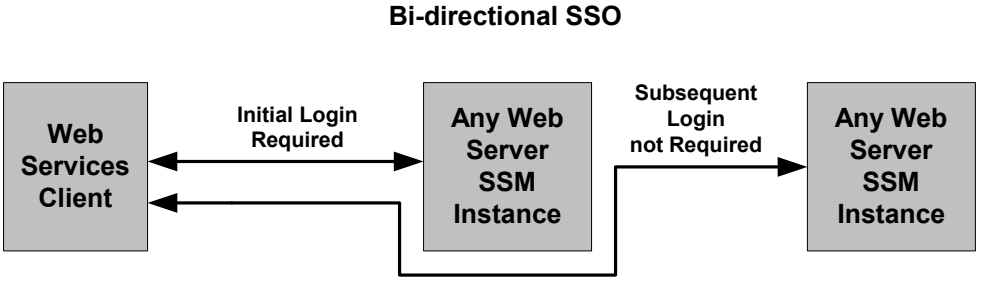
While web single sign-on facilitates access and ease of use, it does not improve security. In fact, security requirements should be considered when implementing a web single sign-on solution.

Single Sign-On Use Cases

The Web Server SSM supports the following single sign-on (SSO) use cases.

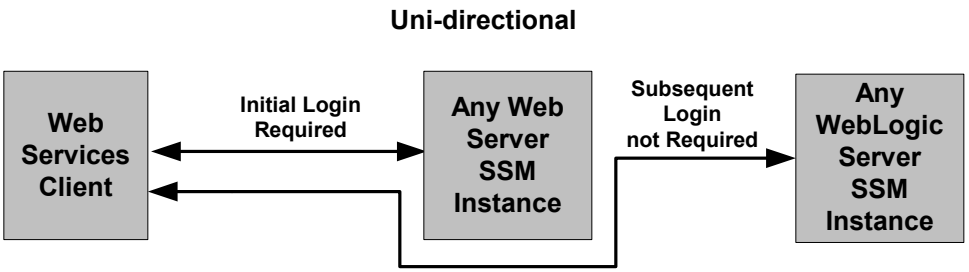
- **Bi-directional SSO among Web Server SSMs**—Once users are authenticated by a Web Server SSM, they are given an identity assertion token that they can use to access other Web Server SSM instances without being required to authenticate again (see Figure 5-4).

Figure 5-4 Web Server SSM to Web Server SSM Single Sign-on



- **Uni-directional SSO from Web Server SSM to WebLogic Server SSM instances**—Once users are authenticated by a Web Server SSM, they are given an identity assertion token that they can use to access WebLogic Server SSM instances without being required to authenticate again (see Figure 5-5).

Figure 5-5 Web Server SSM to WebLogic Server SSM Single Sign-On



Single Sign-On with ALES Identity Assertion

The Web Server and WebLogic Server SSMs support single sign-on using the ALES Identity Assertion provider. For instructions on how to implement Single Sign-On, see Chapter 16, "Enabling SPNEGO-based Single Sign-on."

Authentication Service Features

The authentication service supports the following features:

- **Conversion of JAAS callbacks to asynchronous**—Although the authentication service is JAAS based, the Web Server SSM masks the JAAS synchronous callback protocol from the web server. When form-based authentication is configured, the credentials are initially gathered and used for authentication. In most cases, an initial form gathers all credentials necessary for authentication.
- **All standard JAAS callbacks**—Sun Microsystems defines seven types of callbacks: NameCallback, PasswordCallback, ChoiceCallback, ConfirmationCallback, LanguageCallback, TextInputCallback, TextOutputCallback.

  Note: If a new callback type is encountered during authentication, the Web Server SSM ignores it.

- **Multiple authentication phases**—JAAS may ask several series of questions using callbacks. Therefore, the SSM does not assume that answering one set of callbacks is sufficient.
- **Web farms**—The following features are supported for web farms:
  - **Redirect during credential gathering**—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.
  - **Single sign-on**—Within a web farm it is possible that a user is authenticated on one machine and transparently redirected to another. However, a mechanism must be available by which the second machine can accept the identity from the first without having to re-authenticate the user. The identity is only shared within the same cookie domain.
- **Single sign-on with other SSMs**—Can share identity with a custom application that is protected by the Java SSM or another client of the Web Server SSM. The Java SSM and the Web Server SSM use the ALES Credential Mapper and ALES Identity Assertion providers. Single sign-on is limited to a cookie domain.
- **SAML Browser/POST profile**—Consumes an identity assertion from a SAML 1.1 Browser/POST transaction and provides an identity transfer service to serve SAML 1.1 identities to remote systems.
- **Custom, form-based authentication**—Allows for the editing and customizing the forms used in form-based authentication.

Authorization Service Features

The authorization service supports the following features:

- **Resource form**—De-references any use of "`..`" and decodes URL encoding. The resource is presented as the path element of a URL and the file or application name. For example, `http://www.example.com/framework.jsp?CNT=index.htm&FP=/products/aqualogic/` is presented as `/framework.jsp`. The query arguments CNT and FP and associated values are made available in the application context.
- **Allows for unprotected URLs**—Always uses the `isAuthenticationRequired()` method to check if a resource is protected by a security system. This feature is important because you may want to leave some web server resources unprotected.
- **Checks for resource authorization**—When checking for resource authorization, the following features are supported:

  - **Includes rich web service request context**—Because a web application cannot know what elements may be required for enforcement of security at the time of its authoring, it is important that information about the web services request be available in the context given to the security subsystem. The Web Server SSM provides HTTP headers, cookies, query arguments, and form values to the security subsystem. The SSM also decodes all URL encoded context elements before presenting them to the security subsystem.
  - **Works with existing unmodified web applications**—Does not require modification or special code to work with existing web applications running on the web server.

- **Retrieves response attributes during authorization**—Retrieves response attributes during the authorization process and provides them in a form that a layered web application can use.

Auditing Service Features

The auditing service has the following capabilities:

- **Audits all transactions**—All authentications, identity assertions, authorizations, role mappings, credential mappings and audit failures are automatically made available to the auditing infrastructure by the ALES security framework.
- **Audits session cleanup activity**—The occurrence of idle and absolute time-outs are audited.

Role Mapping Features

The role mapping service supports hard-coded roles in applications. Generally hard-coding behavior into an application based on roles is not recommended. It is possible, however, that some customers may need to replace an existing system that uses this mechanism or may want to use roles for user interface personalization. Support for this feature requires that a list of mapped roles available from a security provider for a particular request be provided in a usable form by applications running within the web server.

Note: It is important to note that roles are not global in ALES but can change depending upon the resource and various elements of the context.

Credential Mapping Features

ALES defines two types of credential objects: username/password credentials and generic credentials; however, there is no limitation as to the format of objects that can be used. Credentials can be mapped and associated with a resource and identity or an alias.

The credential mapping service has the following features:

- **Provides mapped username/password credential**s—Extracts mapped username/password credentials to the application running within the SSM. This username/password can be used for legacy SSO to log into a database or other system. The Web Server SSM does not use these credentials itself; it will make them available to the web server application.
- **Supports unknown credential types**—Provides a way to inject other credential formats. Since these other formats are unknown to the SSM, they must be converted to a string before being presented to the application.

Administration Features

Administering the security configuration involves writing policies for users, groups, roles, and the web application resources that the SSM protects. The Web Server SSM has the following features:

- **Presents the full URL**—The full URL (including the protocol, server name, port, full path, and query string) is presented to the ALES security framework as part of the context to allow its use in access control policy. Note that the resource presented to the system is in the canonical form. 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.
- **Uses the HTTP method as the action**—The HTTP method (GET, POST, HEAD, PUT) is presented by the Web Server SSM as the action for authorization. In the administration system the privilege must match the action for a policy so this feature allows for separate security policies to be applied to POSTs, GETs, and other methods.
- **Passes in an application context**—The application context is passed through to the SSM's authorization and role mapping security providers and is associated with any audit records logged. This context contains values relevant to the request environment at the time the security provider processes the call.

Session Management Features

To manage session behavior, the Web Server SSM supports the following capabilities:

- **Inactivity time-out**—terminates a session after it has been inactive for a configurable period of time

- **Absolute time-out**—terminates a session after a certain (usually large) period of time, thereby preventing a client from staying perpetually connected, which can be a security risk.
- **User logoff**—allows for user initiated logoff.
- **Forced logoff**—forces immediate logoff by terminating the session for a single DNS domain.
- **Session cookie**—uses session cookie, not persistent cookies.

Configuration Features

The web server is configured to use the filter component of the Web Server SSM. Local configuration of the web server should only be necessary once and should be static. The Web Server SSM has the following configuration capabilities:

- **Logging channel**—to support configuration and debugging.
- **Configurable flag to control logging and debug mode**—to determine what messages are logged.

Web Server 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.
- Does not support special handling of third-party cookies.
- Requires use of cookies to maintain session state.
- The Web Server SSM must be manually configured into the web server.
- Does not support load-balancing or failover in accessing the Web Services SSM.
- Must be installed on the same machine as the web server to which it is bound.
- 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 is involved in an attempt 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.

Web Server SSM Integration Tasks

This section provides an overview of integration tasks. Integration tasks center on managing SSM configurations (including the security providers) and configuring the web server to use the web filters. For additional information, see *Installing Security Service Modules*.

The major tasks performed are:

1. Create a SCM and a SSM configuration using the Administration Console. This includes specifying the security providers.
2. Create a parent resource for the application. This will contain ALES's representation of the application.
3. Create the SSM instance on the web server machine and enroll it in the ALES trust environment. The instance will use the security providers defined in step 1 above.

   During the instance creation process, the `default.properties` configuration file is created. This file contains the connection information for the ALES services.
4. Configure the web server environmental binding. This loads the web filter on the server and establishes the connection between the web server and ALES.

Configuring and Deploying Policy for the Web Server SSM

Developing a policy for a web application typically begins by determining which resources you want to protect. You then create authorization mapping policies to define access privileges and roles for each resource, and under what specific conditions. Next, you create role mapping policies that control which users and groups have membership in the defined roles, and under what conditions.

This section describes how to create resources and define authorization and role mapping policies to protect a sample Web server application. This section also describes how to deploy this policy

to the Web Services SSM that you will use to control access to sample Web server application resources.

AquaLogic Enterprise Security provides three tools for configuring application policy: the Administration Console, the Policy Import Tool, and Business Logic Manager (BLM) API. In this section you are directed to use the Administration Console to configure policy.

For more information on how to use the Administration Console to configure policy, see the *Policy Managers Guide* and Console Help.

For instructions on how to use the Policy Import Tool to import policy files, see the Importing Policy section in the *Policy Managers Guide*.

For information on the BLM, see the BLM API Javadocs.

To configure and deploy policy for the Web Server SSM, perform the following tasks:

- "Creating Resources" on page 5-12
- "Creating Policies" on page 5-14
- "Modifying Admin and Everyone Role Mapping Policies" on page 5-15
- "Configuring the Application Deployment Parent" on page 5-15
- "Distributing Policy and Security Configuration" on page 5-16

Creating Resources

This section describes how to use the Administration Console to create resources for the sample web server application resource.

Figure 5-6 shows the resources that you must create for the sample IIS Web Server configuration. You create the same resources for the Apache Web Server, except that you assign the `NamePasswordForm` a file extension of `.html`, instead of `.acc`.

Figure 5-6 Resources Tree for the IIS Web Server

To create these resources, perform the following steps:

1. In the Administration Console, open the Resources folder, and click Resources to display the Resources page.
2. Select Policy and click New. The Create Resource dialog box appears.
3. In the Name text box, enter `ssmws`, select Binding from the Type drop-down list box, and click Ok. The `ssmws` resource appears under the Policy node.
4. Select the `ssmws` resource and click Configure. The Configure Resource dialog box appears.
5. From the Type drop-down list box, select Binding Application, check the Distribution Point check box to on, and click Ok.
6. Select the `ssmws` resource and click New. The Create Resource dialog box appears.
7. In the Name text box, enter `favicon.ico`, and click Ok. The resource appears under `ssmws`.

   Note: The `favicon.ico` file is an icon requested by the Internet Explorer and Mozilla browsers for bookmarking a URL.
8. Select the `ssmws` resource and click New. The Create Resource dialog box appears.
9. In the Name text box, enter `test`, and click Ok. The resource appears under `ssmws`.
10. Select the `test` resource and click New. The Create Resource dialog box appears.
11. In the Name text box, enter `foo.html` and click Ok. The `foo.html` resource appears under the `test` resource.
12. Click the `test` resource and click New. The Create Resource dialog box appears.

13. In the Name text box, enter `NamePasswordForm.acc` for IIS (or `NamePasswordForm.html` for Apache), and click Ok. The resource appears under `test`.

Creating Policies

This section describes how to use the Administration Console to create authorization and role mapping policies to protect the sample Web server application resources. It includes authorization policies for the HTML files and role mapping policies to assign membership to those roles.

Table 5-1 lists and describes the authorization policies that you will create to protect the sample Web server application resources. These authorization policies allow users who are members of the `Everyone` role the `Get` access privilege to the `favicon.ico` and `GET` and `POST` access privileges to `NamePasswordForm.acc` (so users who have membership in the `Everyone` role can access the username/password form when authentication for a protected resource is needed). The policy also restricts access to `foo.html` to users in the `Admin` role.

Table 5-1 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.
`grant(GET, POST, //app/policy/ssmws/test/NamePasswordForm.acc, //role/Everyone) if true;`	On the IIS Web Server, grants `GET` and `POST` privileges for those in the `Everyone` role to access the `NamePasswordForm.acc` page. **Note:** For the Apache Web Server, use `NamePasswordForm.html`.
`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 5-1, perform the following steps:

1. Open the Policy folder, and click Policy. The Policy page appears.
2. Click Authorization Policies and click New. The Create Authorization Policy dialog box appears.
3. Select the Grant radio button.

4. To add privileges for the first policy listed in Table 5-1, click the Privileges tab, select the `GET` privilege from the Select Privileges from Group list box and add it to the Selected Privileges box.
5. Click the Resources tab, select the `favicon.ico` resource from the Child Resource box and add it to the Selected Resources box.
6. Click the Policy Subjects, select the `Everyone` role from the Roles List box, add it to the Selected Policy Subjects box, and click Ok.
7. Repeat steps 2 to 6 for each of the remaining authorization policies listed in Table 5-1. Notice that the `Admin` role is assigned to the `foo.html` resource.

Modifying Admin and Everyone Role Mapping Policies

This section describes how to use the Administration Console to modify the role mapping policies that will be used to control access to the sample Web Server application resources.

To modify the `Admin` and `Everyone` role mapping policies, perform the following steps:

1. Open the Policy folder, and click Role Mapping Policies. The Role Mapping Policies page appears.
2. Select the `Admin` role for ASI, and click Edit. The Edit Role Policy dialog appears.
3. Click the Resources tab, add the `ssmws` resource, and click Ok.
4. Repeat steps 2 to 4 for the `Everyone` role to add `ssmws` to the Everyone role.

Configuring the Application Deployment Parent

For the sample Web server application, the Application Deployment Parent setting on the ASI Authorization provider and the ASI Role Mapping provider must be set to `//app/policy/ssmws` and must be bound to the provider.

To configure these providers, perform the following steps:

1. In the Administration Console, click the ASI Authorization provider and click the Details tab.
2. Set the Application Deployment parent to `//app/policy/ssmws`, and click Apply.
3. Click on the Bindings tab and click bind to bind `//app/policy/ssmws` to this provider.
4. Repeat steps 1 to 3 for the ASI Role Mapping provider.

Configuring the ALES Identity Assertion and Credential Mapping Providers

To configure the ALES Identity Assertion and ALES Credential Mapping providers, perform the following steps:

Note: The ALES Identity Assertion provider and the ALES Credential Mapping provider work with one another; therefore, it is important that you ensure that their configuration settings match.

1. In the Administration Console, click the ALES Identity Assertion provider, select the Details tab, set the parameters as listed in Table 5-2, and click Apply.
2. Click the ALES Credential Mapping provider, select the Details tab, set the parameters as listed in Table 5-2, and click Apply.

Table 5-2 ALES Identity Asserter and Credential Mapper Provider Settings

Parameter	Setting
Trusted CAKeystore	`{HOME}/ssl/demoProviderTrust.jks` `{HOME}` is replaced with the SSM instance directory at runtime.
Trusted CAKeystore Type	`JKS`
Trust Cert Alias	`demo_provider_trust`
Trusted Cert Alias Password and Confirmation	`password`
Trusted Keystore	`{HOME}/ssl/demoProviderTrust.jks` `{HOME}` is replaced with the SSM instance directory at runtime.
Trusted Keystore Type	`JKS`

Distributing Policy and Security Configuration

Distribute the policy and security configuration to the Web Server SSM.

For information on how to distribute policy and security configuration, see the Console Help. Be sure to verify the results of your distribution.

Configuring the Web Server Environmental Binding

The Web Server Environmental Binding configuration procedures vary depending on the type of Web Server SSM you are configuring. AquaLogic Enterprise Security supports two Web server SSMs that require configuration of the Web Server Environmental Binding: the Microsoft IIS Web Server SSM and the Apache Web Server SSM. For configuration instructions, see the appropriate topic below:

- "Configuring the Environmental Binding for the Microsoft IIS Web Server" on page 5-17
- "Configuring the Environmental Binding for the Apache Web Server" on page 5-23

Configuring the Environmental Binding for the Microsoft IIS Web Server

To configure the environmental binding for Microsoft IIS Web Server, perform the following tasks:

- "Configuring the Microsoft IIS Web Server Binding Plug-In File" on page 5-17
- "Configuring the NamePasswordForm.acc File for the IIS Web Server" on page 5-22
- "Deploying and Testing the IIS Web Server Sample Application" on page 5-22

Configuring the Microsoft IIS Web Server Binding Plug-In File

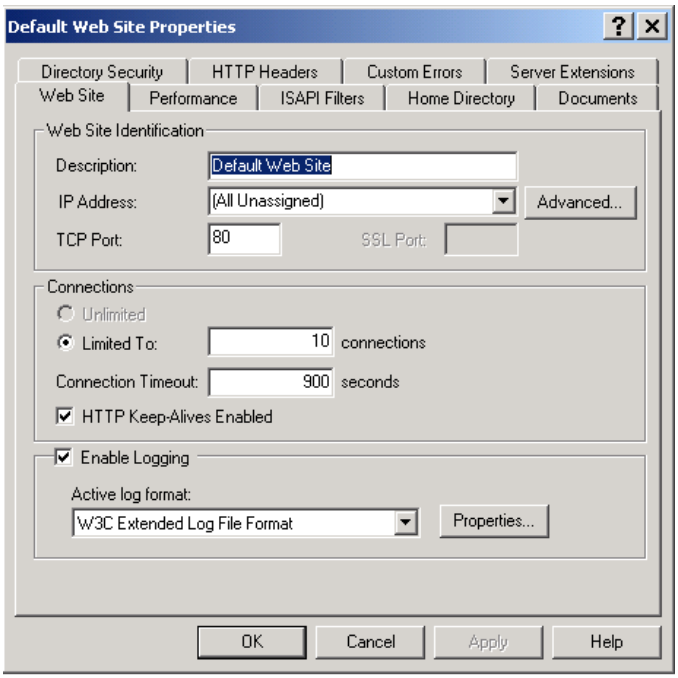
Note: This task assumes you have created an instance of the IIS Web Server SSM according instructions provided in Creating an Instance of a Security Service Module in *Installing Security Service Modules*.

The IIS Web Server Binding plug-in file is named `wles_isapi.dll`. This file is located in the *`BEA_HOME`*`\ales26-ssm\iis-ssm\lib` directory.

To configure the Microsoft IIS Web Binding plug-in, perform the following steps:

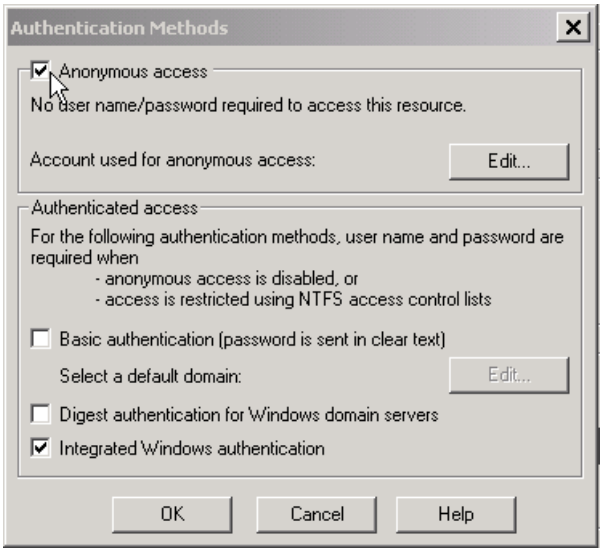
1. To open the Internet Information Services Manger, click Start>Settings>Control Panel, select Administrative Tools, and double-click Internet Services Manager. The Internet Information Services Window appears.
2. In the left-hand pane, expand the machine node, right click Default Web Site, and select Properties. The Default Web Site Properties dialog box appears (see Figure 5-7).

Figure 5-7 IIS Web Site Properties Dialog



3. Click the ISAPI Filters tab, click the Add button, assign a name to the ISAPI filter, use the Browse button to add the `wles_isapi.dll` file (which is located in *`BEA_HOME`*`\ales26-ssm\iis-ssm\lib` directory), and click OK.
4. Click the Directory Security tab. The Authentication Methods dialog appears (see Figure 5-8).

Figure 5-8 Authentication Methods Dialog



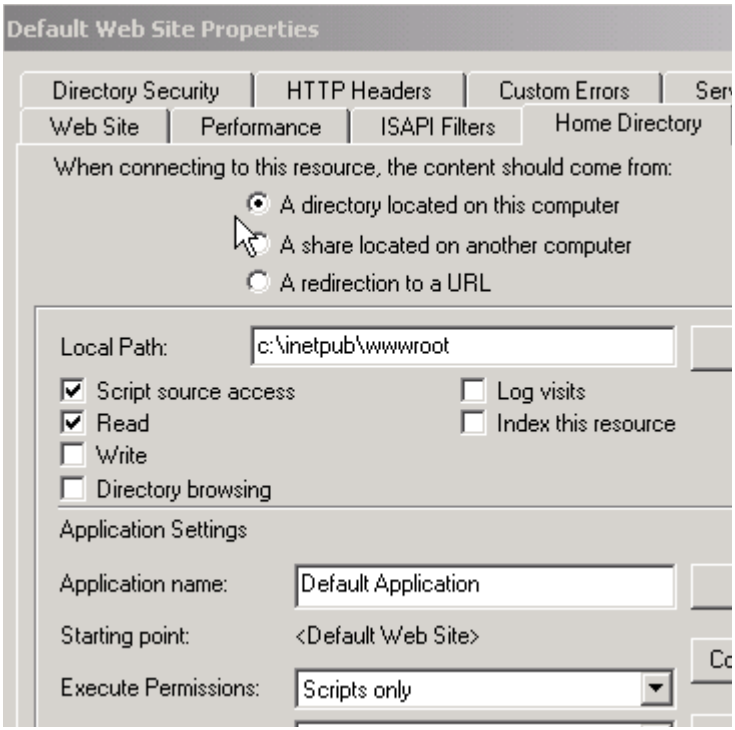
5. Click the Edit button for Anonymous Access, check the Anonymous username, and, if necessary, change the username and password to ensure that the Anonymous user has `Read` and `Read/Execute` permissions on the following directories:

   BEA_HOME`\ales26-ssm\iis-ssm\lib`
   BEA_HOME`\ales26-ssm\iis-ssm\instance\iisssmdemo\ssl`
   BEA_HOME`\ales26-ssm\iis-ssm\instance\iisssmdemo\config`

6. If you put the `NamePasswordForm.acc` file in a virtual directory, repeat the previous step for the virtual directory as well.

7. Return to the Default Web Site Properties dialog box (see Figure 5-7) and click the Home Directory tab. The Home Directory dialog appears (see Figure 5-9).

Figure 5-9 IIS Web Site Home Directory Dialog



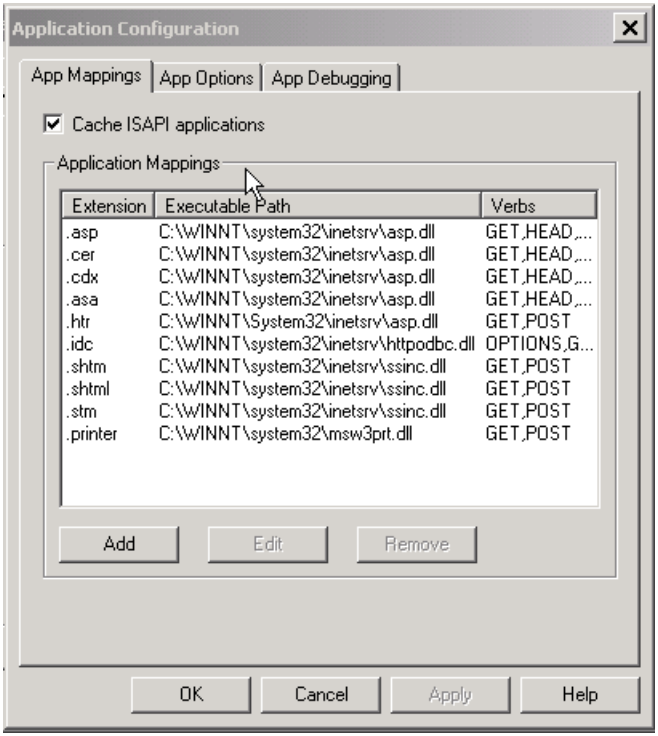
8. Verify that the property settings match the information in Table 5-3 and click Apply and OK.

Table 5-3 Home Directory Setting

Property	Setting
Local Path	`c:\inetpub\wwwroot`
Application name	Default Application
Execute Permissions	Scripts Only

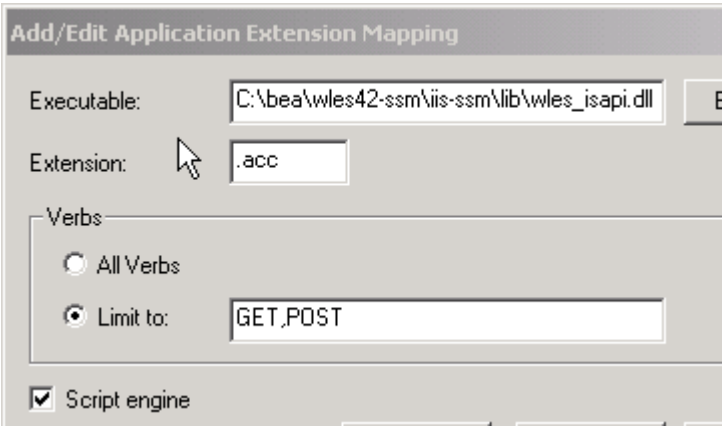
9. Click the Configuration button. The Application Configuration dialog appears (see Figure 5-10).

Figure 5-10 IIS Web Site Application Configuration Dialog



10. Click the Add button. The Add/Edit Application Extension Mapping Dialog appears (see Figure 5-11).

Figure 5-11 IIS Web Site Add/Edit Application Extension Mapping Dialog



11. Use the Browse button to add the `wles_isapi.dll` file to the Executable field, fill in the other fields as shown in Figure 5-11, and click OK.
12. Click OK to close the remaining windows.
13. Right click the Default Web Site again and start the Default Web Site. (Stop the Web Site first if necessary.)
14. Re-open the Default Web Site Properties dialog box and select the ISAPI Filters tab. The IIS Web Server Binding Plug-in status shows a green arrow to indicate that the IIS Web Server Binding Plug-in is loaded. If the green arrow is not displayed, add the `wles_isapi.dll` file again and start the IIS Web Server.

    Note: Be sure to start the IIS Web server with IIS SSM after you have started the Web Services SSM and ARME.

Configuring the NamePasswordForm.acc File for the IIS Web Server

Configure the `NamePasswordForm.acc` file for the IIS Web Server as follows:

```
<FORM METHOD=POST ACTION="test/NamePasswordForm.acc">
```

Deploying and Testing the IIS Web Server Sample Application

To set up the sample web application, perform the following steps:

Note: The Web Services SSM must be started before you perform this task because the filter and extension attempt to connect to the Web Services SSM when they are loaded by the Web server.

1. Set up the `IIS Server/wwwroot/test` directory as shown in Figure 5-12 and copy the following files to the `test` directory:
   - `NamePasswordForm.acc`
   - `foo.html`
   - `atnfailure.html`
   - `atzfailure.html`

   Note: The `NamePasswordForm.acc` file is provided in the *`BEA_HOME`*`\ales26-ssm\iis-ssm\instance\<`*`instancename`*`>\templates` directory. The `foo.html`, `atnfailure.html` and `atzfailure.html` files are not provided in the product installation kit. You should use your own versions of these files.

Figure 5-12 Deploying the Sample Application on the IIS Web Server

2. Start the IIS Web Server, open a browser and go to `http://<machine_name_with DNS_suffix>:80/test/foo.html`.
3. You are redirected to `NamePasswordForm.acc`.
4. Enter the system username/password (a default system username and password was set when you installed the Administration Application) and click OK. You are granted access to `foo.html`.

Configuring the Environmental Binding for the Apache Web Server

To configure the Apache Web Server, perform the following tasks:

- "Downloading and Installing the Apache Web Server" on page 5-24
- "Configuring the ALES Module" on page 5-24

Downloading and Installing the Apache Web Server

To download and install the Apache Web Server software, perform the following steps:

1. Go to the Apache download web site at `http://httpd.apache.org/download.cgi` and download and install the software.
2. Verify the following two modules are included in the installation:
   - *ServerRoot*`/modules/mod_include.so`
   - *ServerRoot*`/modules/mod_ssl.so`

   where *ServerRoot* is the Apache installation directory.

   Note: The Apache Web Server Security Service Module (SSM) requires that the above two modules be included in the Apache installation; otherwise the Secure Sockets Layer (SSL) and the Security Assertion Markup Language (SAML) server-server include (SSI) related functions will not work.

   Note: You may build your own 2.0.x version of the Apache Web Server with the above mentioned modules. If the modules are built into Apache, there may be no such files.

Configuring the ALES Module

Note: This task assumes you have created an instance of the Apache Web Server SSM according instructions provided in Creating an Instance of a Security Service Module in *Installing Security Service Modules*.

The ALES module contains only one file. For Windows, the file name is `mod_wles.dll`. For Sun Solaris and Linux, the file name is `mod_wles.so`.

To install and configure the ALES module:

1. Open the *ServerRoot*`/conf/httpd.conf` file and add a `LoadModule` directive. There are several `LoadModule` directives in the LoadModule section of the `httpd.conf` file. Add the following line to the end of the LoadModule section:

   `LoadModule wles_module <`*APACHE_SSM_HOME*`>/lib/mod_wles.so`

   where <*APACHE_SSM_HOME*> is the Apache Web Server SSM installation directory.

   For example:

   For Windows systems:

```
LoadModule wles_module c:\bea\ales26-ssm\apache-ssm\lib\mod_wles.dll

For UNIX systems:

LoadModule wles_module
/home/tiger/bea/ales26-ssm/apache-ssm/lib/mod_wles.so
```

2. Add a `WLESConfigDir` directive right after the above `LoadModule` directive as follows:

   ```
   <IfModule mod_wles.cpp>
   WLESConfigDir <APACHE_SSM_HOME>/instance/<instance_name>/config
   </IfModule>
   ```

   where the `config` directory is the directory that contains the `default.properties` file.

   Note: In the `IfModule` condition, be sure to specify `mod_wles.cpp`, not `mod_wles.c`.

3. To make sure your server works properly, configure the `ServerName`. For example:

   ```
   ServerName www.yourservername.com:8080
   ```

4. Change the Group directive to have the Apache Web Server running as the `asiusers` group so it can read the `mod_wles` file and other required files:

   ```
   Group asiusers
   ```

5. Edit the `envvars` file in the `ServerRoot/bin` directory, appending the directory where `mod_wles.so` resides to the default `LD_LIBRARY_PATH`, so that the file looks like this:

   ```
   LD_LIBRARY_PATH="/www/apache/lib:$LD_LIBRARY_PATH:<APACHE_SSM_HOME>/lib"
   ```

   Note: This step ensures that the Apache Web Server can load the dependency libraries for the `mod_wles` file.

6. Use the Apache `ctl` script to start or restart Apache Web Server in the `ServerRoot/bin` directory.

Configuring the NamePasswordForm.html File for the Apache Web Server

Configure the `NamePasswordForm.html` file for the Apache Web Server as follows:

```
<FORM METHOD=POST ACTION="/test/NamePasswordForm.html">
```

Deploying and Testing the Apache Web Server Sample Application

To set up the sample web application, perform the following steps:

1. Set up the `Apache Server/wwwroot/test` directory as shown in Figure 5-13 and copy the following files to the `test` directory:

- NamePasswordForm.html
- foo.html
- atnfailure.html
- atzfailure.html

Note: The NamePasswordForm.html file is provided in the *BEA_HOME*\ales26-ssm\apache-ssm\instance\<*instancename*>\templates directory. The foo.html, atnfailure.html and atzfailure.html files are not provided in the product installation kit. You should use your own versions of these files.

Figure 5-13 Deploying the Sample Application on the Apache Web Server

2. Start the Apache Web Server, open a browser, and go to http://<hostmachine.cookiedomain>:8088/test/foo.html.
3. You are redirected to NamePasswordForm.html
4. Enter the system username/password (a default system username and password was set when you installed the Administration Application) and click OK. You are granted access to foo.html.

Configuring Web Single Sign-on with ALES Identity Assertion

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

- Bi-directional web single sign-on between Web Server Security Service Modules (SSMs)

  With SSO configured, 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 web single sign-on between Web Server SSMs and WebLogic Server SSMs

  With SSO configured, 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.

For configuration instructions, see the following topics:

- "Configuring Web Server SSMs to Web Server SSMs for SSO" on page 5-27
- "Configuring Web Server SSMs to WebLogic Server SSMs for SSO" on page 5-27

Configuring Web Server SSMs to Web Server SSMs for SSO

To configure Web Server SSM to Web Server SSM to support web single sign-on, perform the following steps:

1. Using the Administration Console, configure the ALES Identity Assertion and ALES Credential Mapping providers for each Web Server SSM that is to participate in web single sign-on.
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 the SSM configurations to the SSMs.

For instructions on how to perform the above steps, see the Console Help for the Administration Console.

Configuring Web Server SSMs to WebLogic Server SSMs for SSO

To configure Web Server SSM to WebLogic Server SSM to support web single sign-on, perform the following steps:

1. Using the Administration Console, configure the ALES Identity Assertion and ALES Credential Mapping providers for each Web Server SSM and WebLogic Server SSM that is to participate in web single sign-on.
2. Configure the ALES Identity Assertion provider and the ALES Credential Mapping provider in each of the SSMs to use the same Trusted Cert Alias, Trusted Keystore, and Trusted Keystore Type.
3. When configuring the ALES Identity Assertion provider for each of the WebLogic Server SSMs, on the Details tab, be sure leave the Base64 Decoding attribute box unchecked, which is the default setting.

4. Deploy the SSM configurations to the SSMs.

For instructions on how to perform the above steps, see the Console Help.

Configuring Web Server SSM Properties

The Web Server SSM has a configuration file named `default.properties`. All configuration settings for the Web Server SSM instance are defined in this file. This file is pre-configured and placed in the proper location for you.

If you want to edit the `default.properties` file for your particular environment, refer to the parameters descriptions in the following sections:

- "Session Settings" on page 5-28
- "Authentication Settings" on page 5-29
- "Role Mapping Settings" on page 5-34
- "Credential Mapping Settings" on page 5-35
- "Naming Authority Settings" on page 5-36
- "Logging Level Setting" on page 5-37
- "Environment Variables Accessible Using CGI" on page 5-37

Session Settings

The AquaLogic Enterprise Security services are stateless services; it is the calling Web Services client that is responsible for determining session related information. In addition, in a web environment, a session does not necessarily end with an explicit logout, so session termination must be inferred from a lack of activity.

Table 5-4 describes the settings used to manage session behavior. You use these settings to configure the Web server session related behavior for the security configuration to which it applies.

Table 5-4 Session Settings

Session Setting	Description
`session.inactivity.timeout`	The number of seconds of inactivity that causes a session to expire. Default value: 600 seconds (10 minutes)
`session.absolute.timeout`	The number of seconds an active session is allowed to be available before it expires and the user is forced to re-authenticate. If this setting is set to zero, then established active sessions can continue indefinitely. Default value: 3600 seconds (60 minutes)
`session.cookie.name`	The name of the session cookie. Default value: ALESIdentityAssertion.
`session.forcedlogoffURL`	The name of the URL that, when accessed, forces the session to logoff.

Authentication Settings

Table 5-5 describes the settings that you use to configure the Web server authentication behavior for the security configuration to which it applies. Also, for information on mapping JAAS Callbacks, see "Mapping JAAS Callback Type to Form and Form Fields" on page 5-31.

Table 5-5 Authentication Settings

Authentication Setting	Description
`authentication.precedence`	An ordered, comma-separated list of types of identity creation. If identity information is available from multiple types of identity transfers, this list determines which identity to use. The valid identity type is: • FORM—credential information collected from an authentication provider using forms. Default value: `FORM`
`authentication.initialForm`	Specifies the first form presented for form-based authentication.

Table 5-5 Authentication Settings (Continued)

Authentication Setting	Description
`authentication.<callback type>[<prompt>] = <field>,<form URL>`	Given a question, this setting specifies what field on what form will answer that question. Notice that the <prompt> is shown as optional. However, the prompt is required if there are multiple callbacks of the same type, because there is no other way for the SSM to distinguish identical callback types. The prompt is obtained from the callback by calling the `getPrompt()` method, but it is not used in the display of the form. If the prompt setting is missing, then the Web Server SSM attempts to answer the callbacks in the order of the settings. If the order does not match the order of the providers, then authentication fails. For more information on using this setting, see "Mapping JAAS Callback Type to Form and Form Fields" on page 5-31.
`authentication.onatnfailure`	If authentication fails, and this setting is set to a URL, then rather than issuing a 401 Authentication Failed, the user will be redirected to the specified URL.
`authentication.onatzfailure`	If authorization fails and this setting is set to a URL, then rather than issuing a 403 Permission Denied, the user is redirected to the specified URL.

Table 5-6 describes the different types of authentication callbacks that are supported by the Web Server SSM.

Table 5-6 Authentication Callback Type Descriptions

Authentication Callback Type	Description
`authentication.nameCallback`	The form template responsible for collecting a name for a name callback. This form must exist in the same directory as the post handler.
`authentication.passwordCallback`	The form template is responsible for collecting a password for a password callback. This form must exist in the same directory as the post handler.
`authentication.choiceCallback`	The form template is responsible for collecting a choice for a choice callback. This form must exist in the same directory as the post handler.

Table 5-6 Authentication Callback Type Descriptions

Authentication Callback Type	Description
`authentication.confirmationCallback`	The form template is responsible for collecting a confirmation for a confirmation callback. This form must exist in the same directory as the POST handler.
`authentication.textInputCallback`	The form template is responsible for collecting some text input for a text input callback. This form must exist in the same directory as the post handler.

Mapping JAAS Callback Type to Form and Form Fields

There are two required and one optional configuration setting that specify what form and what field contain the information required to satisfy the authentication callbacks. The credential gathering form must use an HTTP `POST` method to specify this information. Listing 5-1 shows an example of how to use the `POST` method in the credential gathering form.

Listing 5-1 Example of Using the POST Method in the Credential Gathering Form

```
<FORM METHOD=POST ACTION="LoginNamePwdTextIn.html">
<!--#AUTHSTATE -->
<TABLE BGCOLOR="C0C0C0"><TR><TD>
<TABLE BGCOLOR="#FFFFFF">
<TR><TD COLSPAN="2" BGCOLOR="#C0C0C0">Please Login</TD></TR>
<TR><TD COLSPAN="2">User Name </TD><TR>
<TR><TD><!--#PROMPT --></TD><TD><INPUT NAME="username"></TD></TR>
<TR><TD COLSPAN="2">Password </TD><TR>
<TR><TD><!--#PROMPT.1--></TD><TD><INPUT TYPE=
             PASSWORD NAME="password"></TD></TR>
<TR><TD COLSPAN="2">Input Text </TD><TR>
<TR><TD><!--#PROMPT --></TD><TD><INPUT NAME="textinput"></TD></TR>
<TR><TD COLSPAN="2"> </TD><TR>
<TR><TD COLSPAN="2" ALIGN="CENTER"><INPUT TYPE="SUBMIT"
VALUE="OK"></TD><TR>
</TABLE>
</TD></TR></TABLE>
</FORM>
```

The form field defines the HTTP `POST` data name that results from a submitted form.

The settings have the following format:

```
authentication.<callback type>[<prompt>] = <field>:<form URL>
```

Given a question, this setting specifies what field on what form will answer that question. Notice that the `<prompt>` is shown as optional. However, if there are multiple callbacks of the same type, the `<prompt>` is required because there is no other way for the Web Server SSM to distinguish identical callback types. The `<prompt>` is obtained from the callback by calling the `getPrompt()` method, but it is not used in the display of the form. If the `<prompt>` setting is missing, then the Web Server SSM attempts to answer the callbacks in the order of the settings. If the order does not match the order of the authentication providers, then authentication fails.

The supported callback types are: `nameCallback`, `passwordCallback`, `textInputCallback`, `textOutputCallback`.

Table 5-7 provides examples of callback usage and more information on each supported callback type.

Table 5-7 Authentication Callback Usage Examples

Authentication Callback Types	Example/Discussion
Name and password callbacks	`authentication.nameCallback[]=username: /ales/NamePasswordForm.htm` `authentication. passwordCallback []= password: /ales/NamePasswordForm.htm`

Name, password, and textInput callbacks	`authentication.initialForm=/test/NamePasswordForm.html`
	`# username/password`
	`authentication.nameCallback[]=username:/test/ NamePasswordForm.html`
	`authentication.passwordCallback[]=password:/test/ NamePasswordForm.html`
	`# username/password/textInput`
	`authentication.nameCallback[]=username:/test/ LoginNamePwdTextIn.html`
	`authentication.passwordCallback[]=password:/test/ LoginNamePwdTextIn.html`
	`authentication.textInputCallback[]=textinput:/test/ LoginNamePwdTextIn.html`
	In this example the user will be prompted for username/password. The authentication provider then prompts for the user's mother's maiden name. The Web Server SSM redirects to `QuestionForm.htm` and knows from what field to get the information.

Table 5-7 Authentication Callback Usage Examples (Continued)

Authentication Callback Types	Example/Discussion

Name, password, and textInput callbacks	`authentication.nameCallback[]=username: /ales/NamePasswordForm.htm`
	`authentication. passwordCallback []= password: /ales/NamePasswordForm.htm`
	`authentication. textInputCallback ["maiden name"]=maiden_name: /ales/ QuestionForm.htm`
	`authentication. textInputCallback ["social security number"]=maiden_name: /ales/ QuestionForm.htm`
	In this example two providers require username/password callbacks, a third provider requires a textInputCallback for mother's maiden name, and a fourth provider requires a textInputCallback for a Social Security number: The prompts distinguish between the two textInputCallbacks.
	Note: The textInputCallback prompt requires quotes only if it contains embedded strings.

TextOutput Callback	The `textOutputCallback` is used to display a message to the user. Because the Web Server SSM does not create or update forms, if it gets a textOutputCallback, it redirects it to the form URL and adds the field as a query string argument and the message value. The application that processes the URL is responsible for parsing the query string and displaying the message.
Language callback	Language callbacks are handled internally by the Web server; the user is never prompted, so no configuration is needed. The user's browser Accept-Language header is checked for the preferred language it supports and that locale is returned to the authentication provider. If the user's browser has no Accept-Language header, the system default locale is used.

Role Mapping Settings

Table 5-8 describes the settings that you use to configure the Web server role mapping behavior for the policy domain to which it applies.

Table 5-8 Role Mapping Settings

Role Mapping Setting	Description
`rolemapping.enable`	If set to `true`, then roles are injected into the request stream as a comma separated list.
`rolemapping.name`	The name of the variable in which to put the roles. The default is: `ALES_ROLES`.

Credential Mapping Settings

Table 5-9 describes the settings that you use to configure the Web server credential mapping behavior for the policy domain to which it applies.

Table 5-9 Credential Mapping Settings

Credential Mapping Setting	Description
`credentialmapping.enable`	If set to `true`, then credentials for each request are injected into the request stream.
`credentialmapping.credtypes`	List of credential types to ask for in this policy domain. Only credentials that are mapped and that are supported by configured Credential Mapping provider are returned for a specific request. Therefore, asking for a credential does not guarantee that it is there. For example, to configure credential mapping to support the password for the database server, perform the following steps: • Set `credentialmapping.credtypes` to: `"credentialmapping.credtypes=DBPASSWORD"` • On the Details tab of the Database Credential Mapping provider in the Web Services SSM, set the `Allowed Types` parameter to `DBPASSWORD`. **Note:** The Database Credential Mapper provider provides identity credentials. An identity credential is the same as a `PasswordCredential` in Java. Others credentials, such SAML assertions, ALES Identity Assertions IA, and so on, are identity assertions. They are the same as a `GenericCredential` in Java. The Web Services SSM can have only one identity credential defined, but many identity assertions.
`credentialmapping.prefix`	Prefix to prepend to credential names, for example `CRED`.

Naming Authority Settings

Table 5-10 describes the settings that you use to configure the Web Server SSM naming authority.

Table 5-10 Naming Authority Settings

Setting	Description
`namingauthority.resource`	Specifies the naming authority for the resource. The naming authority is configured in the Web Services SSM.
`namingauthority.action`	Specifies the action naming authority.
`namingauthority.audit`	Specifies the audit naming authority.
webservice.registry.url	Specifies the URL of the Web Services Registry Service. For example: `http://localhost:8000/ServiceRegistry`

Logging Level Setting

Table 5-11 describes the settings that you use to configure the Web Server SSM naming authority.

Table 5-11 Logging Level Setting

Setting	Description
`log.level`	Specifies the logging level for the log4j Auditing provider.

Environment Variables Accessible Using CGI

The Web Server Security Service Module (SSM) tool kit enables you to access user environment variables using Common Gateway Interface (CGI).

Although security is embedded within the web server itself, requiring no special programming (if the user does not have access, your code will never run), a security administrator may want to use CGI to access and modify environment variables passed in by the Web Server Security Service Module. In order to customize the application according to the details of the security being enforced, a web application may access several environmental values in order to provide a more integrated user experience.

You can use CGI to access the following environment variables:

- ALES_IDENTITY—An authentication environment variable. It is available to a CGI programmer after a user successfully authenticates. This variable contains the username of the user, if available. It specifies the name of the HTTP header that will be added. The value of the variable is a list of the identity principals, including username and groups.
- ALES_DECISIONTIME—An authorization environment variable. It is available to a CGI programmer after a user is authorized to access a secure resource. It contains the date and time this authorization decision was rendered and has this format: "Monday June 23 15:14:21 EDT 2003"
- ALES_ROLES—A role environment variable that stores a list of roles calculated for the user.
- Credential Environment Variable—Table 5-12 describes the credential that is injected into the request stream when the user is authenticated. A CGI application can use this variable to access an LDAP store or database with an appropriate credential, rather than hard coding usernames and passwords. The prefix to this credential variable is configurable, although CRED is the default. Different credential types are handled differently, but the general format of the variable is: CRED_{NAME}={VALUE}

Table 5-12 Credential Environment Variables

Environment Variable	Description
Password Credentials	Password credentials conform to the format javax.resource.spi.security.PasswordCredential. The ManagedConnectionFactory element of this class is ignored. This credential type is rendered in the CGI environment as: {PREFIX}_PASSWORD={NAME}:{PASSWORD} where PREFIX is the configured prefix, NAME is the username, and PASSWORD is the password as a string. This name must match the requested credential type from credentialmapping.credtypes. For example: CRED_PASSWORD=system:weblogic

CHAPTER 6

Configuring the Web Services SSM

This section describes how to configure and use a Web Services Security Service Module (WS-SSM), after you have completed the installation and post-installation procedures described in *Installing Security Service Modules*. It includes the following sections:

Overview of the Web Services SSM

The Web Services SSM provides an application programming interface (API) that allows security developers to write custom application clients that invoke AquaLogic Enterprise Security services through SOAP. These interfaces support the most commonly required security functions, such as authentication, authorization, and auditing, and are organized into services that are logically grouped by functionality. A Web Services client can use the Web Services SSM (which incorporates the Security Services APIs, the Security Framework, and the configured security providers) to make access control decisions for the web server to which it is connected. Then you can use the AquaLogic Enterprise Security Administration Server to configure and deploy a security configuration to protect the web server application resources. Thus, the Web

Services SSM enables security administrators and web developers to perform security tasks for applications running on a web server.

The Web Services SSM includes a set of examples that illustrate Web Services client development in different environments. The examples are located in `BEA_HOME/ales26-ssm/examples`:

ssmWorkshop

Demonstrates how to access the ALES Web Services SSM through its published WSDL in a WebLogic Workshop 8.1 or 9.x environment.

ssmNET

Demonstrates how to access the ALES Web Services SSM through its published WSDL in the .NET 1.1 or 2.0 environment.

javaWebServiceClient

Demonstrates a simple Java client that accesses the ALES Web Services SSM for authorization.

XACMLClient

Demonstrates how to access the ALES Web Services SSM using the XACML protocol.

Note: For a Web Services client developed on Axis, use Axis 1.2 or later. For more information, see the Apache Axis site: http://ws.apache.org/axis/ .

For more information about developing security services for Web Services client applications, see *Programming Security for Web Services*.

Web Services Security Service APIs

The Web Services Security Service APIs enable access to the ALES security framework. These APIs provide the following security services:

Note: The following topics provide a very brief description of these APIs. For more information, see *Programming Security for Web Services*.

- "Authentication Service" on page 6-3
- "Authorization Service" on page 6-3
- "Auditing Service" on page 6-3

- "Role Mapping Service" on page 6-4
- "Credential Service" on page 6-4

Authentication Service

There are two variations of authentication, JAAS-based and identity assertion. JAAS-based authentication collects evidence (credentials) from a user in order to establish user identity.

Note: For more information on JAAS, see Sun's documentation at http://java.sun.com/products/jaas/.

Identity assertion authentication consumes a trusted token object to establish identity. The Web Services SSM supports both types of authentication.

- JAAS authentication is highly variable and is dependent upon the configured authentication provider. An authentication provider can use several different types of questions (modeled as callbacks) to collect information from the user.

  Note: The Web Services SSM does not support custom callback types.

- Identity assertion authentication is linked to a specific protocol. For example, X.509 certificate assertion is only valid within the context of a 2-way SSL handshake, and SAML identity assertion is only valid in the context of an Oasis SAML profile. The Web Services SSM implements the Oasis SAML Browser/POST profile and consumes or produces a SAML Identity Assertion.

Authorization Service

In addition to providing a simple permit or denied decision on a URL, the authorization service also has the ability to return attributes into the request as determined by the access control policy implemented. Because the inclusion of coding in the application to handle these attributes creates an undue coupling between the application and security infrastructure, the SSM inserts these returned attributes into the HTTP request header. Depending upon the technology used (ASP, CGI, ISAPI), these headers can be extracted and used by the application.

Auditing Service

The auditing service audits all transactions through the security subsystem. Every URL accessed is sent through the auditing infrastructure.

Role Mapping Service

Although roles are primarily used in authorization, some applications may wish to have access to the roles to which a user is mapped for the purposes of role-based personalization. In order to provide this information to the running applications, the Web Services SSM adds a list of roles to the HTTP request header. Depending upon the technology used (ASP, CGI, ISAPI), the application can extract this list of roles from the header and use it.

Credential Service

The credential service returns sensitive credentials to an application so that the application can use systems that require a secondary (or tertiary) layer of authentication. The Web Services SSM extracts mapped credentials from the security system and makes them available in the HTTP header for use by the application. Depending upon the technology used (ASP, CGI, ISAPI), the application can extract the credential headers and use them to authenticate to other back-end systems.

Configuring and Deploying Policy for the Web Services SSM

You configure and deploy policy on a Web Services SSM in the same way as you would on an IIS Web Server SSM or an Apache Web Server SSM. For information about the Web Server SSM procedures, see "Configuring and Deploying Policy for the Web Server SSM" on page 5-11.

Binding the Web Services SSM to a Web Services Client

The Web Services SSM can be used to protect application resources on customer designed and implemented Web Services clients. The Web Services Application Programming Interface (API) is provided for this purpose. For a description of the Web Services API, see *Programming Security for Web Services*.

Configuring SSL in the Web Services SSM

When you create a new WS-SSM instance, the WS-SSM is accessible via the HTTP protocol. The protocol is best for development and for debugging purposes, but should not be used in a production environment. For a production environment, BEA recommends that you use either one-way SSL or two-way SSL (SSL with client authentication).

This section describes how to enable one-way and two-way SSL communication between a Web Services SSM and its client. It is assumed that the reader has basic knowledge of the SSL protocol, Certificate Authorities (CA), X.509 certificates and Java Key Stores (JKS).

In this section, `%SSM_INST_HOME%` represents the installation folder of the WS-SSM instance in the file system, for example, `c:\bea\ales26-ssm\webservice-ssm\instance\wsssm`

Configuring One-Way SSL

When you configure a WS-SSM to use one-way SSL communication, the WS-SSM sends its identity certificate, and the CA that signed the identity certificate must be trusted by the client. The WS-SSM does not authenticate the client; thus, the client does not have to have its own certificate.

To configure a WS-SSM to use one-way SSL:

1. Stop the WS-SSM if it is running
2. Delete the contents of the `%SSM_INST_HOME%\apps` directory.
3. Run the following command to regenerate the content of the `apps` directory:

   ```
   %SSM_INST_HOME%\adm\ssmwsInstance.bat -h
   ```

4. Restart the WS-SSM.

The JKS of the certificates (identities) and the alias that uniquely identifies the identity within the store is defined by the `<identity>` XML element in the `%SSM_INST_HOME%\apps\ssmws-asi\SAR-INF\config.xml` file. By default, the key store file is `%SSM_INST_HOME%\ssl\identity.jks` and the alias is `wles-ssm`.

WARNING: Do not delete the `identity.jks` file, as it is used by other ALES servers to authenticate the WS-SSM. However, you can add any number of additional certificates to that file and reference them by unique aliases.

To access the WS-SSM using SSL, the client must use the HTTPS protocol, rather than HTTP. The client must also be configured to trust the server's certificate; otherwise the connection will be closed. This requires that the CA that signed the server's certificate must be in the client's trusted JKS.

The client's trusted JKS is defined by the system property `javax.net.ssl.trustStore` and the JKS password is defined by the system property `javax.net.ssl.trustStorePassword` property. (These properties are defined in the Java Secure Socket Extension (JSSE)

documentation.) You can specify these system properties by running the WS-SSM Java client with command line arguments such as:

```
-Djavax.net.ssl.trustStore=C:\jks\trust.jks
```

```
-Djavax.net.ssl.trustStorePassword="secretword"
```

For testing purposes, the client can use the `%SSM_INST_HOME%\ssl\trust.jks` JKS, which contains the CA that was used to sign the default server's identity.

Configuring Two-Way SSL

In two-way SSL (SSL with client authentication), the WS-SSM and the client must each present the other with a trusted certificate. Configuring two-way SSL is similar to configuring one-way SSL, but addition steps have to be taken to allow the WS-SSM server to trust the client.

Configuring a WS-SSM for Two-Way SSL

To configure the WS-SSM for two-way SSL communication:

1. Add the CA that signed the client's certificate to the trusted CA list of the WS-SSM. By default this list is the `%SSM_INST_HOME%/ssl/trust.jks` file.

   The WS-SSM's JKS for trusted CAs is defined by the `<trust>` element in the `%SSM_INST_HOME%\apps\ssmws-asi\SAR-INF\config.xml` configuration file. You can specify the filter for the alias prefixes inside the trusted JKS using the `aliasPrefix` XML attribute. If you specify an `aliasPrefix` attribute, then only the aliases that start with the given prefix will be used. By default the prefix of the alias must start with `domain(<YOUR_ALES_DOMAIN>)`. You can use one of these two approaches when adding a new trusted CA certificate:

   - Give the certificate an alias that starts with the prefix `domain(<YOUR_ALES_DOMAIN>)`.
   - As an alternative, give the certificate any other alias and modify the config.xml file to remove any `aliasPrefix` XML attribute. If there is no `aliasPrefix` attribute, then the WS-SSM trusts all CAs stored in the `trust.jks`.

2. Add the client's identity to the WS-SSM's trusted peer list. By default this list is the `%SSM_INST_HOME%/ssl/peer.jks` file. The JKS location is determined by the `<peer>` XML element inside the `%SSM_INST_HOME%\apps\ssmws-asi\SAR-INF\config.xml` file.
3. Stop the WS-SSM if it is running
4. Delete the contents of the `%SSM_INST_HOME%\apps` directory.

5. Run the following command to regenerate the content of the `apps` directory:

   ```
   %SSM_INST_HOME%\adm\ssmwsInstance.bat -s
   ```

6. Restart the WS-SSM.

WARNING: Do not delete the `trust.jks` or `peer.jks files`, as they are used for communication between the WS-SSM and other ALES servers. The best practice is to add new certificates to the same files and reference them by unique aliases.

Configuring a Web Services Client for Two-Way SSL

In contrast to one-way SSL, in the case of two-way SSL, the client has to supply its certificate (identity) upon the server's request. The JKS location that stores the identity is defined by the `javax.net.ssl.keyStore` system property. The password for decrypting the identity is defined by the `javax.net.ssl.keyStorePassword` system property. For more information, see the Java Secure Socket Extension (JSSE) documentation.

For development and testing purposes, you can use the `%SSM_INST_HOME%\ssl\identity.jks` key store, as it contains the certificate under the `wles-ssm` alias that it trusted by the server in its default configuration; moreover, the CA that signed the certificate is also trusted. You should never use this approach in production.

The steps necessary to create and use a Web Services client with two-way SSL can vary; the final goal is to create a JKS file that contains the right certificate chain and the private key. This section describes an example of how set up a Web Services client for two-way SSL. In the example, we use the `keytool` utility is used that is shipped with the standard Java Development Kit (JDK). For complete information about this utility, consult the Sun Microsystems `keytool` documentation.

To create the client's identity store.

1. Create your private/public key pair, self-signed certificate and a new JKS to store it.

   ```
   keytool -genkey -dname "cn=WS-SSM Client" -alias "WS-SSM Client"
   -keystore clientkeystore.jks -validity 365
   ```

2. Create Certificate Signing Request:

   ```
   keytool -certreq -alias "WS-SSM Client" -file WS-SSM-Client.csr -keystore
   clientkeystore.jks
   ```

3. Send the `WS-SSM-Client.csr` file to a CA for signing or sign it by your own CA.

4. Import the CA's certificate (`CA.crt`):

```
keytool -import -file CA.crt -alias Trusted-CA -keystore
clientkeystore.jks
```

5. Import the signed Certificate Reply from the CA (`WS-SSM-Client.crt`):

```
keytool -import -file WS-SSM-Client.crt -keystore clientkeystore.jks
-alias "WS-SSM Client"
```

6. Delete the CA's certificate, which is no longer needed:

```
keytool -delete -alias Trusted-CA -keystore clientkeystore.jks
```

You can test the key store by executing the following command:

```
keytool -list -keystore clientkeystore.jks -v
```

The output should be similar to the output in Listing 6-1:

Listing 6-1 Example Keytool List Output

```
Keystore type: jks
Keystore provider: SUN
Your keystore contains 1 entry
Alias name: ws-ssm client
Creation date: Apr 14, 2006
Entry type: keyEntry
Certificate chain length: 2
Certificate[1]:
Owner: CN=WS-SSM Client
Issuer: CN=exampleCN, OU=exampleOU, O="exampleO", L=San Exemplo,
ST=California, C=US
Serial number: 1
Valid from: Fri Apr 14 12:45:11 EDT 2006 until: Sat Apr 14 12:45:11 EDT 2007
Certificate fingerprints:
         MD5:  BD:2F:C4:9E:27:CB:3A:8F:D4:8E:FB:EA:4E:86:6E:9C
         SHA1: 11:56:D9:94:A0:E2:9B:BC:AD:EF:FD:83:0A:39:5F:0C:0A:B0:9D:22
Certificate[2]:
Owner: CN=exampleCN, OU=exampleOU, O="exampleO", L=San Exemplo,
ST=California, C=US
Issuer: CN=exampleCN, OU=exampleOU, O="exampleO", L=San Exemplo,
ST=California, C=US
Serial number: -7abad5cc20db248d7901d4359b06dbb5
```

```
Valid from: Thu Mar 09 18:42:33 EST 2006 until: Sat Mar 10 18:42:33 EST 2018
Certificate fingerprints:
         MD5:  CF:F8:EA:64:84:02:6F:AD:C4:2E:2B:4B:AC:20:7B:76
         SHA1: 7C:5A:C1:9F:E5:03:26:7E:D7:50:8A:72:20:24:8A:7E:0F:D8:22:CF
*****************************************
```

At this point, the client can use the `clientkeystore.jks` file as the client's identity key store.

To make the WS-SSM trust the client identity:

1. Import the CA's certificate to the server's trust key store (the `trust.jks` file):

   ```
   keytool -import -file CA.crt -alias Trusted-CA -keystore trust.jks
   ```

2. Import the client's identity certificate to the server's key store of trusted peers (peer.jks)

   ```
   keytool -import -file WS-SSM-Client.crt -alias Trusted-WS-Client
   -keystore peer.jks
   ```

3. In the `%SSM_INST_HOME%\apps\ssmws-asi\SAR-INF\config.xml` configuration file, remove the `aliasPrefix` XML attribute under the `<trust>` element.

Now you should be able to establish two-way SSL communication using the client's key store created in the example.

Adding New Identity Assertion Types

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

1. Create a new Java class as a holder for the identity assertion. Note that the new holder class must belong to the `com.bea.security.ssmws.credentials` namespace. In this procedure, we use a class named `com.bea.security.ssmws.credentials.TestCredHolderImpl` and a custom identity assertion type named `TestIA` as an example. See Listing 6-2 for an example holder class.

2. Add the JAR file containing the holder class to the Web Services SSM's classpath. To do this, modify the `WLESws.wrapper.conf` configuration file, which is located in *`BEA_HOME`*`/ales26-ssm/webservice-ssm/`*`instance-name`*`/config` . For example, if the holder class is contained in a file named `ssmwsCustomAssertion.jar`, add a line like this to `WLESws.wrapper.conf`:

   ```
   wrapper.java.classpath.40=C:/bea/ales26-ssm/webservice-ssm/lib/ssmwsCus
   tomAssertion.jar
   ```

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

3. Modify the mapping file for incoming messages. Mapping for incoming messages is controlled by the `castor.xml` file in the *BEA_HOME*`/ales26-ssm/webservice-ssm/lib/com/bea/security/ssmws/soap` directory. Add an entry like the following inside the `<mapping>` XML element:

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

4. Modify the mapping file for outgoing messages. Mapping for incoming messages is controlled by the `castor.xml` file in the *BEA_HOME*`/ales26-ssm/webservice-ssm/lib/com/bea/security/ssmws/credentials` directory. Add an entry like the following inside the `<mapping>` XML element

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

5. If you want to log SOAP messages received and sent by the Web Services SSM, modify the `log4j.properties` file in the *BEA_HOME*`/ales26-ssm/webservice-ssm/`*instance-name*`/config` directory. Change this line:

   ```
   log4j.appender.A1.Threshold=ERROR
   ```

   to read instead:

   ```
   log4j.appender.A1.Threshold=DEBUG
   ```

   and add the following entry:

   ```
   log4j.logger.com.bea.security.ssmws.server=DEBUG
   ```

Now, when you restart the Web Services SSM, it will use the new holder implementation and the mapping entries to convert back and forth between the token's XML and Java representations.

Listing 6-2 Sample Identity Assertion Holder Class

```
public class TestCredHolderImpl implements CredentialHolder
{
```

```
    private String m_cookie;
    public static final String m_Type = "TestIA";

    public void setCookie(String cookie)
    {
        m_cookie = cookie;
    }

    public String getCookie()
    {
        return m_cookie;
    }

    public Object getObject()
    {
        return getCookie();
    }

    public void setObject(Object cred)
    {
        setCookie((String)cred);
    }
    public String getType()
    {
        return TestCredentialHolderImpl.m_Type;
    }
    public String getAsString()
    {
        return m_cookie;
    }
}
```

CHAPTER 7

Configuring the WebLogic Server 8.1 SSM

This section covers tasks that you must perform after installing and completing the post-installation tasks for the WebLogic Server 8.1 Security Service Module. Note that the WebLogic Server 9.x Security Service Module uses a different security framework from the one used in the WLS 8.1 SSM and therefore has configuration procedures. See Chapter 8, "Configuring the WebLogic Server 9.x SSM" for more information.

The following topics are covered in this section:

Location of the WebLogic Server Domain

For the purposes of the example presented here, this document assumes that the WebLogic Server domain is in the following location:

`BEA_HOME/user_projects/domains/`*`mydomain`*

However, your domain can be in any location you desire. If you want to create a domain, you can use the WebLogic Server Configuration Wizard to create a domain or create it manually. The domain includes a `startWebLogic` file, which you are instructed to modify in "Modifying the startWebLogic File" on page 7-2.

Modifying the startWebLogic File

The WebLogic startup script does the following:

- Sets environment variables.
- Invokes the `java weblogic.Server` command, which starts a JVM that is configured to run a WebLogic Server instance.

Before you can start a WebLogic Server that uses BEA AquaLogic Enterprise Security, you must edit the `startWebLogic` file that is located in the WebLogic Server domain directory. For example:

`BEA_HOME/user_projects/domains/`*`mydomain`*

where:

- `user_projects` is the directory where your WebLogic Server user projects are located.
- `domains` is the directory where your WebLogic Server domain instances are located.
- *`mydomain`* is the name of the WebLogic Server domain instance you are using.

See Listing 7-1 for an example of a modified `startWebLogic` file. To edit the `startWebLogic` file, do the following:

1. Before the `CLASSPATH` is set, add a call to the `set-wls-env` script file in your the `bin` directory for your instance. The `set-wls-env` script sets environment variables that are used in the next steps: `WLES_PRE_CLASSPATH`, `WLES_POST_CLASSPATH` and `WLES_JAVA_OPTIONS`. For example:

   `BEA_HOME/ales26-ssm/wls-ssm/instance/`*`wls-ssm`*`/bin`

   Where:

   `ales26-ssm` is the directory where you installed the Security Service Module.

   `instance` is the directory where all instances are stored.

   `wls-ssm` is the name of the Security Service Module instance you created earlier.

   For example, if you created an instance called `myInstance`, the call looks like this:

On Windows:

```
call
"C:\bea\ales26-ssm\wls-ssm\instance\myInstance\bin\set-wls-env.bat"
```

On UNIX:

```
. "/bea/ales26-ssm/wls-ssm/instance/myInstance/bin/set-wls-env.sh"
```

2. Add the following line to the `CLASSPATH`:

   On Windows:

   ```
   %WLES_PRE_CLASSPATH% and %WLES_POST_CLASSPATH%
   ```

   On UNIX:

   ```
   ${WLES_PRE_CLASSPATH} and ${WLES_POST_CLASSPATH}
   ```

3. On Windows, add quotes to `%JAVA_HOME%\bin\java` in the `weblogic.Server` command.

   ```
   "%JAVA_HOME%\bin\java"
   ```

4. Add the following to the `java` command that starts WebLogic Server with the `weblogic.Server` class:

   On Windows:

   ```
   %WLES_JAVA_OPTIONS%
   ```

   On UNIX:

   ```
   ${WLES_JAVA_OPTIONS}
   ```

Listing 7-1 Modifying the startWebLogic.cmd File for Windows

```
...

set SERVER_NAME=myserver

call "C:\BEA_HOME\ales26-ssm\wls-ssm\instance\myInstance\bin\set-wls-env.bat"

set CLASSPATH=%WLES_PRE_CLASSPATH%;%WEBLOGIC_CLASSPATH%;
%POINTBASE_CLASSPATH%;%JAVA_HOME%\jre\lib\rt.jar;
%WL_HOME%\server\lib\webservices.jar;%CLASSPATH%;
%WLES_POST_CLASSPATH%

@REM Call WebLogic Server
echo .
echo CLASSPATH=%CLASSPATH%
echo .
```

```
echo PATH=%PATH%
echo .

echo ************************************************
echo *  To start WebLogic Server, use a username and   *
echo *  password assigned to an admin-level user.  For *
echo *  server administration, use the WebLogic Server *
echo *  console at http:\\[hostname]:[port]\console    *
echo ************************************************

"%JAVA_HOME%\bin\java" %JAVA_VM% %MEM_ARGS% %JAVA_OPTIONS% %WLES_JAVA_OPTIONS%
-Dweblogic.Name=%SERVER_NAME%
-Dweblogic.ProductionModeEnabled=%PRODUCTION_MODE%
-Djava.security.policy="%WL_HOME%\server\lib\weblogic.policy" weblogic.Server

ENDLOCAL
```

Defining Security Properties

You can use the `security.properties` file to set the necessary security properties. To set the security properties, create a `security.properties` file and put it in the WebLogic Server domain directory; for example:

`BEA_HOME/user_projects/domains/mydomain`

Include the information shown in Listing 7-2 in the `security.properties` file, where:

- `wles.realm` is set to the value of the Configuration ID entered for the Security Service Module using the ALES Administration Console (see the Console Help).
- `wles.default.realm` must be set to the same value as `wles.realm`.

You may also copy this file from the `BEA_HOME/ales26-ssm/wls-ssm/instance/`*`myInstance`*`/config` folder.

Note: The `security.properties` file is not required if you add these parameters to Java Options.

Listing 7-2 Security.properties File

```
wles.realm=ConfigurationID
wles.default.realm=ConfigurationID
```

Starting and Stopping Processes

After you install the Security Service Module, create the instance, and enroll it, you must start the necessary processes by running the appropriate batch or shell scripts. Before you start these processes, make sure that the Administration Server and all of its services are running.

For each machine, you must start the following processes:

- One Service Control Manager
- One Authorization and Role Mapping Engine (ARME) for each Security Service Module instance.

For instructions on how to start and stop the required processes, see Starting and Stopping Processes for Security Service Modules in the *Administration and Deployment Guide.*

Additional Post-Installation Considerations

When using the Database Authentication provider, ASI Authorization provider and ASI Role Mapping provider, refer to the following sections for important information:

- "Setting the Boot Login for WebLogic Server" on page 9-1
- "Creating a WebLogic Boot Policy" on page 9-2
- "Creating a WebLogic Console Policy" on page 9-5
- "Protecting Resources" on page 9-7

Protecting a Cluster of WebLogic Servers

If you want to protect a cluster of WebLogic Servers using AquaLogic Enterprise Security, you must make some addition changes to the security configuration and resource configuration. For information on how to protect cluster of WebLogic Servers, see the following topics:

- "Security Configuration" on page 7-6
- "Resource Configuration" on page 7-7
- "Policy Configuration" on page 7-8

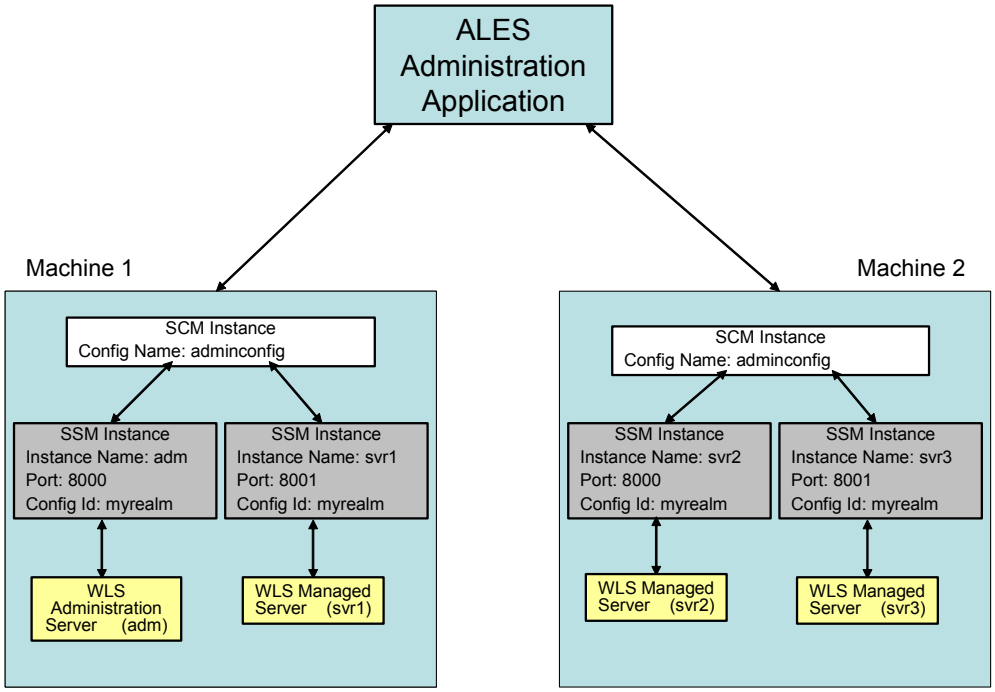
Security Configuration

Figure 7-1 shows a Security Service Module configuration named `myrealm`, located under a Service Control Manager named `adminconfig` in the AquaLogic Enterprise Security Administration Console. Your actual Security Service Module configuration will vary from this example based on the needs of your WebLogic domain.

Figure 7-1 Service Control Manager Configuration

Figure 7-2 shows a configuration for a cluster of four WebLogic Servers: one administration server (`adm`) and three managed servers (`svr1`, `svr2`, `svr3`), with one Security Service Module instance for each server. The Service Control Manager on both machines must use the same Configuration Name (`adminconfig`). Each Security Service Module must have a unique Instance Name and Port number per machine, but always shares a common Configuration ID (`myrealm`) across all machines. Thus, each server uses the same security provider configuration and receives the same policy.

Figure 7-2 WebLogic Server Clusters



Resource Configuration

You must also create the following three resources shown in Figure 7-3, setting them each as virtual resources.

The `myrealm/wl_management_internal1` resource is accessed on the cluster's administration server by the WebLogic Admin Console to view WebLogic Server related log files.

The `myrealm/wl_management_internal2` resource is accessed on the cluster's administration server by a managed server during bootstrap and file distribution operations.

The `myrealm/bea_wls_internal` resource is accessed when one managed server is synchronizing with another managed server.

The `myrealm/wl_management_internal1`, `myrealm/wl_management_internal2` and `myrealm/bea_wls_internal` resources must be configured to allow virtual resources.

Figure 7-3 Resources for Managing WebLogic Server Clusters

Policy Configuration

You must create the policy listed in Table 7-1.

Table 7-1 Policy Configuration

Privileges	Resources	Policy Subjects	Conditions
any	myrealm/bea_wls_internal	//sgrp/alesusers/allusers/	none
any	myrealm/wl_management_internal1, myrealm/wl_management_internal2	//sgrp/alesusers/allusers/	none

To create this policy in the ALES Administration Console:

1. Click Authorization Policies.
2. On the Authorization Policies page, click New.
3. In the Create Authorization Policy dialog, select the Privileges tab, click any in the Select Privileges from Group list box, and then click Add.

4. Select the Resources tab, expand `myrealm` in the Child Resources list box, select `bea_wls_internal`, and then click Add.
5. Select the Policy Subjects tab, select `allusers` from the Groups List list box, click Add, and then click OK. Be sure that the selected identity store is `alesusers`.
6. On the Authorization Policies page, click New.
7. In the Create Authorization Policy dialog, select the Privileges tab, click `any` in the Select Privileges from Group list box, and then click Add.
8. Select the Resources tab, expand `myrealm` in the Child Resources list box, select `wl_management_internal1`, and then click Add.
9. Select `wl_management_internal2` also and click Add.
10. Select the Policy Subjects tab, select `allusers` from the Groups List list box, click Add, and then click OK. Be sure that the selected identity store is `alesusers`.

CHAPTER 8

Configuring the WebLogic Server 9.x SSM

This section covers tasks that you must perform after completing the post-installation tasks for the WebLogic Server 9.x Security Service Module. The following topics are covered in this section:

Overview of the WebLogic Server 9.x SSM

The WebLogic Server 9.x Security Service Module integrates AquaLogic Enterprise Security with BEA WebLogic Server versions 9.1 and 9.2. It uses a different security framework from the one used in the WLS 8.1 SSM and the other ALES SSMs. When you install the WLS 9.x SSM, ALES uses the WLS 9.x security framework. As a consequence, 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. You still use the ALES Administration Console to configure SSMs other than the WLS 9.x SSM and to write security policies for any

SSM. You must also use the ALES Administration Console to configure the ASI Authorizer and ASI Role Mapper providers.

Simplified Procedure for Configuring WebLogic Server 9.x SSM

The manual (advanced) procedure for setting up the WebLogic Server 9.x SSM, described in "Manual (Advanced) Procedure for Configuring WebLogic Server 9.x SSM" on page 8-10 has several configuration options. Although these options increase flexibility, they make setup cumbersome for simple configurations. The ConfigTool provides a simplified procedure for setting up the WLS 9.x SSM for commonly-used configurations.

You can use this tool either in *silent* or *interactive* mode. In silent (or non-interactive) mode, all required configuration parameters are placed in ConfigTool.properties. In interactive mode, the tool prompts the user for configuration parameters.

The configuration tool does the following:

1. Creates the WLS 9.x SSM instance.
2. Generates random passwords for enrollment.
3. Enrolls the instance.
4. Creates the SSM configuration on the ALES Administration Console.
5. Edits the `startWeblogic` script for the WLS 9.x domain, to include ALES classpath and options.
6. Creates a new security realm on the WLS 9.x domain.

Prerequisites for Configuring the WebLogic Server 9.x SSM

Before you configure a WebLogic Server 9.x SSM, you must:

1. Install WebLogic Server 9.x . See the WebLogic Server *Installation Guide*.
2. Install the ALES Administration Server. See *Installing the Administration Server*.
3. Install the WebLogic Server 9.x SSM. See *Installing Security Service Modules*.
4. Create a WebLogic Server 9.x domain.

Configuring the WebLogic Server 9.x SSM

To configure the WLS9.x SSM:

1. Make sure ALES Administration Server is started. You can verify this by logging into ALES Administration Console Web page using Microsoft Internet Explorer. The default URL is `https://your-host:7010/asi`.
2. Make sure the WLS 9.x domain is not running. You can verify this by accessing the WebLogic Server Administration Console URL, at `http://your-host:7001/console/`.
3. Change the directory to *`BEA_HOME`*`/ales26-ssm/wls9-ssm/adm`.
4. Make sure that variables `JAVA_HOME` and `SSM_HOME` in the file `ConfigTool.bat|sh` are set correctly.
5. Make sure that file `log4j.properties` has the parameter `log4j.appender.ASIlogFile.File`, which is used for logging the output, set correctly.

Silent Configuration Mode

In this mode, you create the file `ConfigTool.properties` with the required options described in Table 8-1, and run `ConfigTool ConfigTool.properties`.

Table 8-1 Information Required in Silent Configuration Mode

Option Name	Description
instance.name:	Name of SSM instance
arme.port:	Authorization and role mapping engine (ARME) port number. You can use default value of 8000.
ales.admin.name:	Name of ALES Administrative user. You can use the default value *system*.
ales.admin.password:	Password used for ALES Administrative user
wls9.admin.name:	Name of Weblogic Administrative user. You can use the default value *weblogic*.
wls9.admin.password:	Password used for Weblogic Administrative user
domainDir:	Location of the new WLS9 domain
weblogic.home:	Location of weblogic server

Sample Contents of File ConfigTool.Properties:

A sample ConfigTool.Properties file is shown in Listing 8-1.

Listing 8-1 Sample ConfigTool.Properties File

```
instance.name = new_ssm
arme.port = 8000
ales.admin.name = system
ales.admin.password = weblogic
wls9.admin.name = weblogic
wls9.admin.password = weblogic
domainDir = D:/bea_home/user_projects/domains/new_wls_domain
weblogic.home = D:/bea_home/weblogic92
```

ConfigTool Interactive Mode Sample Output

Sample output from the ConfigTool is shown in Listing 8-4. User input is shown in bold.

Listing 8-2 Sample ConfigTool Output

```
D:\bea_home\ales26-ssm\wls9-ssm\adm>.\ConfigTool.bat configTool.properties
=========================================================================
AquaLogic Enterprise Security WLS9 SSM Configuration Utility
=========================================================================

====Creating WLS9.X SSM instance.....
Creating WLS9X SSM instance [new_ssm] ok
====WLS9 SSM Instance was Created====

====Enrolling SSM Instance ......
```

```
=========================================================================
Enrollment Tool Menu
=========================================================================
1.) Enroll SSM with generated random password
2.) Enroll SSM with input password

Select an option :> 1
Submitting enrollment request
Processing enrollment response
Updating trusted CA keystore
Updating peer keystore
====SSM Enrollment was Done====

====Creating ALES Security Realm ......
====ALES Security Realm was created====

====Creating WLS9 Domain Security Realm ......
running
[D:/bea_home/ales26-ssm/wls9-ssm/instance/new_ssm\..\..\adm\build.xml]
running default target[run]
configureing WLS9 domain is done.
====WLS9 Domain Security Realm was created====

=========================================================================
AquaLogic Enterprise Security WLS9 SSM Configuration Finished
=========================================================================
D:\bea_home\ales26-ssm\wls9-ssm\adm>
```

Interactive Configuration Mode

Run `ConfigTool.bat|sh`. The configuration tool prompts you to enter specific information about your system and configuration, as described in Table 8-2.

Table 8-2 Data Required in Interactive Configuration Mode

Data Element Name	Description	Default Values Available
SSM Instance Name	Provide a name which is used as: • `Configuration ID` of SSM instance • Parent resource on ALES admin console • Identity directory name	
ARME Port	SSM instance will use this port to communicate with ALES Admin.	8000
ALES Admin User Name	The name used to log in to ALES admin console	system
ALES Admin Password	The password used to log in to ALES admin console	
SSM private key password	Password to protect SSM private key, used in enrollment	
password for identity.jks	Password to protect identity key store, used in enrollment	
password for peer.jks	Password to protect peer key store, used in enrollment	
password for trust.jks	Password to protect trusted key store, used in enrollment	
WLS 9.x domain admin username	The user name used to log in to WLS console. This value is set when creating WLS 9.x domain	weblogic
WLS 9.x domain admin password	The password used to log in to WLS console. This value is set when creating WLS 9.x domain	

Table 8-2 Data Required in Interactive Configuration Mode

database user name	Database username of ALES policy database.	
We will use ALES policy database to store user information for Authentication & meta directory for Authorization		
database password	Database password of ALES policy database	
WLS 9.x domain's admin server name	WebLogic domain admin server name, the value by default is "AdminServer". This value can be obtained by looking up "admin-server-name" in file DOMAINPATH/config/config.xml	AdminServer
WLS 9.x domain's location (with domain name)	The directory where the WLS 9.x domain was created (for e.g. BEA-HOME/wlp92/user_projects/domains/my-domain)	
WLS 9.x domain name	This value is set when creating WLS 9.x domain	
WLS 9.x domain listening port	This value is set when creating WLS 9.x domain	7001

ConfigTool Interactive Mode Sample Output

Listing 8-3 shows the sample output when runing the ConfigTool in interactive mode. User input is shown in bold.

Listing 8-3 Interactive Mode Sample Output

```
D:\bea_home\ales26-ssm\wls9-ssm\adm> .\ConfigTool.bat

================================================================

AquaLogic Enterprise Security WLS9 SSM Configuration Utility
```

```
=====================================================================

====Creating WLS9.X SSM instance.....
please input ssm instance name:> new_ssm
please input arme port number:> 8000
Creating WLS9X SSM instance [new_ssm] ok
====WLS9 SSM Instance was Created====

====Enrolling SSM Instance ......

=========================================================================
==
Enrollment Tool Menu
=========================================================================
==
1.) Enroll SSM with generated random password
2.) Enroll SSM with input password

Select an option :> 1
please enter ales administration username:> system
Enter ales administration password:> password
Confirm ales administration password:> password
Submitting enrollment request
Processing enrollment response
Updating trusted CA keystore
Updating peer keystore
====SSM Enrollment was Done====
```

```
====Creating ALES Security Realm ......
please enter wls9 domain administration username:> weblogic
Enter wls9 domain administration password:> password
Confirm wls9 domain administration password:> password
====ALES Security Realm was created====

====Creating WLS9 Domain Security Realm ......
please input wls9 domain's location(with domain
name):>D:/bea_home/user_projects/domains/new_wls_domain
please input weblogic9 server's location:> D:/bea_home/weblogic92
running
[D:/bea_home/ales26-ssm/wls9-ssm/instance/new_ssm\..\..\adm\build.xml]
running default target[run]
configureing WLS9 domain is done.
====WLS9 Domain Security Realm was created====

================================================================================
==
AquaLogic Enterprise Security WLS9 SSM Configuration Finished
================================================================================
==
D:\bea_home\ales26-ssm\wls9-ssm\adm>
```

Post ConfigTool Tasks

Perform the following steps after you run the ConfigTool:

1. Start the WLS 9.x domain that is now secured by ALES. To do this, run `startWebLogic` from the WLS domain home.
2. Deploy an application or service that needs to be protected by ALES on this domain.

Manual (Advanced) Procedure for Configuring WebLogic Server 9.x SSM

The procedure described in "Simplified Procedure for Configuring WebLogic Server 9.x SSM" on page 8-2 is the preferred method for configuring the WebLogic Server 9.x SSM. However, the procedure described in this section is available as an alternate method for advanced users.

Prerequisites for Configuring the WebLogic Server 9.x SSM

Before you configure a WebLogic Server 9.x SSM, you must first:

1. Install WebLogic Server 9.x and create a WebLogic domain. See the WebLogic Server *Installation Guide*.
2. Install the ALES Administration Server and the ALES policy and configuration database. See *Installing the Administration Server*.
3. Install the WebLogic Server 9.x SSM. See *Installing Security Service Modules*.
4. Using the ALES Administration Console, create an instance of the WebLogic Server 9.x SSM, enroll the instance, and set the password for the SSM's ASI database.

Configuring the WebLogic Server 9.x SSM: Main Steps

To configure the ALES WebLogic Server 9.x SSM:

1. Copy the WLS 9.x console extension for the ALES security providers into the `console-ext` directory of your WebLogic Server domain. See "Console Extension for Security Providers in the WLS 9.x Console" on page 8-12.
2. Modify the WebLogic Server `startWebLogic` file. See "Modifying the startWebLogic File" on page 8-12.
3. Start WebLogic Server, using the modified `startWebLogic` file.
4. Using the WebLogic Server Administration Console, create a new security realm in WebLogic Server. See Configure new security realms in the WebLogic Server Console Help.
5. Configure security providers in the new WebLogic Server security realm. See "Configuring Security Providers for the WebLogic Server 9.x SSM" on page 8-15.
6. Make the new security realm the active security realm for WebLogic Server. See Change the default security realm in the WebLogic Server Console Help.

7. In the ALES Administration Console, create an SSM configuration using the same name as you used for the WLS security realm.
8. In the ALES Administration Console, create an instance of the ASI Authorizer and ASI Role Mapper providers. Set the Identity Directory attribute of the ASI Authorizer and ASI Role Mapper to the same value in the ALES Administration Console and the WebLogic Server Administration Console.
9. In the ALES Administration Console, create the Resource tree. See "Additional Post-Installation Considerations" on page 9-1.
10. In the ALES Administration Console, create users, groups, attributes and policy. See "Additional Post-Installation Considerations" on page 9-1.
11. Distribute policy and configuration. The WLS 9 SSM instance must be started after the configuration has been deployed. Policy changes can be deployed while the WLS 9 SSM instance is running.
12. Restart the WebLogic Server instance.

Console Extension for Security Providers in the WLS 9.x Console

ALES includes an extension to the WebLogic Server 9.x Administration Console. If you are using the WebLogic Server 9.x SSM, you must install the console extension in order for the ALES security providers to be visible in the WebLogic Server 9.x Administration Console.

To install the ALES security provider console extension, copy `ales_security_provider_ext.jar` from *`BEA_HOME`*`/ales26-ssm/wls9-ssm/lib` to the *`BEA_HOME`*`/`*`WLS_HOME`*`/domains/`*`DOMAIN_NAME`*`/console-ext` directory, where *`DOMAIN_NAME`* is the name of your WebLogic Server 9.x domain.

Modifying the startWebLogic File

The WebLogic Server startup script does the following:

- Sets environment variables.
- Invokes the `java weblogic.Server` command, which starts a JVM that is configured to run a WebLogic Server instance.

Before you can start a WebLogic Server instance that uses BEA AquaLogic Enterprise Security, you must edit the `startWebLogic` file. This file is located in the WebLogic Server domain directory. For example:

```
BEA_HOME/user_projects/domains/mydomain
```

where:

- `user_projects` is the directory where your WebLogic Server user projects are located.
- `domains` is the directory where your WebLogic Server domain instances are located.
- `mydomain` is the name of the WebLogic Server domain instance you are using.

See Listing 8-4 for an example of a modified `startWebLogic` file. To edit the `startWebLogic` file, do the following:

1. Make a copy of `/domains/mydomain/startWebLogic.cmd` or `startWebLogic.sh` and name it `startWebLogicALES.cmd` or `startWebLogicALES.sh`.
2. Make a copy of `/domains/mydomain/bin/startWebLogic.cmd` or `startWebLogic.sh` and name it `startWebLogicALES.cmd` or `startWebLogicALES.sh`.

3. Edit `/domains/mydomain/startWebLogic` so that it calls `/domains/mydomain/bin/startWebLogicALES` rather than `/domains/mydomain/bin/startWebLogic`. For example:

   ```
   call "%DOMAIN_HOME%\bin\startWebLogicALES.cmd" %*
   ```

4. Edit `/domains/mydomain/bin/startWebLogicALES`. Before the `CLASSPATH` is set, add a call to the `set-wls-env` script file in your the `bin` directory for your instance. The `set-wls-env` script sets environment variables that are used in the next steps: `WLES_POST_CLASSPATH` and `WLES_JAVA_OPTIONS`. For example:

   ```
   BEA_HOME/ales26-ssm/wls9-ssm/instance/wls-ssm/bin/set-wls-env.sh
   ```

   Where:

   `ales26-ssm` is the directory where you installed the Security Service Module.

   `instance` is the directory where all instances are stored.

   `wls-ssm` is the name of the Security Service Module instance you created earlier.

   For example, if you created a WLS SSM instance called `myInstance`, the call looks like this:

   On Windows:

   ```
   call
   "C:\bea\ales26-ssm\wls9-ssm\instance\myInstance\bin\set-wls-env.bat"
   ```

   On UNIX:

   ```
   . "/bea/ales26-ssm/wls9-ssm/instance/myInstance/bin/set-wls-env.sh"
   ```

5. Append the following to the `CLASSPATH`:

   On Windows:

   ```
   %WLES_POST_CLASSPATH%
   ```

   On UNIX:

   ```
   ${WLES_POST_CLASSPATH}
   ```

6. On Windows, add quotes to `%JAVA_HOME%\bin\java` in the `weblogic.Server` command.

   ```
   "%JAVA_HOME%\bin\java"
   ```

7. Add the following to the command that starts the server application:

   On Windows:

   ```
   %WLES_JAVA_OPTIONS%
   ```

On UNIX:

```
${WLES_JAVA_OPTIONS}
```

Listing 8-4 Modified startWebLogic File

```
...

. /BEA_HOME/ales26-ssm/wls9-ssm/instance/myInstance/bin/set-wls-env.sh

...
if [ "${WLS_PW}" != "" ] ; then
        JAVA_OPTIONS="${JAVA_OPTIONS} -Dweblogic.management.password=${WLS_PW}"
fi

CLASSPATH="${CLASSPATH}${CLASSPATHSEP}${MEDREC_WEBLOGIC_CLASSPATH}${WLES_POST_
CLASSPATH}"

echo "."

if [ "${WLS_REDIRECT_LOG}" = "" ] ; then
        echo "Starting WLS with line:"
        echo "${JAVA_HOME}/bin/java ${JAVA_VM} ${MEM_ARGS} ${JAVA_OPTIONS}
${WLES_JAVA_OPTIONS} -Dweblogic.Name=${SERVER_NAME}
-Djava.security.policy=${WL_HOME}/server/lib/weblogic.policy ${PROXY_SETTINGS}
${SERVER_CLASS}"

        ${JAVA_HOME}/bin/java ${JAVA_VM} ${MEM_ARGS} ${JAVA_OPTIONS}
${WLES_JAVA_OPTIONS} -Dweblogic.Name=${SERVER_NAME}
-Djava.security.policy=${WL_HOME}/server/lib/weblogic.policy ${PROXY_SETTINGS}
${SERVER_CLASS}

else

        echo "Redirecting output from WLS window to ${WLS_REDIRECT_LOG}"
        ${JAVA_HOME}/bin/java ${JAVA_VM} ${MEM_ARGS} ${JAVA_OPTIONS}
${WLES_JAVA_OPTIONS} -Dweblogic.Name=${SERVER_NAME}
-Djava.security.policy=${WL_HOME}/server/lib/weblogic.policy ${PROXY_SETTINGS}
${SERVER_CLASS} >"${WLS_REDIRECT_LOG}" 2>&1

fi
```

Configuring Security Providers for the WebLogic Server 9.x SSM

The WebLogic Server 9.x security framework includes a full set of security providers that are available out of the box. The WLS 9.x security providers are described in the WLS documentation, in the following chapters of *Securing WebLogic Server*:

- Configuring WebLogic Security Providers
- Configuring Authentication Providers

In addition, you can use the following ALES security providers by adding them to your WebLogic Server security realm:

- ASI Authorizer
- ASI Role Mapper
- ASI Adjudicator
- ALES Identity Asserter
- Log4J Auditor
- PerfDB Auditor
- Database Authenticator

Note: While you can use the WebLogic Server Administration Console to add these ALES security providers to a WebLogic Server security realm and to configure those security providers, the WLS console does not provide online help for the ALES security providers.

See the following topics in this section for detailed information about configuring the WebLogic 9.x SSM:

- "Configuring Security Providers for the WebLogic Server 9.x SSM" on page 8-15
- "Using the WebLogic Server Console to Configure Security Providers" on page 8-16
- "Using the ALES Administration Console to Configure Security Providers" on page 8-20

Configuring a WLS 9.x Security Realm for ALES

When you configure a WebLogic 9.x security realm for ALES, you must include at a minimum the following ALES security providers:

- ASI Authorizer—In order to take advantage of the ALES Authorization and Role Mapping Engine (ARME), your WLS security realm must include an instance of the ASI Authorizer.
- ASI Role Mapper—Your WLS security realm must also include an instance of the ASI Role Mapper in order to take advantage of the ALES Authorization and Role Mapping Engine.
- Log4J Auditor—The ALES security providers use a different logging system than the system used by WLS security providers. In order to support logging from any ALES security providers that are present, your WLS security realm must include an instance of the Log4J Auditor.
- ASI Adjudicator—If there are multiple authorization providers configured and unanimous permit is false and all authorization providers return ABSTAIN, then the ASI Adjudicator returns false, denying access. The default WLS Adjudicator returns true in the same scenario. Therefore, it is recommended that you use the ASI Adjudicator in order to obtain an appropriate adjudication result.
- If you plan to use WebLogic Portal (WLP), your security realm must include the XACML Authorizer and XACML Role Mapper providers in addition to the ones listed above.

Using the WebLogic Server Console to Configure Security Providers

To configure security providers for the WebLogic Server 9.x Security Service Module, you use the WebLogic Server Administration Console, not the ALES Administration Console. In order to create and configure ALES security provider instances using the WebLogic Server Administration Console, you must first install an extension to the console. See "Console Extension for Security Providers in the WLS 9.x Console" on page 8-12.

Before starting the procedure you might want to make a backup copy of the `config/config.xml` file in your domain directory. If you make a mistake following the steps below and the domain refuses to boot, you can undo the changes by restoring the `config/config.xml` file.

To configure security providers for ALES and WebLogic Server 9.x:

1. Start the WebLogic Server instance and log into the WebLogic Server Administration Console. The default URL for the console is `http://localhost:7001/console`.
2. In the Change Center in the upper left corner, click Lock & Edit.
3. In the left panel of the WebLogic Server Administration Console, under Domain Structure, select Security Realms.
4. On the Summary of Security Realms page, click New to create a new security realm. Create a new realm using a name that matches the configuration ID you used when you created the WebLogic Server 9.x SSM instance. For the purposes of this procedure, we will assume that the new security realm is named `mywls9ssm`.
5. On the Summary of Security Realms page, select the `mywls9ssm` security realm.
6. On the Configuration: General page:
   a. Set Security Model Default to Advanced.
   b. Uncheck Combined Role Mapping Enabled.
   c. Click Save.
   d. If Check Role and Policies is not visible, click Advanced.
   e. Set Check Role and Policies to All Web applications and EJBs.
   f. Click Save.
7. Select the Providers tab. You will configure new authentication, authorization, adjudication, role mapping, and auditing providers for the `mywls9ssm` security realm.
8. On the Providers: Authentication page, configure a new Database Authenticator security provider. To do this:
   a. Click New.
   b. Give the new Database Authenticator a name, such as `ALESDatabaseAuthenticator`.
   c. Select Database Authenticator as the Type.
   d. Click OK.
   e. Select the new Database Authenticator. On its Configuration: Common page, set the Control Flag to REQUIRED and click Save.

f. On the new Database Authenticator's Configuration: Provider Specific page, set the database login, password, JDBC driver class name and JDBC Connection URL. Click Save.

9. This step is required only if you are securing a WebLogic Portal (WLP) domain (that is, if you selected the "portal" option when creating the WebLogic domain). Make sure that your security realm's providers include a XACML Authorizer and a XACML Role Mapper.

   On the Providers: Authorization tab, check to make sure that a XACML Authorizer is present in your security realm. If it is missing, create a XACML Authorizer:

   a. Click New.

   b. Select XACML Authorizer and enter a name, such as XACMLAuthorizer, then click OK.

   c. If the XACML Authorizer is not the first authorization provider in the list, click the Reorder button, change the order, and click OK.

   On the Providers: Role Mapping tab, check to make sure that a XACML RoleMapper is present in your security realm. If it is missing, create a XACML Role Mapper:

   a. Click New.

   b. Select XACML RoleMapper and enter a name, such as XACMLRoleMapper, then click OK

   c. If the XACML RoleMapper is not the first role mapping provider in the list, click the Reorder button, change the order, and click OK.

10. Select the Providers: Authorization page and configure a new ASI Authorization provider:

    a. Click New.

    b. Give the new ASI Authorization provider a name, such as `ASIAuthorizationProvider`.

    c. Select ASIAuthorizationProvider as the Type.

    d. On the new ASI Authorization provider's Configuration: Provider Specific page, set Identity Directory and Application Deployment Parent. Click Save.

11. Select the Providers: Adjudication page and configure a new ASI Adjudication provider:

    a. Click Replace.

    b. Give the new ASI Adjudication provider a name, such as `ASIAdjudicator`.

    c. Select ASIAdjudicator as the Type.

d. On the ASIAdjudicator's Configuration: Provider Specific page, uncheck Require Unanimous Permit and click Save.

12. Select the Providers: Role Mapping page and configure a new ASI Role Mapper provider:

    a. Click New.

    b. Give the new ASI Role Mapper provider a name, such as `ASIRoleMapperProvider`.

    c. Select ASIRoleMapperProvider as the Type.

    d. On the new ASI Role Mapper provider's Configuration: Provider Specific page, set Identity Directory and Application Deployment Parent. Click Save.

13. Select the Providers: Auditing page and configure a new Log4j Auditing provider:

    a. Click New.

    b. Give the new Log4j Auditing provider a name, such as `Log4jAuditor`.

    c. Select Log4jAuditor as the Type.

14. Select the Providers: Credential Mapping page and configure a new Credential Mapping provider:

    a. Click New.

    b. Give the new Credential Mapping provider a name, such as `DefaultCredentialMapper`.

    c. Select DefaultCredentialMapper as the Type.

15. Select the Providers: Certification Path page and configure a Certification Path provider:

    a. Click New.

    b. Select WebLogicCertPathProvider and click Next.

    c. Click Next.

    d. Check Replace Existing Builder.

    e. Click Finish.

16. Change the default (active) security realm to your newly configured security realm. By default, a realm named `myrealm` is the active security realm when you install a WebLogic Server instance. To change the default security realm:

a. In the left pane of the WebLogic Server Administration Console, select your domain to open the Settings page for the domain.

b. On the Settings page for the domain, expand Security > General.

c. Select your new security realm, `mywls9ssm`, as the default security realm and click Save.

Note: If you create a new security realm but do not configure the required security providers, the new realm will not be available in the pull-down menu.

17. In the Change Center in the upper left corner, click Activate Changes.

Using the ALES Administration Console to Configure Security Providers

After you have configured security providers for the WebLogic Server 9.x Security Service Module using the WebLogic Server Administration Console, you need to make some configuration changes in the ALES Administration Console also. You need to configure the ASI Authorization and ASI Role Mapping providers and create required users and policy for the WebLogic Server 9.x SSM to start.

To configure security providers in the ALES Administration Console:

1. Log into the ALES Administration Console. The default URL for the console is `https://localhost:7010/asi`.
2. Create an SSM configuration using the same name as you used for the WebLogic Server security realm. The default used previously was `mywls9ssm`.
3. Create an instance of the ASI Authorizer and ASI Role Mapper providers.

   Set the Identity Directory attribute of the ASI Authorizer and ASI Role Mapper to the same value you specified when configuring the provider in the WebLogic Server Administration Console, as described in Using the WebLogic Server Console to Configure Security Providers.

   Identity Directory is the only attribute that you need to explicitly set; you can accept the default for the other attributes.
4. Create the Resource tree. For information about how to do this, see “Additional Post-Installation Considerations” on page 7-5.
5. Create users, groups, attributes and policy. For information about how to do this, see “Additional Post-Installation Considerations” on page 7-5.

6. Distribute policy.

Note: The WebLogic Server instance must be started after the configuration has been deployed. Other policy changes can be deployed while the WebLogic Server instance is running.

CHAPTER 9

Post-Installation Considerations for WLS 8.1 SSM and WLS 9.x SSM

This section describes to post-installation steps that you perform for both the WLS 8.1 SSM and the WLS 9.x SSM.

Additional Post-Installation Considerations

When using the Database Authentication provider, ASI Authorization provider and ASI Role Mapping provider, refer to the following sections for important information:

Setting the Boot Login for WebLogic Server

The WebLogic Server uses the login information contained in the `boot.properties` file to start the server. This file contains a `username` and `password` that must match a username and password in the configured authentication policy. The `boot.properties` file is located in the WebLogic Server domain directory on the machine on which the Security Service Module is installed, for example:

`BEA_HOME/user_projects/domains/`*`mydomain`*

If you used a username of `system` and a password of `weblogic`, then modify WebLogic Server `boot.properties` in the domain as follows:

```
user = system
password = weblogic
```

The next time you start the WebLogic Server, the username and password you specified are encrypted.

Creating a WebLogic Boot Policy

Before you can use the ASI Authorization provider with the WebLogic Server, you need to configure a boot policy, and then distribute it to the WebLogic Server Security Service Module. The boot policy allows the user named `system` to start the WebLogic Server instance. If you need instructions on how to perform any of the following tasks, see the Console Help for details. You may also want to refer to the *Policy Managers Guide* for information on how the policy language is constructed and how it appears in the console.

To configure and distribute a boot policy, perform the following tasks:

Creating the User Identity

To create the user identity named `alesusers`, perform these steps:

1. Using the ALES Administration Console, create an Identity directory called `alesusers`.
   a. Open the Identity folder and click Identity.
   b. Click New. In the Name text box, enter `alesusers`, and then click OK.
2. Within this directory, create a user named `system` and set the password for system to `weblogic`. Replace `system` and `weblogic` with the values used in `boot.properties` file.
   a. Click Users, click New, enter `system`, and click OK.
   b. Click Edit, click Set Password, enter `weblogic`, and click OK.

c. Click OK.

Creating Resources for WebLogic Server

Create the following resources below the resource called `policy` for the defined user, `alesusers`:

- `wlsserver` as a bound application node.
- `wlsserver/shared` as virtual
- `wlsserver/shared/svr`

To create these resources using the ALES Administration Console:

1. Click Resources and then click New.
2. In the Name box, type `wlsserver`, select Binding from the Type drop-down menu, and then click OK.
3. Select `wlsserver` and click Configure.
4. From the Type drop-down menu, select Binding Application, check Distribution Point, and then click OK.
5. Select `wlsserver`, click New, enter `shared` in the name box, and then click OK.
6. Select `shared`, click Configure, check Allow Virtual Resources, and then click OK.
7. Select `shared`, click New, enter `svr` in the name box, and then click OK.

Grant Server Resource to Admin Role

Create the following policy:

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

1. Expand the Policy node in the left pane, and click Authorization Policies.
2. In the Authorization Policies page, click New.
3. In the Create Authorization Policy dialog page, select the Privileges tab, select `any` in the Select Privileges from Group list box, and then click Add.
4. Select the Resources tab, expand the `wlsserver` and `shared` nodes in the Child Resources list box, select `svr`, and then click Add.

5. Select the Policy Subjects tab, select `Admin` from the Roles List list box, click Add, and click OK.

Grant Admin Role to WebLogic User/Group

Create the following role mapping policy:

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

1. Click Role Mapping Policies.
2. In the Role Mapping Policies page, click New.
3. In the Create Role Mapping Policy dialog page, select the Roles tab, select Admin from the Available Roles list box, and click Add.
4. Select the Resources tab, select `wlsserver` in the Child Resources list box, and click Add.
5. Select the Policy Subjects tab, select Users from the Select Policy Subjects From: drop-down menu, change the directory to `alesusers`, select `system` from the list box, click Add, and click OK.

Binding the Resource to the ASI Authorization Provider

To bind the resource `//app/policy/wlsserver` to the ASI Authorization provider for this Security Service Module, perform the following steps:

1. Open the Security Configuration and Security Control Manager folders.
2. Open the Security Service Module folder and click Authorization.
3. The Authorization page appears.
4. Click Create a new ASI Authorization Provider.
5. The Edit ASI Authorization Provider page appears.
6. Enter a name for the provider in the Name text box, and then click Create.
7. Click the Details tab, set the Identity Directory to `alesusers`, set the Application Directory Parent to `//app/policy/wlsserver`.
8. Click Apply.

9. Click the Bindings tab and select the resource you want to bind to the provider from the Bind drop-down menu, and then click Bind.

Distributing the Policies to the Security Service Module

Distribute the policies to the WebLogic Server Security Service Module.

For information on how to distribute policies, see the Administration Console help system. Be sure to verify the results of the distribution.

Creating a WebLogic Console Policy

Before you can login into the WebLogic Server Administration Console, you need to configure a console policy and then distribute it to the WebLogic Server Security Service Module. This is needed if you want to access the WebLogic Server Administration Console.

To configure and distribute a WebLogic Server Administration Console policy, do the following on the AquaLogic Enterprise Security Administration Console:

1. Create the following resource:

   `//app/policy/wlsserver/console`

   a. Click Resources. The Resources page appears.

   b. Select `wlsserver`, click New, enter `console` in the name box, and then click OK.

   c. Select `console`, click Configure, check Allow Virtual Resources, and then click OK.

2. Create the following resource:

   `//app/policy/wlsserver/console/url/console/login/bea_logo.gif`

   The resource represents the BEA logo image at the top-right corner on the login page of the Server Administration Console. To create this resource:

   a. Click Resources. The Resources page appears.

   b. Right-click on resource `//app/policy/wlsserver/console` and select Add Resource in the context menu..

   c. In the Create Resource dialog window, give the name `url` to the new resource.

   d. Right-click on the resource `//app/policy/wlsserver/console/url` and select Add Resource in the context menu..

   e. In the Create Resource dialog window, give the name `console` to the new resource.

f. Right-click on the resource `//app/policy/wlsserver/console/url/console` and select Add Resource in the context menu..

g. In the Create Resource dialog window, give the name `login` to the new resource.

h. Right-click on the resource `//app/policy/wlsserver/console/url/console/login` and select Add Resource in the context menu..

i. In the Create Resource dialog window, give the name `bea_logo.gif` to the new resource.

3. Create the following authorization policy, which allows a user with role Admin to access all the resources associated with the `console` application:

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

a. Click Authorization Policies.

b. In the Authorization Policies page, click New.

c. In the Create Authorization Policy dialog page, select the Privileges tab, click `any` in the Select Privileges from Group list box, and then click Add.

d. Select the Resources tab, expand `wlsserver` in the Child Resources list box, select `console`, and then click Add.

e. Select the Policy Subjects tab, select `Admin` from the Roles List list box, click Add, and then click OK.

4. Create the following authorization policy, which allows any user to see the BEA logo image at the top-right corner on the login page of the Server Administration Console:

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

a. Click Authorization Policies.

b. In the Authorization Policies page, click New.

c. In the Create Authorization Policy dialog page, select the Privileges tab, click GET in the Select Privileges from Group list box, and then click Add.

d. Select the Resources tab, expand `wlsserver/console/url/console/login` in the Child Resources list box, select `bea_logo.gif`, and then click Add.

e. Select the Policy Subjects tab, select `allusers` from the Groups List list box, click Add, and then click OK. Be sure that the selected identity store is `alesusers`.

5. Distribute the policies to the WebLogic Server Security Service Module. For information on how to distribute policy, see the Administration Console's help system. Be sure to verify the results of the distribution.

Protecting Resources

When you secure an EJB using a WebLogic Server Security Service Module, you must follow these steps if you want to use the AquaLogic Enterprise Security providers instead of the default WebLogic providers.

1. Modify the EJB deployment descriptor (`ejb-jar.xml`) so that the assembly-descriptor does not have any method-permissions set to unchecked or excluded.

   If either of these settings is present in the deployment descriptor, then the EJB container enforces them rather than calling into the security subsystem.

2. Set the following system property to true, indicating that the EJB container delegates other security checks to the security subsystem, by adding this line to the `WLES_JAVA_OPTIONS` in the `set-wls-env` script:

   ```
   weblogic.security.fullyDelegateAuthorization=true
   ```

CHAPTER 10

Integrating with BEA Workshop for WebLogic Platform

This section describes how to use AquaLogic Enterprise Security with BEA Workshop for WebLogic Platform. It covers how to:

- Add security-related annotations to your EJB code and
- Use the ALES Tag Library in your Java Servlet Pages (JSPs) to get user roles and entitlements for a user and a particular resource, and to determine whether access is allowed to that resource.

It includes the following topics:

Overview of the ALES Annotations Plug-in

BEA Workshop for WebLogic Platform is an IDE for developing standards-based enterprise-level applications that integrates standard and open source technologies such as Java EE, Eclipse and Apache Beehive. Using Workshop, you can easily add annotations to your Java code. Annotations is a feature of J2SE 5.0 that enables adding metadata to Java code. Later, the metadata can be used for different purposes such as generation of documentation, compile-time validation, and creation of deployment descriptors. The metadata can also be accessed during runtime.

AquaLogic Enterprise Security includes a plug-in for Workshop that gives you the ability 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 you can then import 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 seurity 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.

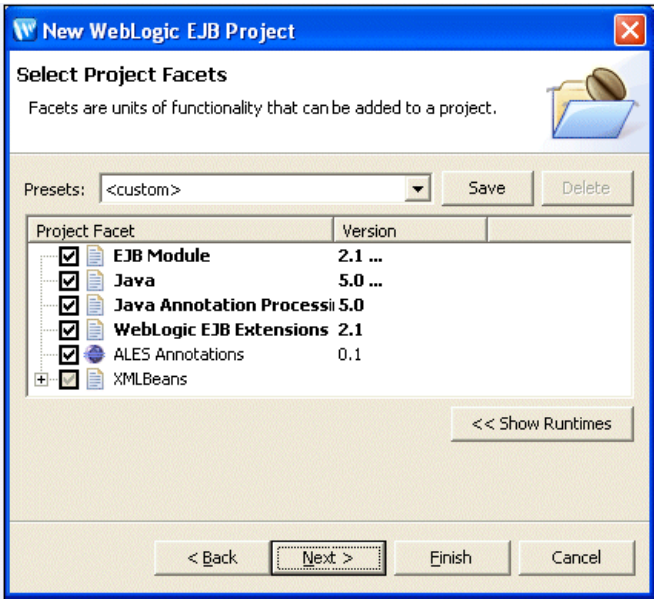
Setting Up the ALES Annotations Plug-in for Workshop

To set up the ALES Annotations plug-in for Workshop:

1. Install WebLogic Server with BEA Workshop for WebLogic Platform. For information about installing WebLogic Server, see the *BEA Products Installation Guide*.
2. Install the ALES Administration Server. For information about installing the ALES Administration Server, see *Installing the Administration Server*.
3. Copy the plug-in JAR file from ALES to Workshop. The Workshop plug-in for AquaLogic Enterprise Security is a JAR file named `com.bea.wlw.ales.annotations_9.2.0.jar`, which is located at `ALES_ADMIN_HOME/lib/eclipsePlugins`. To install the plug-in, copy `com.bea.wlw.ales.annotations_9.2.0.jar` to `BEA_HOME/WORKSHOP_HOME/workshop4WP/eclipse/plugins`.

The next time you start Workshop, you will see the ALES Annotations facet in a new or existing WebLogic EJB project.

Figure 10-1 Project Facets



Example: Using ALES Annotations in a WebLogic Bean Class

This section describes how you might use Workshop with the ALES Annotations plug-in. In this example, we show how to:

- "Create a WebLogic SessionBean" on page 10-4
- "Add ALES Annotations to the WebLogic Bean Class" on page 10-4
- "Configure ALES Annotations Properties" on page 10-5
- "Export the ALES Policy File from Workshop" on page 10-6
- "Import an ALES Annotations policy using policyIX" on page 10-7
- "Use the Resources Defined with ALES Annotations to Write Policies" on page 10-9

Create a WebLogic SessionBean

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

1. Create a new project named `EjbExample`.
2. Select `EjbExample` in the left panel, right click the `src` node and select **New > Package**. Name the package `beans`.
3. Right click on `beans`, select **New > WebLogic Session Bean**, and set the file name to `AccountService`.

Add ALES Annotations to the WebLogic Bean Class

Next, we will use Workshop to add security annotations to our `AccountService` class.

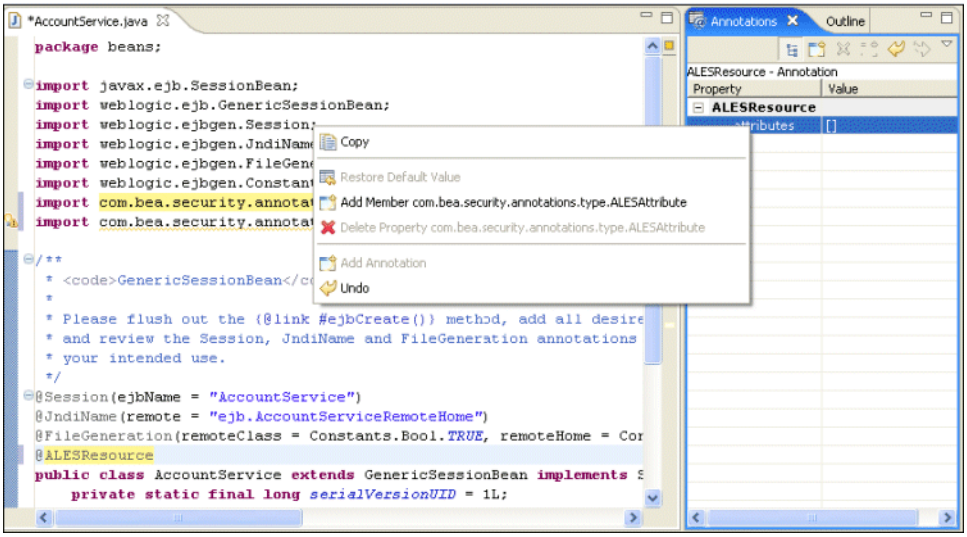
In the edit window for the Weblogic Session Bean class `AccountService.java`, import the following two classes:

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

Define the Bean class `AccountService` as an ALES resource:

1. Before the class definition, add the annotation `@ALESResource`. The **ALESResource – Annotation** property appears in the Annotations viewer.
2. Under `ALESResource`, right click **attributes** and click **Add Member com.bea.security.annotations.type.ALESAttribute**. We can now assign attributes to this resource.

Figure 10-2 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 these resources using the ALES Annotations plug-in. For example, annotate the `ejbCreate()` method to define it as an ALES resource with the ALESAttribute `operation = create`.

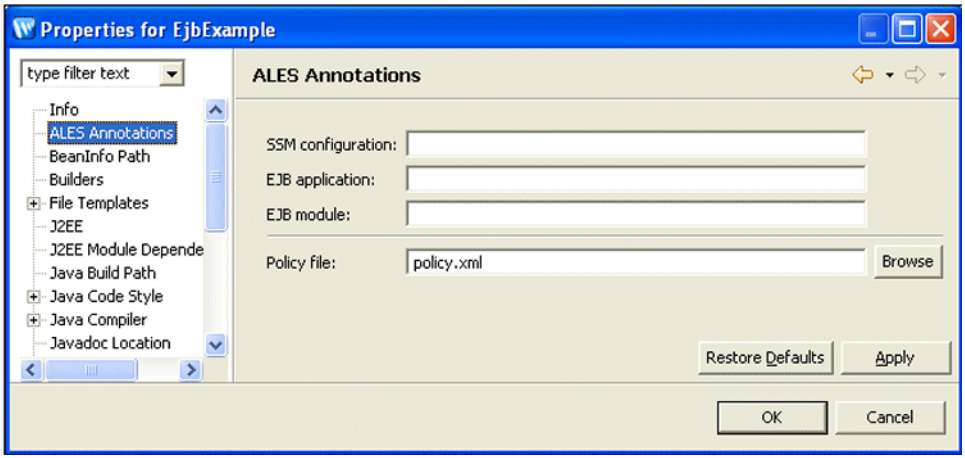
   ```
   @ALESResource(attributes={@ALESAttribute(name = "operation", value =
   "create")})
   ```

Configure ALES Annotations Properties

Next, we configure ALES-specific properties for our project:

1. In the left panel of Workshop, right click the **EjbExample** project node and click **Properties**. The **Properties** window for the project **EjbExample** appears.

Figure 10-3 ALES Annotation Project Properties



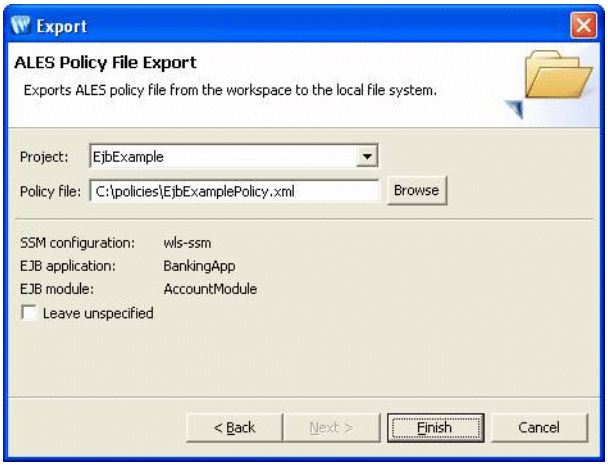
2. Select **ALES annotations** and configure the following four properties:
   - SSM configuration — the ALES SSM name we are going to bind the EJB application with; for example, `wls-ssm`.
   - EJB application — the name of the EJB application we are going to deploy; for example, `BankingApp`.
   -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

Export the ALES Policy File from Workshop

After annotating the project in the example, we can export the security policy file from Workshop. This security policy file can then be imported and deployed through ALES:

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

Figure 10-4 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 resulting policy file includes tokens for SSM configuration, EJB application, and EJB module instead of the values you have 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 an ALES Annotations policy using policyIX

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

1. If you did not already load the admin policies, start the ALES Administration Server and load them by running the `install_ales_schema` script.

2. If, when you created the policy file, you checked **Leave unspecified**, your policy file includes tokens, rather than specific values for SSM configuration, EJB application, and EJB module. Before you import the policy file into ALES, you need to replace these tokens with their actual values. In `ALES_ADMIN_HOME/config/annotation_config.properties`, replace the following 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 you modify the `annotation_config.properties` file, run `ALES_ADMIN_HOME/bin/annotation_transform` to create the policy file with actual values substituted for the tokens. The created policy file will have the same name as the `exported.res.file` parameter with the extension `.import` appended.

3. After the ALES Administration Server has started, use the policyIX utility to import the policy file. For example:

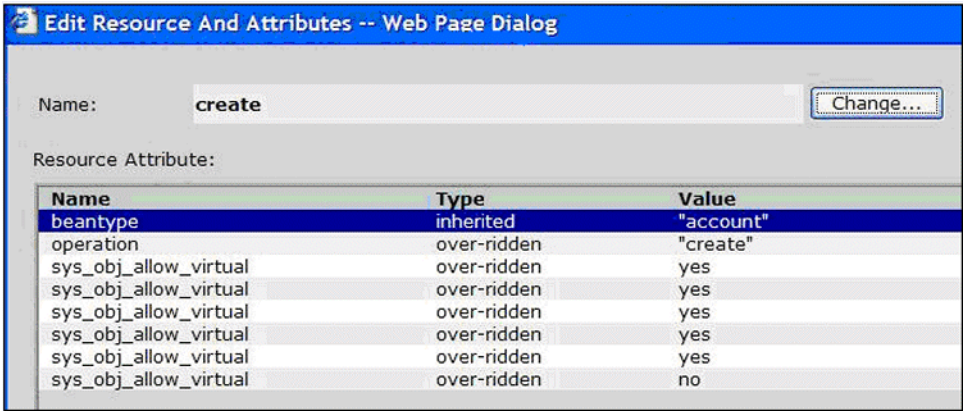
```
C:\ALES_ADMIN_HOME/bin> policyIX -import config.xml EjbExamplePolicy.xml
```

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

After you import the policy file into ALES, you can use the ALES Administration Console to view the protected 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 10-5 Resource Attributes for the create Method



Use the Resources Defined with ALES Annotations to Write Policies

Now that we have imported the policy file created in Workshop with ALES Annotations, we can define rules to access the resources defined with ALES annotations. For example, the following rule will grant 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 a user with the Customer role 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 Plug-in for Workshop

This section describes the ALES Tag Library for Workshop, which you can use to add ALES security functionality to your Java Server Pages (JSPs).

ALES Tag Library Overview

The ALES tag library allows you to easily secure JSP-page-level components (ALES resources) using ALES tags, and to retrieve information such as the set of roles a user has from the ALES security system.

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. Your 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.

ALES Tag Library Tags

The following tags are available in the ALES Tag Library for Workshop. See "ALES Tag Library Reference" on page 10-16 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 later 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 later to process responses.
- isAccessAllowedQueryResources – This tag 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 later 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 later for processing.
- isUserInRole – Returns true if the current user has a specific role.
- recordEvent – Passes an audit event into ALES. This tag is an event tag that provides input to the ALES auditing system.
- setSecurityContext – This 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.

ALES Tag Library Walk-Through

1. On the local system, in your JSP you determine the set of components that you want to protect. You can query the security system for data, such as which roles the user has.

   As a more elaborate usage, you could use query resources to return a collection that could populate a drop down list. An example might be the set of currencies to which a trader can convert.

2. On the local system, create your JSP pages. These JSP pages utilize the ALES tags to determine whether access is allowed to the components you developed in Step 1.

   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.

3. Use the ALES Administration Console to create policies for the resources you developed in Step 1. The fully qualified name (using the optional *<setSecurityContext-Value>)* for such a resource is

   `//app/policy/`*`<binding node>`*`/`*`<setSecurityContext-Value>`*`/resource`

4. Use the ALES Administration Console to distribute the policy data to the WLS 9.x SSM running on the local system.

5. Deploy your application on the local system.

6. 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.

Authenticated Subject is Determined by WebLogic Server

You do not need to supply an authenticated subject to the ALES tags for Workshop. This is because the container, WebLogic Server, determines and authenticates the subject and then makes the authenticated subject available for authorization decisions. You do not need to take any special action in your JSP page.

Example of Using ALES Tags in a JSP Page

This section provides an example of using ALES tags in a JSP page.

Consider the example JSP page shown in Listing 10-1:

Listing 10-1 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 10-1 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 is prepended with the SSM binding node at runtime. In the example, the resource URI is `testtagapp`.

Therefore, after adding the binding node and resource URI, at runtime the fully qualified resource in Listing 10-1 is `//app/policy/`*`<binding_node>`*`/testtagapp/isAllowed`.

After you have written the JSP in Listing 10-1 you would then:

1. Secure your container with an ALES SSM.
2. Use the ALES Administration Console to create policies for the resources on your 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 reasource to determine what users are allowed to see the secret text.

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.

Adding the ALES Tag Library to Workshop

This section describes how to integrate the ALES Tags in Workshop.

Integration Prerequisites

Before you begin, you must ensure that the following prerequisites are satisfied:

Note: You can test your JSP page without having ALES deployed in your development environment. JSP pages with ALES tags do not fail to compile or run when a WLS SSM is not installed.

If you were to deploy the JSP page on an instance of WLS without ALES, the tags do not break. Instead, the security tags allow both the allow- and deny cases to display on the page, and data retrieval methods return empty results.

- The system that serves the JSP pages with the embedded ALES tags (assumed here to be the local machine) must have BEA Workshop for WebLogic Platform version 9.2 installed, and must use WebLogic Server 9.2 as the container.
- The WLS 9.x SSM must be installed on the local machine.
- You must have access to an ALES Administration Console on a version 2.6 ALES Administration Server on either the local machine or a remote machine. You need the Administration server in order to import the resource definitions and distribute the resulting policy to the WLS 9.x SSM.

Integrating the Tag Library with Workshop: Main Steps

The tag library is packaged in its own jar file, `alestags.jar`, which contains all of the tag library supporting classes.

The `alestags.jar` is itself packaged in the *BEA_HOME*\ales26-admin\lib\eclipsePlugins\com.bea.wlw.ales.tags_9.2.0.jar file.

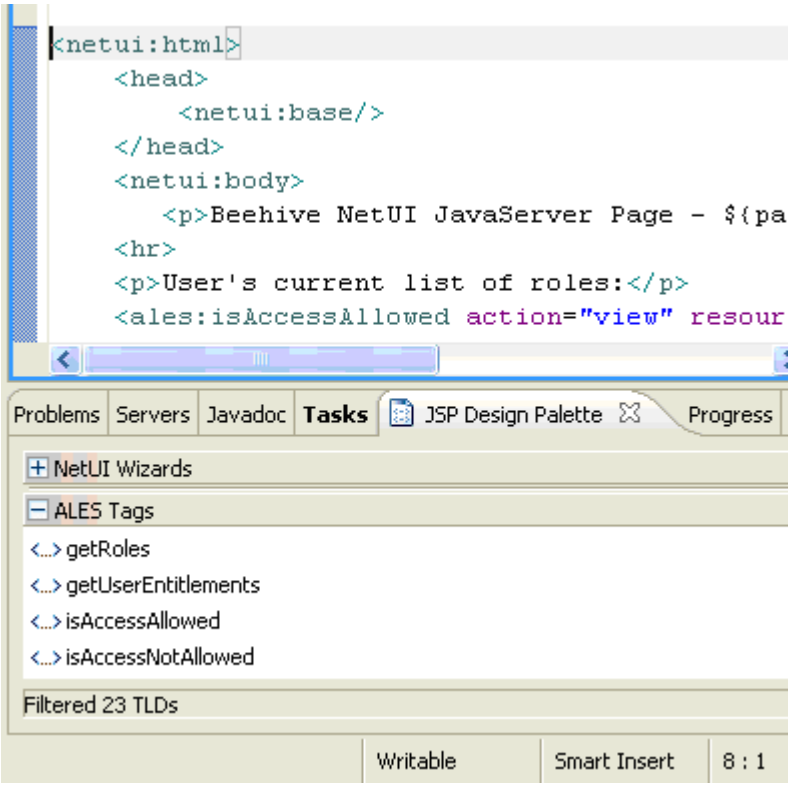
Follow these steps to integrate the tag library JAR file:

1. Copy *BEA_HOME*\ales26-admin\lib\eclipsePlugins\com.bea.wlw.ales.tags_9.2.0.jar to your *BEA_HOME*\workshop92\workshop4WP\eclipse\plugins directory.
2. Start or restart Workshop.
3. If creating a new project:
   a. Create a new dynamic web project.
   b. Name your new project and click Next.
   c. Check the ALES Tag Support project facet.
   d. Click Finish.
   e. Click Window->Show View->JSP Design Palette to display the JSP Design Palette.

   You should now have `alestags.jar` file in your `web-inf/lib` directory. In addition, in your JSP Design Palette you should see the ALES Tags.
4. If adding to an existing project:
   a. Right click on the project.
   b. Click properties.
   c. Click the project facets properties in the left navigation bar of the popup.
   d. Click add/remove facets.
   e. Check the ALES Tag Support project facet.
   f. Click finish.
   g. Click Window->Show View->JSP Design Palette to display the JSP Design Palette.

   You should now have `alestags.jar` in your `web-inf/lib` directory. In addition, in your JSP Design Palette you should see the ALES tags, as shown in Figure 10-6.

Figure 10-6 Workshop Integration Screen



Using Tag Resources in Your ALES Policy Definitions

1. You typically define the resources for your application as a part of defining the ALES tags. Take note of these resource names and the actions you want to perform on them.

   The parameter you pass 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 is 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.

2. Use the resources and actions you specified in the tags to create your ALES policy definitions.

   When you then create ALES policies to manage these resources, be aware that the fully qualified name of the resource is

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

How to Write Policies That Return Response Attributes

There is nothing unique about the ALES policies that you create to 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, if you use result and response attributes such as `isAccessAllowed.resultVar`, 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.

The JSP Standard Tag Library (JSTL) provides a set of core functionality common to most web applications. This functionality includes 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, in your JSP file you could use a JSTL tag in the following way:

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

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 see 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 10-1 isAccessAllowed

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

isAccessAllowed Concepts

Listing 10-2 shows the concepts of using `isAccessAllowed` in a JSP.

Listing 10-2 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 see 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 10-2 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

Table 10-2 isAccessNotAllowed

Attribute	Return Type	Description	Required
responseVar	Collection of Strings	The name of the variable used for returning responses from calling isAccessAllowed	No
responseVarScope	n/a	The scope of the variable containing responses from is access allowed (page, request, session, or application). The default scope is page.	No

isAccessNotAllowed Concepts

Listing 10-3 shows the concepts of using `isAccessNotAllowed` in a JSP.

Listing 10-3 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 10-3 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 10-4 shows the concepts of using `isAccessAllowedQueryResources` in a JSP.

Listing 10-4 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 10-4 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

Table 10-4 getUser Roles

Attribute	Return Type	Description	Required
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 10-5 shows the concepts of using `getUserRoles` in a JSP.

Listing 10-5 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 10-28, the body of the tag will be processed. In addition. the `resultVar` can be used for later processing.

Table 10-5 isUserInRole

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

isUserInRole Concepts

Listing 10-6 shows the concepts of using `getUserRoles` in a JSP.

Listing 10-6 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 10-6 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 10-7 shows the concepts of using setSecurityContext in a JSP.

Listing 10-7 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 10-28. All attributes are added to the audit context as additional information for the audit event.

Table 10-7 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 10-8 shows the concepts of using `recordEvent` in a JSP.

Listing 10-8 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 10-8 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 10-9 shows the concepts of using attribute in a JSP.

Listing 10-9 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 11

Integrating with WebLogic Portal

This section covers the following topics:

Introduction

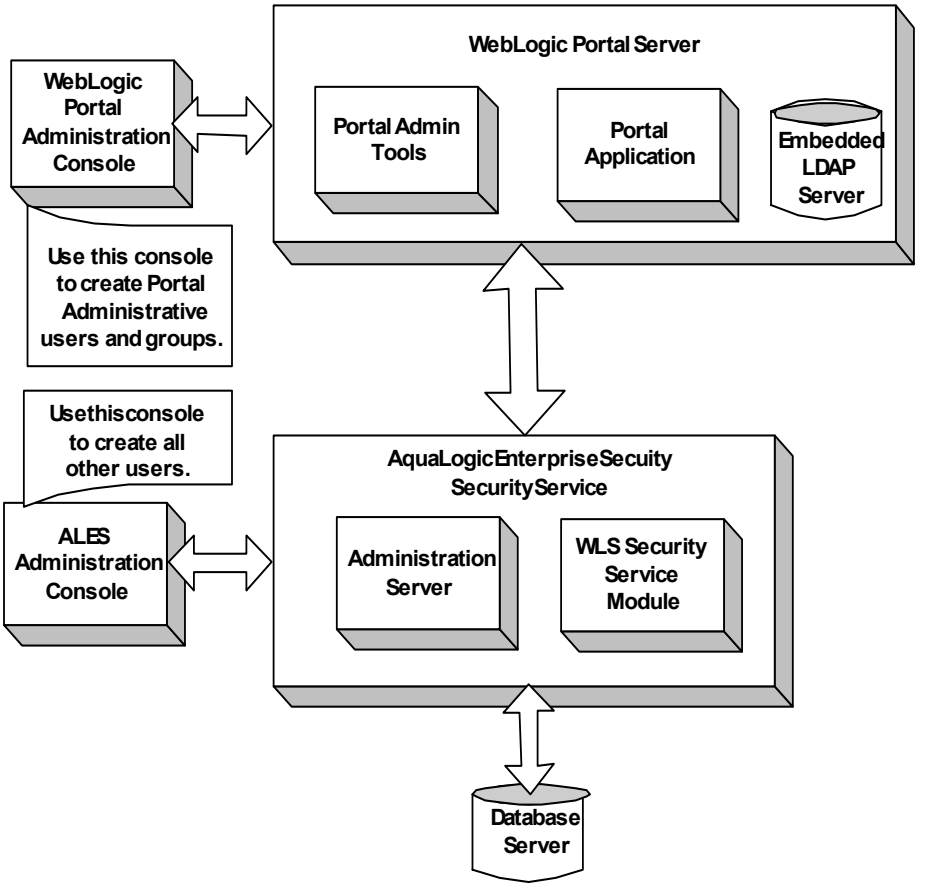
Integrating AquaLogic Enterprise Security with WebLogic Portal server and portal application results in an enhanced set of security services for use in protecting WebLogic Portal (see Figure 11-1). AquaLogic Enterprise Security participates in the authoring and management of policy for WebLogic Portal resources. Once AquaLogic Enterprise Security is integrated with WebLogic Portal, you use AquaLogic Enterprise Security Administration Server to manage resources related to portal desktops, books, pages, and portlets.

Therefore, the intent is that you use AquaLogic Enterprise Security for authorization of the resources associated with a portal application as well as standard WebLogic Server J2EE resources. The benefit of using AquaLogic Enterprise Security to manage visitor entitlements is that it offers fine-grained, dynamic role-based authorization. Additionally, AquaLogic Enterprise Security allows you to have common security policies for a heterogeneous environment. For

example, you may have a single security infrastructure that supports WebLogic Portal, WebLogic Server, and custom applications.

The AquaLogic Enterprise Security security service does not replace all of the management functionality provided by the Portal Administration Tools. For example, as shown in Figure 11-1, AquaLogic Enterprise Security is not used to manage administrative users and resources associated with Portal Delegated Administration and Portal Content Management; use the Portal Administration Tools for those management tasks.

Figure 11-1 Portal Integration Overview



AquaLogic Enterprise Security enables you to write, deploy, and manage fine-grained policy for controlling access to WebLogic Portal application resources. You can use AquaLogic Enterprise Security to protect portal desktops, books, pages, portlets, and application look and feels.

For more information, see the following topics:

Integration Features

AquaLogic Enterprise Security and WebLogic Portal can be used with WebLogic Server 9.2 or WebLogic Server 8.1 Service Pack 4 or Service Pack 5. While several different security providers can be used with WebLogic Portal, the following security providers include enhanced WebLogic Portal support:

- The ASI Authorization provider—Enables you to use AquaLogic Enterprise Security to write, deploy, and manage fine-grained authorization of WebLogic Portal application resources related to desktops, books, pages, portlets, and application look and feels.
- The Database Authentication provider—Enables user and group controls. This provider is integrated into the Portal Administration Tools environment so as to handle user and group queries.

  Note: Use of the Database Authentication provider with WebLogic Portal is not mandatory. You may use other authentication providers as well.

- ASI Role Mapper—Enables dynamic mapping of principals (users and groups) to security roles at runtime. Role Mapping determines which security roles to apply to the principals stored in a subject when the subject is attempting to perform an operation on a portal application resource. Because this operation usually involves gaining access to the portal application resource, the ASI Role Mapper is used in conjunction with the ASI Authorization provider.

Supported Use-case Scenario

The following use-case scenario is supported when you integrate AquaLogic Enterprise Security with WebLogic Portal:

- The AquaLogic Enterprise Security Administration Server assumes responsibility for management and policy of resources related to Portal visitor entitlements.
- The AquaLogic Enterprise Security Administration Server is responsible for management of J2EE application resources associated with Portal applications and portal administration tools.
- The Portal Administration Tools continue to be responsible for the rest of Portal Management and Administration, including the creation and management of portal administrative users and groups.

Note: To implement this use case scenario, you must define the security configuration as specified in "Creating the Portal Application Security Configuration" on page 11-9.

Constraints and Limitations

When integrated with AquaLogic Enterprise Security, WebLogic Portal has the following constraints and limitations:

- Portal application administrators will not use the WebLogic Portal Administration Tools to create and manage visitor entitlements.

  Use of AquaLogic Enterprise Security with a Portal application implies that an administrator will not use the Portal Administration Tools to create "Visitor Entitlements" on portal desktops, books, pages, and portlets. Managing visitor entitlements from the Portal Administration Tools is not a supported use case.

- Users will not use application deployment descriptors to deploy policy.

  Use of Deployment descriptors to deploy policy is not supported in AquaLogic Enterprise Security.

- Migration of existing portal application policy is not supported.

  AquaLogic Enterprise Security 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.

- AquaLogic Enterprise Security 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.

  You cannot use AquaLogic Enterprise Security to manage the resources associated with Portal Delegated Administration and Portal Content Management. AquaLogic Enterprise Security does not support Portal Unified User Profiles.

Integration Pre-Requisites

Before you begin, you must ensure that the following pre-requisites are satisfied:

- The WebLogic Platform/Portal 9.2 or 8.1, with Service Pack 4 or Service Pack 5, must be installed on the local machine.
- The AquaLogic Enterprise Security 2.6 WebLogic Server Security Service Module must be installed on the local machine.
- You must have access to an Administration Console that is running on the AquaLogic Enterprise Security 2.6 Administration Server on either the local machine or a remote machine.
- When using Weblogic Platform/Portal 9.2, you must have access to the WebLogic Server Administration Console, as well as the ALES Administration Console, to set the security configuration. For Weblogic Platform/Portal 8.1, the security configuration is set using the ALES Administration Console.
- You have created a WebLogic Portal domain on the local machine and installed a portal application in that domain.

Integrating with WebLogic Portal 9.2: Main Steps

This section describes how to integrate AquaLogic Enterprise Security with WebLogic Portal 9.2. The procedure for integrating ALES with WebLogic Portal 8.1 is different; for information about that, see "Integrating with WebLogic Portal 8.1: Main Steps" on page 11-8. Once integrated, you can use the AquaLogic Enterprise Security Administration Console to write and deploy a set of authorization and role mapping policies to protect WebLogic Portal application resources.

AquaLogic Enterprise Security includes an example that demonstrates how to secure a sample WebLogic Portal 9.2 domain with ALES. Follow the instructions in the example's README file, which is located at `BEA-HOME/ales22-ssm/wls9-ssm/examples/ALESEnabledWLP92Domain/README`. It is strongly recommended that you try the example once before following this procedure.

To integrate AquaLogic Enterprise Security with WebLogic Portal 9.2, perform the following tasks:

- "Creating the Portal Application Security Configuration" on page 11-9
- "Using the WebLogic Server Console to Configure Security Providers" on page 11-6

- "Modifying the Portal Server startWeblogic File" on page 11-7

Creating the Portal Application Security Configuration

This section describes how to create a new security configuration named `myrealm`.

1. Using the AquaLogic Enterprise Security Administration Console, create a security configuration with the Configuration ID `myrealm`.
2. Create instances of the ASI Authorization and ASI Role Mapper providers. Remember or record the setting for these two providers; you will need the information for a later step.
3. Bind the security configuration to a Service Control Manager.
4. Distribute the security configuration.
5. Create an instance of a WebLogic Server 9.x Security Service Module, as described in Creating an Instance of a Security Service Module in *Installing Security Service Modules*. Set the instance name to `portalInstance` and the Configuration ID to `myrealm`.
6. Enroll the WebLogic Server 9.x Security Service Module instance, as described in Enrolling the Instance of the Security Service Module in *Installing Security Service Modules*.
7. Use the WebLogic Server Administration Console to finish the security configuration for your WebLogic 9.x SSM. See "Using the WebLogic Server Console to Configure Security Providers" on page 11-6.

Using the WebLogic Server Console to Configure Security Providers

To configure security providers for the WebLogic Server 9.x Security Service Module, you use the WebLogic Server Administration Console, not the ALES Administration Console. In order to create and configure ALES security provider instances using the WebLogic Server Administration Console, you must first install an extension to the console. See "Modifying the startWebLogic File" on page 8-12.

Your security realm must include the XACML Authorizer and XACML RoleMapper security providers and they must each be the first providers of their type in the list. Also, the configuration settings for the ASI Authorization and ASI Role Mapper providers must match the configuration of these providers in the ALES Administration Console. For more information about how to configure security providers for ALES and WebLogic Portal 9.2, see "Configuring Security Providers for the WebLogic Server 9.x SSM" on page 8-15.

Modifying the Portal Server startWeblogic File

Before you can start a WebLogic Portal server that uses BEA AquaLogic Enterprise Security, you must modify the `startWeblogic` file that is located in the WebLogic Portal domain that you are using for you WebLogic Portal server.

The `startWeblogic` file for the WebLogic Portal sample domain named `portalDomain` is located at: `BEA_HOME\weblogic92\samples\domains\portal`

To edit the `startWeblogic` file, perform the steps:

Note: This procedure assumes a Windows installation of WebLogic Portal in the directory `c:\bea` with a WebLogic Server Security Service Module instance named `portalInstance`.

1. Before you modify the script, make sure to make a backup copy. For example, for Microsoft Windows, copy `startWeblogic.cmd` to `startWeblogic.cmd.original`.
2. Add a line to call the environment batch file `set-wls-env.bat`. For example, add it below the line: `set SAVE_JAVA_OPTIONS=`

   ```
   call
   "c:\bea\ales26-ssm\wls9-ssm\instance\portalInstance\bin\set-wls-env.bat
   "
   ```

3. Add the AquaLogic Enterprise Security `classpath` variables to the `classpath`. For example, add the following text before the line: `echo CLASSPATH=%CLASSPATH%`

   ```
   set CLASSPATH=%CLASSPATH%;%WLES_POST_CLASSPATH%
   ```

4. Add `%WLES_JAVA_OPTIONS%` to the server start command after `%JAVA_OPTIONS%`. Listing 11-2 shows, in bold text, where to make this change.

Listing 11-1 Adding WLES_JAVA_OPTIONS to the startWebLogic File

```
if "%WLS_REDIRECT_LOG%"=="" (
      echo Starting WLS with line:
      echo %JAVA_HOME%\bin\java %JAVA_VM% %MEM_ARGS% %JAVA_OPTIONS%
%WLES_JAVA_OPTIONS% -Dweblogic.Name=%SERVER_NAME%
-Djava.security.policy=%WL_HOME%\server\lib\weblogic.policy
%PROXY_SETTINGS% %SERVER_CLASS%
      %JAVA_HOME%\bin\java %JAVA_VM% %MEM_ARGS% %JAVA_OPTIONS%
%WLES_JAVA_OPTIONS% -Dweblogic.Name=%SERVER_NAME%
```

```
-Djava.security.policy=%WL_HOME%\server\lib\weblogic.policy
%PROXY_SETTINGS% %SERVER_CLASS%
) else (
      echo Redirecting output from WLS window to %WLS_REDIRECT_LOG%
      %JAVA_HOME%\bin\java %JAVA_VM% %MEM_ARGS% %JAVA_OPTIONS%
%WLES_JAVA_OPTIONS% -Dweblogic.Name=%SERVER_NAME%
-Djava.security.policy=%WL_HOME%\server\lib\weblogic.policy
%PROXY_SETTINGS% %SERVER_CLASS%  > "%WLS_REDIRECT_LOG%" 2>&1
)
```

After you have completed the procedures in this section, proceed to "Configuring Policy for the Portal Application" on page 11-14.

Integrating with WebLogic Portal 8.1: Main Steps

This section describes how to integrate AquaLogic Enterprise Security with WebLogic Portal 8.1. Once integrated, you can use the AquaLogic Enterprise Security Administration Console to write and deploy a set of authorization and role mapping policies to protect WebLogic Portal application resources.

Note: While the instructions provided in this section use a WebLogic Portal server and the sample portal application that ships with the WebLogic Platform software distribution, you can use this procedure to integrate AquaLogic Enterprise Security with your WebLogic Portal server and portal application.

To integrate AquaLogic Enterprise Security with WebLogic Portal, perform the following tasks:

- "Creating the Portal Application Security Configuration" on page 11-9
- "Binding the Security Configuration" on page 11-10
- "Distributing the Security Configuration" on page 11-10
- "Creating an Instance of the Security Service Module" on page 11-10
- "Enrolling the Instance of the Security Service Module" on page 11-10
- "Modifying the Portal Server startWeblogic File" on page 11-11
- "Creating the security.properties File" on page 11-12
- "Replacing the Portal p13n_ejb.jar File" on page 11-12
- "Replacing the Portal p13n_system.jar File" on page 11-13

- "Replacing the DefaultAuthorizerInit.ldift File" on page 11-14

Creating the Portal Application Security Configuration

This section describes how to create a new security configuration named `myrealm`. A security configuration defines the set of security providers to use for adjudication, authentication, auditing, authorization, role mapping, and credential mapping services. The security configuration named `myrealm` matches the default security configuration for the WebLogic Portal sample portal application.

Note: To implement the use-case scenario described in "Supported Use-case Scenario" on page 11-3, you are required to define the security configuration as described in this section. This security configuration is a requirement; it is not optional.

Refer to Table 11-1 and use the AquaLogic Enterprise Security Administration Server to configure the security providers listed there. Set the Configuration ID to myrealm. For instructions on creating a security configuration, see the administration console's help system

Table 11-1 Portal Security Configuration

Security Provider	Configuration Settings
ASI Adjudication Provider	Uncheck the Require Unanimous Permit check box, and click Create.
Log4j Auditor	Accept the default settings, and click Create.
Database Authentication Provider	Set the Control Flag to `SUFFICIENT`, and click Create. For the Details tab settings, except for the Identity Scope, the parameters are populated automatically. Set the Identity Scope to `myusers`, and click Apply. **Note:** Even though you set the Identity Scope to `myusers`, you do not actually create the `myusers` identity until you perform the steps in "Creating the Realm Resource" on page 11-16.
WebLogic Authentication Provider	Set the Control Flag to `SUFFICIENT`, and click Create. **Note:** Make sure the authentication providers are configured in the following order: 1) Database Authenticator and 2) WebLogic Authenticator. **Note:** The WebLogic Authentication provider can be replaced with another authentication provider that supports write access to users and groups.

Table 11-1 Portal Security Configuration (Continued)

Security Provider	Configuration Settings
ASI Authorization Provider	On the General tab, accept the default settings, and click Create. On the Details tab, set the Identity Scope to `myusers`.
WebLogic Authorization Provider	Uncheck the Policy Deployment Enabled check box, and click Create.
WebLogic Credential Mapper Provider	Uncheck the Credential Mapping Deployment Enabled check box, and click Create.
ASI Role Mapping Provider	On the General tab, accept the default settings, and click Create. On the Details tab, set the Identity Scope to `myusers`.
WebLogic Role Mapper Provider	Uncheck the Role Deployment Enabled check box, and click Create.

Binding the Security Configuration

The security configuration must be bound to a Service Control Manager.

To bind the `myrealm` security configuration, see the Console Help

Distributing the Security Configuration

The `myrealm` security configuration must be distributed.

To distribute the `myrealm` security configuration, see the Console Help.

Creating an Instance of the Security Service Module

Before starting a WebLogic Server Security Service Module, you must first create an instance of the WebLogic Server Security Service Module using the Create New Instance Wizard.

To create an instance of a WebLogic Server Security Service Module, see Creating an Instance of a Security Service Module in *Installing Security Service Modules*.

Enrolling the Instance of the Security Service Module

You must have the Administration Server running prior to enrolling the Security Service Module.

Note: While you can use the demonstration digital certificate in a development environment, you should never use it in a production environment.

To enroll a security service module, see Enrolling the Instance of the Security Service Module in *Installing Security Service Modules*.

Modifying the Portal Server startWeblogic File

Before you can start a WebLogic Portal server that uses BEA AquaLogic Enterprise Security, you must modify the `startWeblogic` file that is located in the WebLogic Portal domain that you are using for you WebLogic Portal server.

The `startWeblogic` file for the WebLogic Portal domain named `portalDomain` is located at: `BEA_HOME\user_projects\domains\portalDomain`

To edit the `startWeblogic` file, perform the steps:

Note: This procedure assumes a Windows installation of WebLogic Portal in the directory `c:\bea` with an WebLogic Server Security Service Module instance named `portalInstance`.

1. Before you modify the script, make sure to make a backup copy. For example, for Microsoft Windows, copy `startWeblogic.cmd` to `startWeblogic.cmd.original`.
2. Add a line to call the environment batch file `set-wls-env.bat`. For example, add it below the line: `set SAVE_JAVA_OPTIONS=`

   ```
   call
   "c:\bea\ales26-ssm\wls-ssm\instance\portalInstance\bin\set-wls-env.bat"
   ```

3. Add the AquaLogic Enterprise Security `classpath` variables to the `classpath`. For example, add the following text before the line: `echo CLASSPATH=%CLASSPATH%`

   ```
   set CLASSPATH=%WLES_PRE_CLASSPATH%;%CLASSPATH%;%WLES_POST_CLASSPATH%
   ```

4. Add `%WLES_JAVA_OPTIONS%` to the server start command after `%JAVA_OPTIONS%`. Listing 11-2 shows, in bold text, where to make this change.

Listing 11-2 Adding WLES_JAVA_OPTIONS to the startWebLogic File

```
if "%WLS_REDIRECT_LOG%"=="" (
      echo Starting WLS with line:
      echo %JAVA_HOME%\bin\java %JAVA_VM% %MEM_ARGS% %JAVA_OPTIONS%
%WLES_JAVA_OPTIONS% -Dweblogic.Name=%SERVER_NAME%
-Djava.security.policy=%WL_HOME%\server\lib\weblogic.policy
%PROXY_SETTINGS% %SERVER_CLASS%
```

```
      %JAVA_HOME%\bin\java %JAVA_VM% %MEM_ARGS% %JAVA_OPTIONS%
%WLES_JAVA_OPTIONS% -Dweblogic.Name=%SERVER_NAME%
-Djava.security.policy=%WL_HOME%\server\lib\weblogic.policy
%PROXY_SETTINGS% %SERVER_CLASS%
) else (
      echo Redirecting output from WLS window to %WLS_REDIRECT_LOG%
      %JAVA_HOME%\bin\java %JAVA_VM% %MEM_ARGS% %JAVA_OPTIONS%
%WLES_JAVA_OPTIONS% -Dweblogic.Name=%SERVER_NAME%
-Djava.security.policy=%WL_HOME%\server\lib\weblogic.policy
%PROXY_SETTINGS% %SERVER_CLASS%  > "%WLS_REDIRECT_LOG%" 2>&1
)
```

Creating the security.properties File

Create a text file named `security.properties` and place it in the portal domain directory. You use this file to define the AquaLogic Enterprise Security realm and the default realm. Listing 11-3 shows the content of this file for the realm `myrealm`.

Listing 11-3 security.properties File

```
# AquaLogic Enterprise Security Configuration File
#
# This file contains AquaLogic Enterprise Security configuration
# properties. By default, the AquaLogic Enterprise Security runtime
# looks for a property file called 'security.properties' in the
# working directory.
wles.realm=myrealm
wles.default.realm=myrealm
```

Replacing the Portal p13n_ejb.jar File

To integrate AquaLogic Enterprise Security with WebLogic Portal, you must replace the `p13n_ejb.jar` file in the top-level portal application directory with the version of that file that is provided in the AquaLogic Enterprise Security software distribution. The AquaLogic Enterprise Security version of `p13n_ejb.jar` is located in `BEA_HOME/ales26-ssm/wls-ssm/lib` directory.

Note: BEA AquaLogic Enterprise Security 2.6 includes two versions of `p13n_ejb.jar`, the WebLogic Server 8.1 SP4 version: `p13n_ejb_81SP4.jar`, and the SP5 version: `p13n_ejb_81SP5.jar`. Be sure to use the correct version.

Note: Because these instructions assume that you are using the sample portal application that ships with WebLogic Portal, this procedure instructs you to replace the `p13n_ejb.jar` in the sample portal application. To use a different portal application, replace `p13n_ejb.jar` in that application as well.

To replace `p13n_ejb.jar`, perform the following steps:

1. Rename the portal version of the `p13n_ejb.jar`. For example, rename it to `p13n_ejb.jar.original`. The portal application version of this file is located in `BEA_HOME/weblogic81/samples/portal/portalApp`.
2. Depending on which version of the WebLogic Server 8.1 you are using (SP4 or SP5), copy either `p13n_ejb_81SP4.jar` or `p13n_ejb_81SP5.jar` from `BEA_HOME/ales26-ssm/wls-ssm/lib/` to `BEA_HOME/weblogic81/samples/portal/portalApp` and `rename` it to `p13n_ejb.jar`.

Replacing the Portal p13n_system.jar File

To integrate AquaLogic Enterprise Security with WebLogic Portal, you must replace the `p13n_system.jar` file in the `BEA_HOME/weblogic81/p13n/lib` directory with the version of that file that is provided in the AquaLogic Enterprise Security software distribution. The AquaLogic Enterprise Security version of this file is located in `BEA_HOME/ales26-ssm/wls-ssm/lib` directory.

Note: BEA AquaLogic Enterprise Security 2.6 includes two versions of `p13n_system.jar`, the WebLogic Server 8.1 SP4 version: `p13n_system_81SP4.jar`, and the SP5 version: `p13n_system_81SP5.jar`. Be sure to use the correct version.

Note: Once you replace `p13n_system.jar` in the `/lib` directory of the WebLogic Platform installation, all portal domains configured for that installation must be AquaLogic Enterprise Security enabled.

To replace `p13n_system.jar`, perform the following steps:

1. Rename the portal version of the `p13n_system.jar`. For example, rename it to `p13n_system.jar.original`. The portal version of this file is located in `BEA_HOME/weblogic81/p13n/lib`.
2. Depending on which version of the WebLogic Server 8.1 you are using (SP4 or SP5), copy either `p13n_system_81SP4.jar` or `p13n_system_81SP5.jar` from

BEA_HOME/ales26-ssm/wls-ssm/lib/ to BEA_HOME/weblogic81/p13n/lib and rename it to p13n_system.jar.

Replacing the DefaultAuthorizerInit.ldift File

WebLogic Server uses the DefaultAuthorizerInit.ldift file to establish access controls for J2EE resources. By default, WebLogic Server allows access to all J2EE resources to users in the Everyone role. To protect these resources, WebLogic Server provides the Administration Console and other tools to define security policies.

When using AquaLogic Enterprise Security, there is a need to supersede the WebLogic Server J2EE access controls. The DefaultAuthorizerInit.ldift file, provided in the AquaLogic Enterprise Security 2.6 for the WebLogic Server Security Service Module, is used for this purpose.

To enable the AquaLogic Enterprise Security DefaultAuthorizerInit.ldift file to supersede WebLogic Server access controls for J2EE resources in the sample portal application, perform the following steps:

1. Copy the DefaultAuthorizer.ldift file from BEA_HOME/ales26-ssm\wls-ssm\template\config to BEA_HOME/user_projects/domains/mydomain.
2. If the /ldap directory exists at the following location, delete it: BEA_HOME/user_projects/domains/portalDomain/portalServer

Note: Because these instructions assume that you are using the sample portal domain that ships with WebLogic Portal, this procedure instructs you to delete the ldap directory in the portalDomain/portalServer directory. Repeat the above steps for all AquaLogic Enterprise Security enabled portal domains.

Configuring Policy for the Portal Application

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 portal application resources.

AquaLogic Enterprise Security provides two means for writing portal application policy, the Administration Console and the Policy Import Tool. In this section you are directed to use the

Administration Console to write policy. For more information on how to use the Administration Console to write policy, see the *Policy Managers Guide* and the Console Help.

In addition, the ALES Administration Server installation includes a set of sample polices for BEA WebLogic Portal, located at `BEA_HOME/ales26-admin/examples/policy/portal_sample_policy`. You can import these sample policies and use them as a starting point for developing a full set of policies for your applications. For information about how to import the sample policies, see the README files in each of the sample directories and see also Importing Policy Data in the *Policy Managers Guide*.

This section covers the following topics:

Creating the Identity Directory and Users

This section describes how to use the Administration Console to create an identity directory, groups, and users for a portal application.

Note: This procedure uses `myusers` as the name of the Identity directory; however, you can use a different name.

To create the Identity directory and users:

1. In the left pane, click Identity. The Identity page displays the name of each directory available.
2. Click New. The Create Directory dialog box appears.
3. In the Name text box, type `myusers` and click OK. The `myusers` directory appears in the list of Identity directories.
4. In the left pane, click Users. The myusers>Users page displays.
5. Click New. The Create User dialog box appears.
6. Create the users that will visit your portal application.

If you are using the WebLogic 9.x SSM, 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`.

Configuring Resources and Privilege

This section describes how to use the Administration Console to define the portal application resources that you will protect using AquaLogic Enterprise Security.

To configure resources, perform the following tasks:

- "Creating the Realm Resource" on page 11-16
- "Refer to Table 11-2 and modify the configuration of the ASI Authorization provider as described there." on page 11-17
- "Creating the Console Resources" on page 11-18
- "Creating the PortalApp Resources" on page 11-19

Creating the Realm Resource

Note: `myrealm` is used is this procedure as the realm name because the WebLogic Portal 8.1 sample portal application exists in the `myrealm` realm. You can choose any realm name for your portal application; however, if you are integrating WebLogic Portal 9.2 using the WebLogic Server 9.x SSM, the Configuration ID must match the name of the WebLogic Server security realm.

To create a realm resource, perform the following steps:

1. Expand the Resources folder, and click Resources. The Resource page displays.
2. In the Resources page, select the Policy node, and click New. The Create Resource dialog box appears.
3. In the Name text box, enter `myrealm`, select `Binding` from the Type drop-down list box, and click Ok. The `myrealm` resource appears under the Policy node.
4. Select the `myrealm` resource and click Configure. The Configure Resource dialog box appears.
5. From the Type drop-down list box, select Binding Application, check the Distribution Point and Allow Virtual Resources check boxes, and click Ok.

6. Refer to Table 11-2 and modify the configuration of the ASI Authorization provider as described there.

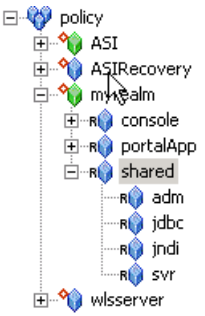
Table 11-2 Portal Security Configuration Modifications

Security Provider	Configuration Settings
ASI Authorization Provider	On the Details tab, set the Application Deployment Parent to `//app/policy/myrealm` and click Apply. On the Bindings tab, from the Bind drop-down menu, select `//app/policy/myrealm`, and click Bind.

Creating the Shared Resources

Figure 11-2 shows the shared resources that you must create.

Figure 11-2 Shared Resources



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

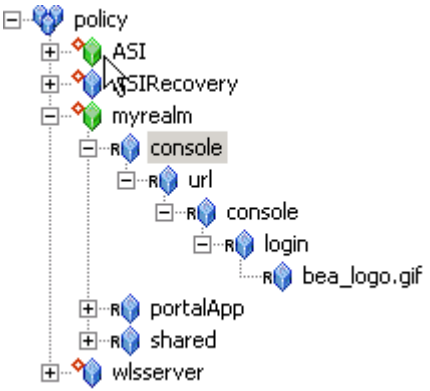
1. Select the `myrealm` resource and click New. The Create Resource dialog box appears.
2. In the Name text box, enter `shared`, and click Ok. The `shared` resource appears under `myrealm`.
3. Select the shared resource and click Configure. The Configure Resource dialog box appears.
4. Check the Allow Virtual Resources and click Ok.
5. Click the `shared` resource and click New. The Create Resource dialog box appears.
6. In the Name text box, enter `adm` and click Ok. The `adm` resource appears under the `shared` resource.

7. To configure the `jdbc`, `jndi`, and `svr` resources as shown in Figure 11-2, repeat steps 5 and 6 for each resource.

Creating the Console Resources

Figure 11-3 shows the console resources that you must create.

Figure 11-3 Console Resources



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

1. Select the `myrealm` resource and click New. The Create Resource dialog box appears.
2. For WebLogic Portal 9.2, in the Name text box, enter `consoleapp`, and click Ok. The `consoleapp` resource appears under `myrealm`.

   For WebLogic Portal 8.1, in the Name text box, enter `console`, and click Ok. The `console` resource appears under `myrealm`.
3. To create the `url`, `console`, `login`, `and bea_logo.gif` resources as shown in Figure 11-3, repeat steps 1 and 2 for each resource.
4. Select the `console` or `consoleapp` resource directly under `myrealm` and click Configure. The Configure Resource dialog box appears.
5. Check the Allow Virtual Resources and click Ok.

Creating the PortalApp Resources

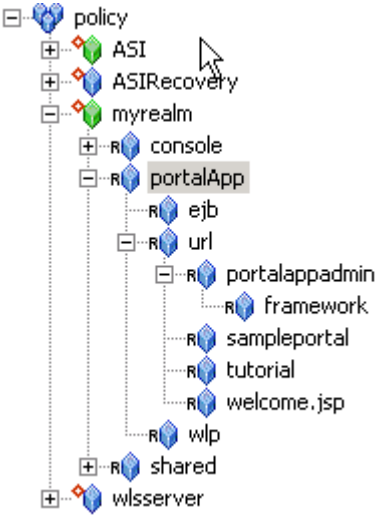
Note: This procedure uses `portalApp` as the name of the portal application resource because it is the name of the WebLogic Portal sample portal application. However, you should use the name of your portal application when creating the portal application resource.

Figure 11-4 shows the `portalApp` resources that you must create. In addition, if you are integrating WebLogic Portal 9.2 using the WebLogic Server 9.x SSM, you must create these virtual resources under `myrealm`:

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

Figure 11-4 PortalApp Resources

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



1. Select the `myrealm` resource and click New. The Create Resource dialog box appears.
2. In the Name text box, enter `portalApp`, and click Ok. The `portalApp` resource appears under `myrealm`.
3. Select the `portalApp` resource and click New. The Create Resource dialog box appears.

4. In the Name text box, enter `ejb` and click Ok. The `ejb` resource appears under the `portalApp` resource.
5. Select the `ejb` resource and click Configure. The Configure Resource dialog box appears.
6. Check the Allow Virtual Resources and click Ok.
7. To configure the `url` resource, repeat steps 5 and 6.
8. Select the `portalApp` resource and click New. The Create Resource dialog box appears.
9. In the Name text box, enter `wlp` and click Ok. The `wlp` resource appears under the `portalApp` resource. Do **not** configure the `wlp` resource to allow virtual resources.
10. Select the `url` resource and click New. The Create Resource dialog box appears.
11. In the Name text box, enter `portalappadmin` and click Ok. The `portalappadmin` resource appears under the `url`.
12. Repeat steps 10 and 11 to create the remaining resources shown under the `portalApp` resource in Figure 11-4. Do **not** configure any of the remaining resources to allow virtual resources.

Creating the Role Mapping Policy

This section describes how to use the Administration Console to create role mapping policy that will be used to control access to portal application resources.

Table 11-3 lists and describes the role mapping policy that you have to create for the WebLogic Portal domain.

To create the role mapping policy, refer to Table 11-3 and perform the following steps:

Table 11-3 Portal Application Role Mapping Policy

Role Mapping Policy	Description
`grant(//role/Everyone, //app/policy/myrealm, //sgrp/myusers/allusers/) if true;`	Creates the role mapping policy necessary for the Everyone role to be used in the `myrealm` Identity directory. **Note:** If you do not create the `Everyone` role mapping policy correctly, none of the policy rules defined in Table 11-4 that use the `Everyone` role will work properly.

Caution: If you do not create the `Everyone` role mapping policy correctly, none of the authorization policies defined in Table 11-4 and Table 11-5 that use the `Everyone` role will work properly.

1. Expand the Policy folder in the left pane, and click Role Mapping Policies. The Role Mapping Policies page appears.
2. Click New. The Create Role Mapping Policy dialog box appears.
3. Select the Grant radio button.
4. Select the Roles tab, select `Everyone` in the Available Roles list box, and click Add.
5. Select the Resources tab, select `myrealm`, and click Add.
6. Select the Policy Subjects tab, select the `allusers` in the list box, click Add, and click Ok.

Creating Authorization Policies

This section describes how to use the Administration Console to create authorization policies to protect portal application resources.

Table 11-4 lists and describes the authorization policies that you have to create for the WebLogic Portal domain to protect the sample portal application resources. In addition, Table 11-5 lists the authorization policies required for WebLogic Portal 9.2.

Table 11-4 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.

Table 11-4 Portal Application Authorization Policies (Continued)

Authorization Policies	Description
`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.
`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.

Perform the following steps create the authorization policies listed in Table 11-4 (and, for WebLogic Portal 9.2, Table 11-5).

1. Expand the Policy folder in the left pane, and click Authorization Policies. The Authorization Policies page appears.
2. Click New. The Create Authorization Policy dialog box appears.
3. Select the Grant radio button.
4. To create the first authorization policy listed in Table 11-4, click the Privileges tab, select the `any` privilege from the Select Privileges from Group list box, and click Add.

5. Click the Resources tab, select the `svr` resource from the Child Resource box and click Add.

   Note: If you want to assign multiple resources to a single privilege and role, you may define all of the resources in one authorization policy.

6. Click the Policy Subjects, select the `Admin` role from the Roles List box, click Add, and click Ok.

7. Repeat steps 4 to 6 for each of the remaining authorization policies.

Policy for Visitor Entitlements to Portal Resources

Visitor entitlements is a mechanism used by WebLogic Portal for determining who may access the resources in a portal application and what they may do with those resources. AquaLogic Enterprise Security provides a means of defining robust 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 11-5 shows the capabilities of each of these resources:

Table 11-5 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 11-5 are defined as follows:

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

The following topics provide information on how to use AquaLogic Enterprise Security to configure portal resources:

- "Configuring Policy for Desktops" on page 11-25
- "Table 11-6 shows an authorization policy that grants the `view` privilege to the `samplePortal` desktop for visitors in the SampleVisitor role." on page 11-25
- "Table 11-7 shows an authorization policy that grants the view privilege to the book_1 book for visitors in the SampleVisitor role." on page 11-26
- "Table 11-8 shows an authorization policy that grants the `view` privilege to the page_2 page for visitors in the SampleVisitor role" on page 11-26
- "Configuring Policy for Look and Feels" on page 11-27
- "Defining Policy for Portlets using Instance ID" on page 11-28

Configuring Policy for Desktops

A desktop is a view of the portal that the visitor accesses. There can be one or more desktops per portal, so the portal is effectively a container for the desktops. A Desktop is referenced as a resource in AquaLogic Enterprise Security in the following manner:

`//app/policy/`*`myrealm`*`/`*`portalapp`*`/wlp/`*`sampleportal`*`/com_bea_p13n/Desktop/`*`samplePortal`*

where:

- *`myrealm`* is the realm in which the portal application is installed
- *`portalapp`* is the portal application directory
- *`sampleportal`* is the name of the sample portal application
- *`samplePortal`* is the label definition of the desktop.

If you define an authorization policy at the `samplePortal` level, you can control access at the `samplePortal` desktop level.

Table 11-6 shows an authorization policy that grants the `view` privilege to the `samplePortal` desktop for visitors in the SampleVisitor role.

Table 11-6 SamplePortal Authorization Policy

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

Configuring Policy for Books

A book is a collection of pages. A book is referenced as a resource in AquaLogic Enterprise Security 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.

If you define an authorization policy at the `book_1` level, you can control access at the `book_1` book level.

Table 11-7 shows an authorization policy that grants the view privilege to the book_1 book for visitors in the SampleVisitor role.

Table 11-7 Book_1 Authorization Policy

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

Configuring Policy for Pages

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

`//app/policy/myrealm/portalapp/wlp/sampleportal/com_bea_p13n/Page/`*`page_2`*

where *`page_2`* is the label definition of the page.

If you define an authorization policy at the `page_2` level, you can control access at the page_2 page level.

Table 11-8 shows an authorization policy that grants the `view` privilege to the page_2 page for visitors in the SampleVisitor role

Table 11-8 Page_2 Authorization Policy

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

Configuring Policy for Portlets

Note: See Accessing a Portlet Requires Policy that Grants View to the com_bea_p13n Resource for important information about 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 AquaLogic Enterprise Security 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 would be `portlet_login_1`. This name is then mapped to `portlet_login` by the WebLogic Portal Resource Converter. `portlet_login` is used for the policy definition.)

If you define an authorization policy at the *`portlet_login`* level, you can control access at the *`portlet_login_1`* Portlet level.

Table 11-9 shows an authorization policy that grants the view privilege to the `portlet_login_1` Portlet for visitors in the SampleVisitor role.

Table 11-9 Portlet_login_1 Authorization Policy

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

Accessing a Portlet Requires Policy that Grants View to the com_bea_p13n Resource

You must have a policy that grants the View privilege to the `com_bea_p13n` resource or you will get authorization errors when accessing a portlet. However, if you grant the View privilege to the com_bea_p13n resource, all resources below it in the hierarchy are by default also granted View, which may not be appropriate in your application.

As described in Restricting Policy Inheritance, you can restrict policy inheritance by limiting its applicability. You can write the rule such that the lower resources will not get the View privilege given for `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;
```

Configuring Policy for Look and Feels

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

`//app/policy/myrealm/portalapp/wlp/sampleportal/com_bea_p13n/LookAndFeel/`*`textLookAndFeel`*

where *`textLookAndFeel`* 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 11-10 shows an authorization policy that grants the view privilege to the textLookAndFeel Look and Feel visitors in the SampleVisitor role.

Table 11-10 textLookAndFeel Look and Feel Authorization Policy

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

Defining Policy 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 AquaLogic Enterprise Security, 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 11-11 shows an authorization policy that grants the `view` privilege to the *`portlet_login_1`* Portlet for visitors in the Operator role.

Table 11-11 Portlet_login_1 Authorization Policy Using Instance ID

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

Discovering Portal Application Resources

When developing policies for use with a Security Service Module, you can use the discovery mode feature of the AquaLogic Enterprise Security Administration Server to help define your policy components. Instructions for using discovery mode are provided in the Resource Discovery section in the *Policy Managers Guide*.

Distributing Policy and Security Configuration

Distribute policy and security configuration to the WebLogic Server Security Service Module.

For information on how to distribute policy and security configuration, see the Console Help. Be sure to verify the results of your distribution.

Starting the WebLogic Portal Server

To start a WebLogic Portal server, perform the following steps:

1. Open a shell (command prompt) on the machine on which you created the portal domain.

2. Change to the portal domain directory:
   - For WebLogic Portal 9.2, `BEA_HOME\weblogic92\samples\domains\portal`.
   - For WebLogic Portal 8.1, `BEA_HOME\user_projects\domains\portalDomain`.
3. Run the following script:

   On Windows: `startWebLogic.cmd`

   On UNIX: `startWeblogic.sh`

Configuring Portal Administration to Use the WebLogic Authenticator

To use the WebLogic Authentication provider to manage administrative users for portal administration, perform the following steps:

1. Within the Portal Administration console, go to `Service Administration`.
2. Select `Authentication Hierarchy Service`.
3. Add `WebLogicAuthenticator` to the Authentication Providers to Build list.

Using Portal Administration Tools to Create a Portal Desktop

Before you can use AquaLogic Enterprise Security to control access to a portal desktop, you must use WebLogic Portal Administration Tools to create a portal desktop.

To create a portal desktops, perform the following steps:

1. To have full control of the definition and instance labels, use WebLogic Workshop to create the initial portal application.
2. Using the initial portal application as a template, use the Portal Administration Tools to create the portal desktops.

For instructions on using Portal Administration Tools to create portal desktops, see "Create a Desktop" in the WebLogic Administration Portal Online Help.

To create a portal desktop for the sample portal application using the Portal Administration Tools, perform the following steps:

1. Open a browser and point it to: `http://<hostname>:7001/testportalappAdmin`. The Sign In page appears.
2. Enter Username: `portaladmin`, Password: `portaladmin`, and click Sign In. The Portal Resources navigation tree appears.
3. Right click on Portals and select New Portal. The Portal Resource page appears.
4. Enter a Name for this Portal, for example: `myportal`, enter a partial URL for this Portal, for example: `myportal`, and click Save. The Available Portals list appears.
5. Select the `myportal` link from the list. The Editing Portal: `myportal` page appears.
6. Click Create New Desktop. The New Desktop Properties dialog appears.
7. Enter Title, for example: `mydesktop` and a Partial URL, for example: `mydesktop`.
8. Choose a Template. Choose: `testportalweb/test.portal` and click Create New Desktop. Desktops contained in Portal `myportal` list appear.
9. Select `mydesktop` from the list. The Editing Desktop: `mydesktop` page appears.
10. Select View Desktop. `mydesktop` is displayed in a new browser window. Verify that the desktop contains the sample portal application.
11. Close the mydesktop browser and Logout. The Sign In page appears.
12. Exit the Portal Administration Tool by closing its browser.

Accessing the Portal Application

To access a portal application running on a portal server, open a browser and point it to the desktop URL. For example, if you set up the desktop for the sample portal application as described in "Using Portal Administration Tools to Create a Portal Desktop" on page 11-29, you can access the sample portal application using the following URL:

`http://<host_name>:7001/sampleportal/appmanager/myportal/mydesktop`

where <*host_name*> is the machine on which portal application is running.

CHAPTER 12

Integrating with AquaLogic Data Services Platform

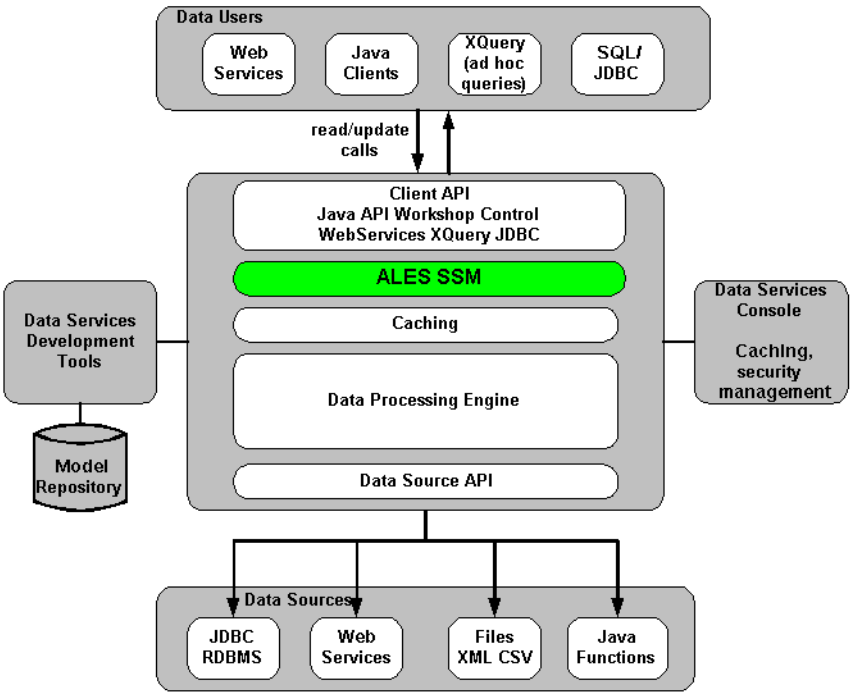
This section describes how to integrate AquaLogic Enterprise Security with AquaLogic Data Services Platform, using the WebLogic Server 8.1 SSM or WebLogic Server 9.x SSM. It includes the following topics:

Introduction

AquaLogic Enterprise Security (ALES) can provide fine-grained entitlements for Data Services serviced by AquaLogic Data Services Platform (ALDSP) 2.1. AquaLogic Enterprise Security can be used to manage access control to entire services or elements of those services. AquaLogic Enterprise Security allows you to have common set of security policies for a heterogeneous environment, and a single security infrastructure that supports WebLogic Portal, WebLogic Server, and custom applications.

The ALES service does not replace all of the management functionality provided by the ALDSP. The ALDSP Administrative console (ldconsole) is still used to manage all of the attributes of the various data services aggregated by ALDSP (see Figure 12-1).

Figure 12-1 AIDSP Integration Overview



The AquaLogic Enterprise Security WebLogic SSM enables you to write, deploy, and manage fine-grained policy for controlling access to all WebLogic server application resources, including data services. A specific resource type `ld` allows a security administrator to represent the data services in the ALES resource hierarchy. Elements of that data service are also converted to the ALES format for evaluation by the ASI authorization engine.

For more information, see the following topics:

- "Integration Features" on page 12-3
- "Supported Use-case Scenario" on page 12-3
- "Constraints and Limitations" on page 12-3

Integration Features

AquaLogic Data Service Platform (ALDSP) 2.1 requires WebLogic Server 9.2, 9.1, or 8.1 (with Service Pack 4 or 5) and uses the WLS 9.x or WLS 8.1 SSM. While the ALES framework allows for different security providers to be used with ALDSP, the following providers were certified:

- The ASI Authorization and Role Mapping providers enable you to use AquaLogic Enterprise Security to write, deploy, and manage fine-grained authorization of Data Services.
- The Database Authentication provider performs authentication services for the SSM.

Supported Use-case Scenario

The following use-case scenario is supported when you integrate AquaLogic Enterprise Security with AquaLogic Data Services Platform:

- The AquaLogic Enterprise Security Administration Server assumes responsibility for management and policy of data services and elements of those services through the ALES console or Policy Management API.
- The AquaLogic Enterprise Security Administration Server is responsible for access control of J2EE applications deployed on the ALDSP WebLogic Server.
- The ALDSP Administration Console continues to be the management point for data services.

Constraints and Limitations

AquaLogic Enterprise Security integration with ALDSP has the following constraints and limitations:

- Data service elements must be enabled for security through the ALDSP Admin console before ALES can manage element-based access control.
- ALES cannot provide entitlements and control which records are returned by the data service. This can still be done, but must be performed through the ALDSP Admin console.

Integration Pre-Requisites

Before you begin, you must ensure that the following pre-requisites are satisfied:

- ALDSP 2.5 patch for ALES (must be requested from BEA Support)

- WebLogic Server 9.1 or 9.2, or 8.1 with Service Pack 4 or 5
- WebLogic Server 8.1 or 9.x Security Service Module
- You must have access to an ALES Administration Console that is running on the AquaLogic Enterprise Security 2.5 Administration Server on either the local machine or a remote machine.
- AquaLogic Data Services Platform 2.1

Integrating with AquaLogic Data Services Platform: Main Steps

This section describes how to integrate AquaLogic Enterprise Security with AquaLogic Data Services Platform. Once integrated, you can use the AquaLogic Enterprise Security Administration Console to write and deploy a set of authorization and role mapping policies to protect Data Services and elements of those services.

Note: The instructions provided in this section use as an example the ALDSP sample application RTL App that ships with the ALDSP 2.1 software distribution. This procedure is representative of any integration of AquaLogic Enterprise Security with ALDSP.

To integrate AquaLogic Enterprise Security with AquaLogic Data Services Platform, perform the following tasks:

1. Install the ALES Administration Server, as described in *Installing the Administration Server*.
2. Install the WebLogic Server SSM, as described in *Installing Security Service Modules*. If you use ALSDSP with WebLogic Server 8.1, install the WebLogic Server 8.1 SSM. If you use ALSDSP with WebLogic Server 9.1 or 9.2, install the WebLogic Server 9.x SSM.
3. Use the ALDSP Administration console to enable the elements to which you want to control access, as described in "Enabling Elements for Access Control" on page 12-5.
4. Configure the WebLogic Server SSM, as described in "Creating the WebLogic Server SSM Configuration" on page 12-6. Note that this procedure varies, depending on whether you are using WLS 8.1 or WLS 9.x.
5. Bind the WebLogic Server SSM configuration, as described in "Binding the SSM Configuration" on page 12-7.

6. Distribute the WebLogic Server SSM configuration, as described in "Distributing the SSM Configuration" on page 12-7.
7. Create an instance of the WebLogic Server SSM, as described in "Creating an Instance of the Security Service Module" on page 12-7.
8. Enroll the instance of the WebLogic Server SSM, as described in "Enrolling the Instance of the Security Service Module" on page 12-7.
9. Create the `startWebLogicALES` file for WebLogic Server, as described in "Creating the WebLogic Server startWebLogicALES File" on page 12-7. Note that this procedure varies, depending on whether you are using WLS 8.1 or WLS 9.x.
10. Create the security configuration file, as described in "Creating the security.properties File" on page 12-8. Note that this step applies if you are using WLS 8.1 and does *not* apply if you are using WLS 9.x.
11. Configure security policies for your data services, as described in "Configuring Policy for Data Services" on page 12-8.
12. Replace `ld-server-core.jar` in `<WL_HOME>/liquiddata/lib` with the jar in ALDSP 2.5 patch for ALES.
13. Restart ALDSP.

Enabling Elements for Access Control

Before enabling your ALDSP domain for ALES, open the ALDSP Administration console:

1. Open a browser to visit `http://<hostname>:<port>/ldconsole`.
2. Login as Administrator.
3. Browse to the data services elements that are to be controlled by ALES. For this example, enable the following:

   a. Expand RTLServices/OrderSummaryView and select Security Tab.

   b. Select Secured Elements Tab.

   c. Expand elements and check OrderSummary > OrderDate as an element to be secured. (This allows the element call to go to the security check.)

   Do the same to secure CustomerView.ds > CUSTOMER > ORDERS> ORDER_SUMMARY > OrderDate.

Creating the WebLogic Server SSM Configuration

Securing ALDSP with ALES employs either the WLS 8.1 SSM or the WLS 9.x SSM. Install the WLS SSM on the machines on which you have installed ALDSP, as described in *Installing Security Service Modules*.

Next, create a new WLS SSM configuration named `aldsprealm`. An SSM configuration defines the set of security providers to use for adjudication, authentication, auditing, authorization, role mapping, and credential mapping services.

Refer to Table 12-1 and use the AquaLogic Enterprise Security Administration Console (for the WLS 8.1 SSM) or the WebLogic Server Administration Console (for the WLS 9.x SSM) to configure the security providers listed there. Set the Configuration ID to `aldsprealm`. For instructions on creating an SSM configuration, see Configuring and Binding a Security Service Module in *Installing Security Service Modules* and the Console Help.

Table 12-1 Providers for Use in ALDSP Integration

Provider	Configuration Settings
ASI Adjudication Provider	Accept default settings.
Log4j Auditor	Accept the default settings, and click Create.
Database Authentication Provider	Set the Control Flag to SUFFICIENT, and click Create. For the Details tab settings, except for the Identity Scope, the parameters are populated automatically. Set the Identity Scope to `aldspusers`, and click Apply. Note: Even though you set the Identity Scope to `aldspusers`, you do not actually create the `aldspusers` identity until you perform the steps in Creating the Realm Resource.
ASI Authorization Provider	On the General tab, accept the default settings, and click Create. On the Details tab, set the Identity Scope to `aldspusers`.
WebLogic Credential Mapper Provider	Uncheck the Credential Mapping Deployment Enabled check box, and click Create.
ASI Role Mapping Provider	On the General tab, accept the default settings, and click Create. On the Details tab, set the Identity Scope to `aldspusers`.

Binding the SSM Configuration

The SSM configuration must be bound to a Service Control Manager (SCM).

To bind the `aldsprealm` SSM configuration, see "Binding a Security Service Module to a Service Control Manager" in the Console Help.

Distributing the SSM Configuration

The `aldsprealm` SSM configuration must be distributed.

To distribute the `aldsprealm` SSM configuration, see "Distributing Configuration" in the Console Help.

Creating an Instance of the Security Service Module

Before starting a WebLogic Server Security Service Module, you must first create an instance of the WebLogic Server Security Service Module using the Create New Instance Wizard.

For information about creating an instance of a WebLogic Server Security Service Module, see Creating an Instance of a Security Service Module in *Installing Security Service Modules*.

Enrolling the Instance of the Security Service Module

You must have the ALES Administration Server running prior to enrolling the Security Service Module. For information about enrolling a security service module, see Enrolling the Instance of the Security Service Module in *Installing Security Service Modules*.

Creating the WebLogic Server startWebLogicALES File

Before you can start a WebLogic Server instance that uses BEA AquaLogic Enterprise Security, you must create the `startWeblogicALES` file based on the `startWeblogic` file that is located in the WebLogic domain. For information about how to do this, see:

- For the WLS 8.1 SSM: "Modifying the startWebLogic File" on page 7-2
- For the WLS 9.x SSM: "Using the ALES Administration Console to Configure Security Providers" on page 8-11

The `startWeblogic` file for the ALDSP domain for RTLApp is located in:

```
<bea_home>\<weblogic_home>\samples\domains\ldplatform
```

Creating the security.properties File

If you are using the WLS 8.1 SSM, create a text file named `security.properties` and place it in the domain directory. You use this file to define the AquaLogic Enterprise Security realm and the default realm.

```
# AquaLogic Enterprise Security Configuration File
#
# This file contains AquaLogic Enterprise Security configuration
# properties. By default, the AquaLogic Enterprise Security runtime
# looks for a property file called 'security.properties' in the
# working directory
wles.realm=aldsprealm
wles.default.realm=aldsprealm
```

Note: This step does not apply if you are using the WLS 9.x SSM

Configuring Policy for Data Services

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.

For more information on how to use the ALES Administration Console to write policy, see the *Policy Managers Guide* and the Console Help. In addition, the ALES Administration Server installation includes a set of sample polices for BEA AquaLogic Data Services Platform, located at `BEA_HOME/ales26-admin/examples/policy/aldsp_sample_policy`. You can import these sample policies and use them as a starting point for developing a full set of policies for your applications. 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 covers the following topics:

- "Creating Authorization Policies" on page 12-13

Creating the Identity Directory and Users

This section describes how to use the ALES Administration Console to create an identity directory, groups, and users for an ALDSP application.

Note: This procedure uses `aldspusers` as the name of the Identity directory; however, you can use a different name.

To create the Identity directory and users:

1. In the left pane, click Identity. The Identity page displays the name of each directory available.
2. Click New. The Create Directory dialog box appears.
3. In the Name text box, type `aldspusers` and click OK. The `aldspusers` directory appears in the list of Identity directories.
4. In the left pane, click Groups. The aldspusers > Groups page displays.
5. Click New. The Create Group dialog box appears.
6. Create the `LDSampleUsers` Group.
7. Create the sample users used in RTLApp and add them to the `LDSampleUsers` group:

   Jack (password: weblogic)
   Steve (password: weblogic)
   Tim (password: weblogic)

8. Create ldconsole administrator:

   weblogic (password: weblogic)

Configuring Resources and Privilege

This section describes how to use the ALES Administration Console to define the application resources that you will protect using ALES.

To configure resources, perform the following tasks:

- "Creating the RTLApp Application Resources" on page 12-10
- "Creating the ALDSP Resources" on page 12-10

Creating the RTLApp Application Resources

Note: You can choose any application name for your ALDSP application.

To create application resources, use the Administration Console to perform the following steps:

1. Expand the Resources folder, and click Resources. The Resource page displays.
2. In the Resources page, select the Policy node, and click New. The Create Resource dialog box appears.
3. In the Name text box, enter `aldsprealm`, select Binding from the Type drop-down list box, and click Ok. The `aldsprealm` resource appears under the Policy node.
4. Select the `aldsprealm` resource and click Configure. The Configure Resource dialog box appears.
5. From the Type drop-down list box, select Binding Application, check the Distribution Point and Allow Virtual Resources check boxes, and click Ok.
6. Refer to Table 12-2 and modify the configuration of the ASI Authorization provider and the ASI Role Mapper provider as described there.

Table 12-2 ALDSP SSM Configuration Modifications

Security Provider	Configuration Setting
ASI Authorization Provider	1. On the Details tab, set the Application Deployment Parent to `//app/policy/aldsprealm` and click Apply. 2. On the Bindings tab, from the Bind drop-down menu, select `//app/policy/aldsprealm`, and click Bind.
ASI Role Mapper Provider	1. On the Details tab, set the Application Deployment Parent to `//app/policy/aldsprealm` and click Apply. 2. On the Bindings tab, from the Bind drop-down menu, select `//app/policy/aldsprealm`, and click Bind.

Creating the ALDSP Resources

Figure 12-2 shows the ALDSP resource tree with all nodes expanded except the RTLApp node. The resources under that RTLApp node are shown in Figure 12-3. You must create the resources shown in Figure 12-2 and Figure 12-3.

Figure 12-2 ALDSP Resource Tree with RTLApp Node Collapsed

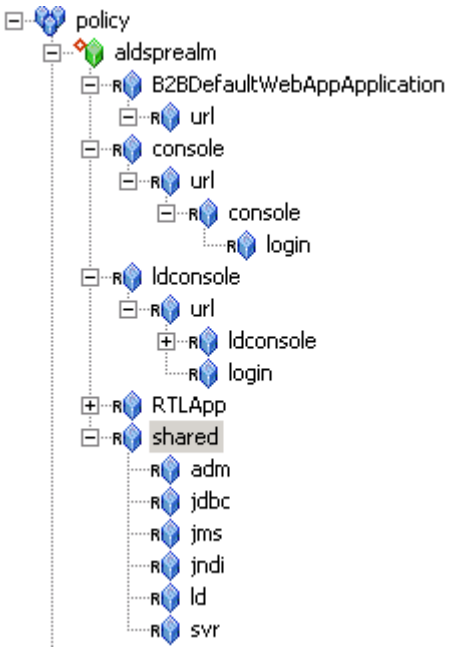
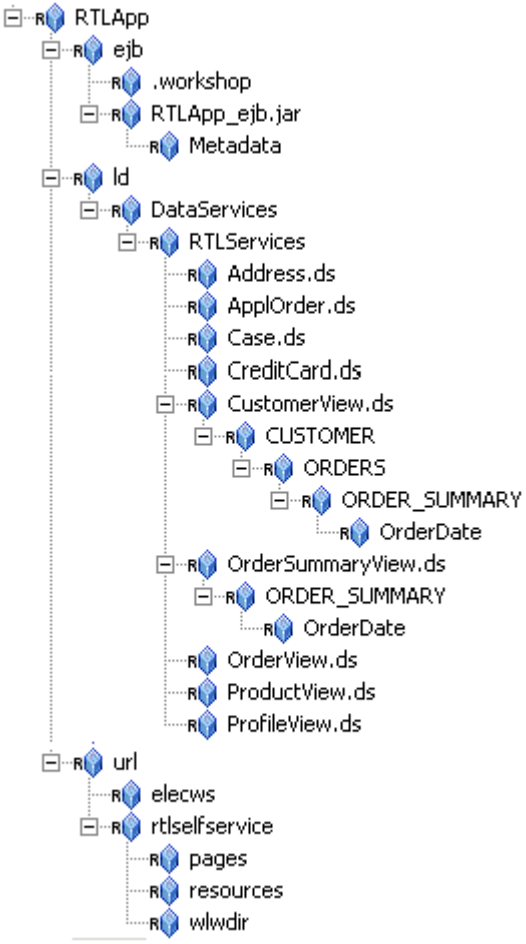


Figure 12-3 ALDSP Resource Tree with RTLApp Node Expanded



Creating the Role Mapping Policies

This section describes how to use the Administration Console to create the role mapping policies that will be used to control access the sample ALDSP application.

Table 12-3 lists the role mapping policies required for the WebLogic domain.

Table 12-3 ALDSP Application Role Mapping Policy

Role Mapping Policy	Description
grant(//role/Everyone, //app/policy/aldsprealm, //sgrp/aldspusers/allusers/) if true;	Creates the role mapping policy necessary for the Everyone role to be used in the aldsprealm Identity directory. Note: If you do not create the Everyone role mapping policy correctly, none of the policy rules defined in Table
grant(//role/Admin, //app/policy/aldsprealm, //user/aldspusers/weblogic/) if true;	Grants the weblogic user Admin role within the aldsp realm.

To create the role mapping policies, refer to Table 12-3 and perform the following steps.

Note: If you do not create the Everyone role mapping policy correctly, none of the authorization policies defined in Figure 12-4 will work.

1. Expand the Policy folder in the left pane, and click Role Mapping Policies. The Role Mapping Policies page appears.
2. Click New. The Create Role Mapping Policy dialog box appears.
3. Select the Grant radio button.
4. Select the Roles tab, select Everyone in the Available Roles list box, and click Add.
5. Select the Resources tab, select `aldsprealm`, and click Add.
6. Select the Policy Subjects tab, select `allusers` in the list box, click Add, and click Ok.

Creating Authorization Policies

This section describes how to use the Administration Console to create authorization policies to protect data services and application resources. Table 12-4 lists the authorization policies required for WebLogic Server, the WebLogic Server console, and the RTL sample application.

Table 12-4 Authorization Policies

Authorization Policy	Description

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/RTLApp/url/rtlselfservice/pages], [//role/Admin]) if true;
grant(any, [//app/policy aldsprealm /RTLApp/ejb/RTLApp_ejb.jar/Metadata,//app/policy/aldsprealm/RTLApp/ejb/RTLApp_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;
	Grants Admin Role and/or weblogic user permission to boot the WebLogic Server and perform administrative tasks.

grant(//priv/any, //app/policy/aldsprealm/ldconsole/url/ldconsole, //role/Admin) if true;	This policy gives WebLogic full access to the LD console.

Table 12-4 Authorization Policies (Continued)

Authorization Policy	Description

grant(//priv/lookup, //app/policy/aldsprealm/shared/jms, //role/Everyone) if true;
grant(any, //app/policy/aldsprealm/shared/ld, //role/Everyone) if true;
grant(//priv/lookup, [//app/policy/aldsprealm/shared/jdbc,//app/policy/aldsprealm/shared/jndi], //role/Everyone) if true;
grant(//priv/send, //app/policy/aldsprealm/shared/jms, //role/Everyone) if true;
grant(//priv/GET, //app/policy/aldsprealm/console/url/console/login, //role/Everyone) if true;
grant(//priv/reserve, //app/policy/aldsprealm/shared/jdbc, //role/Everyone) if true;
grant([//priv/GET,//priv/POST], //app/policy/aldsprealm/ldconsole/url/ldconsole/login, //role/Everyone) if true;
grant([//priv/GET,//priv/POST], //app/policy/aldsprealm/RTLApp/url/elecws, //role/Everyone) if true;
grant(//priv/GET, //app/policy/aldsprealm/ldconsole/url/ldconsole/images, //role/Everyone) if true;
grant(//priv/GET, [//app/policy/ aldsprealm /B2BDefaultWebAppApplication/url,//app/policy/aldsprealm/RTLApp/url/rtlselfservice/resources,//app/policy/aldsprealm/RTLApp/url/rtlselfservice/wlwdir], //role/Everyone) if true;
	Grants permission to those in the role Everyone (includes the anonymous user) to access all of the shared open resources.

grant([//priv/GET,//priv/POST], //app/policy/ aldsprealm /RTLApp/url/rtlselfservice, //user/aldspusers/Steve/) if true;
deny(any, [//app/policy/ aldsprealm /RTLApp/ld/DataServices/RTLServices/OrderSummaryView.ds/ORDER_SUMMARY/OrderDate,//app/policy/aldsprealm/RTLApp/ld/DataServices/RTLServices/CustomerView.ds/CUSTOMER/ORDERS/ORDER_SUMMARY/OrderDate], //user/aldspusers/Steve/) if true;
	Denies Steve access to the Order Date element of the Customer View Data Service

Table 12-4 Authorization Policies (Continued)

Authorization Policy	Description
deny(any, //app/policy/aldsprealm/RTLApp/ld/DataServices/RTLServices/ProfileView.ds, //user/aldspusers/Jack/) if true;	Denies Jack access to an entire data service
grant(any, [//app/policy/aldsprealm/RTLApp/ejb,//app/policy/aldsprealm/RTLApp/ld,//app/policy/aldsprealm/RTLApp/url/rtlselfservice/pages], [//sgrp/aldspusers/LDSampleUsers/,//role/Admin]) if true;	Grants Admin and Sample Users access to Data Services

Perform the following steps to create the authorization policies listed in Table 12-4.

1. Expand the Policy folder in the left pane, and click Authorization Policies. The Authorization Policies page appears.
2. Click New. The Create Authorization Policy dialog box appears.
3. Select the Grant radio button.
4. To create the first authorization policy listed in Table 12-4, click the Policy Subjects, select the Admin role from the Roles List box, click Add, and click Ok.

   Note: If Table 12-4 lists multiple resources for a single privilege and role, you may define all of the resources in one authorization policy.
5. Repeat for each of the remaining authorization policies listed in Table 12-4.

Discovering Data 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*.

Distributing Policy and SSM Configuration

Distribute policy and SSM configuration to the WebLogic Server SSM.

For information on how to distribute policy and SSM configuration, see “Deployment” in the Console Help. Be sure to verify the results of your distribution.

Starting the WebLogic Server

To start a WebLogic Server instance, perform the following steps:

1. Open a shell (command prompt) on the machine on which you created the domain.
2. Change to the ALDSP sample domain directory:
   `<bea_home>\<weblogic_home>\samples\domains\ldplatform`
3. Run one of the following scripts:

   On Windows: `startWebLogicALES.cmd`

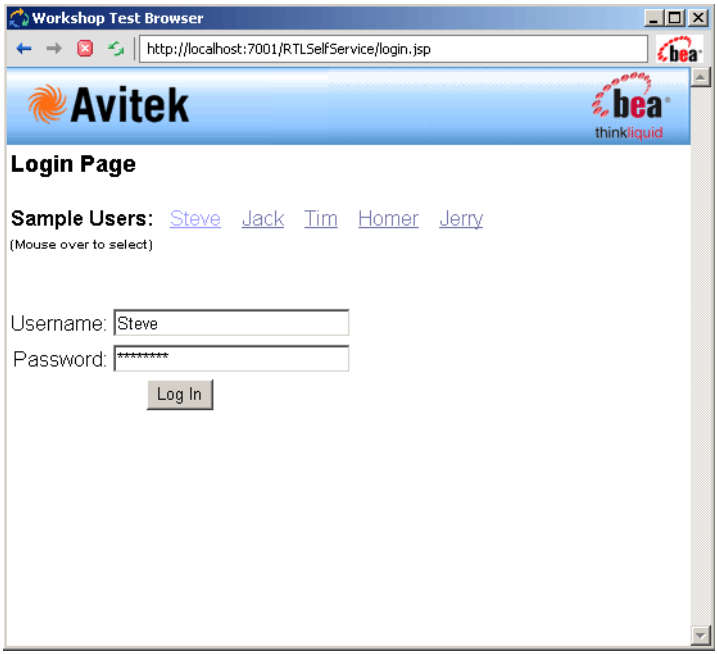
   On UNIX: `startWeblogicALES.sh`

Accessing the ALDSP Application

To access the RTLApp running on an ALDSP server:

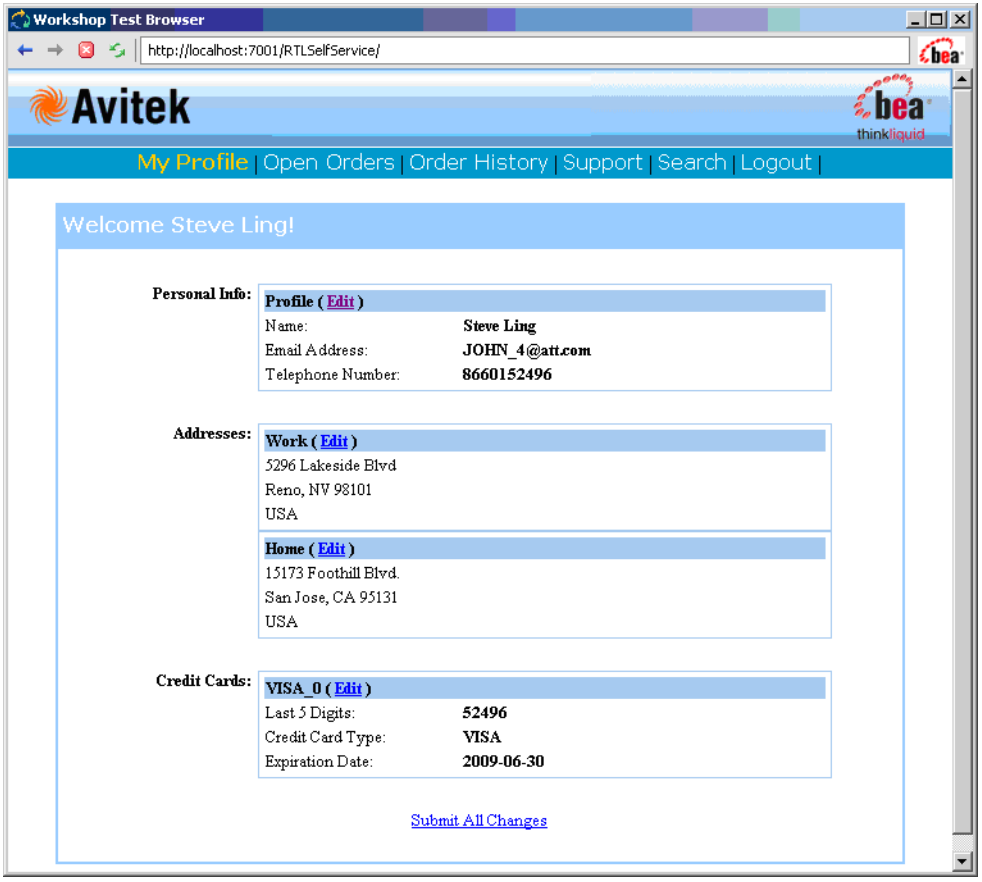
1. Open browser to visit `http://<hostname>:<port>/RTLSelfService`, where `<hostname>` is the machine on which RTL application is running. The browser is redirected to the authentication page (see Figure 12-4).

Figure 12-4 Authentication Page



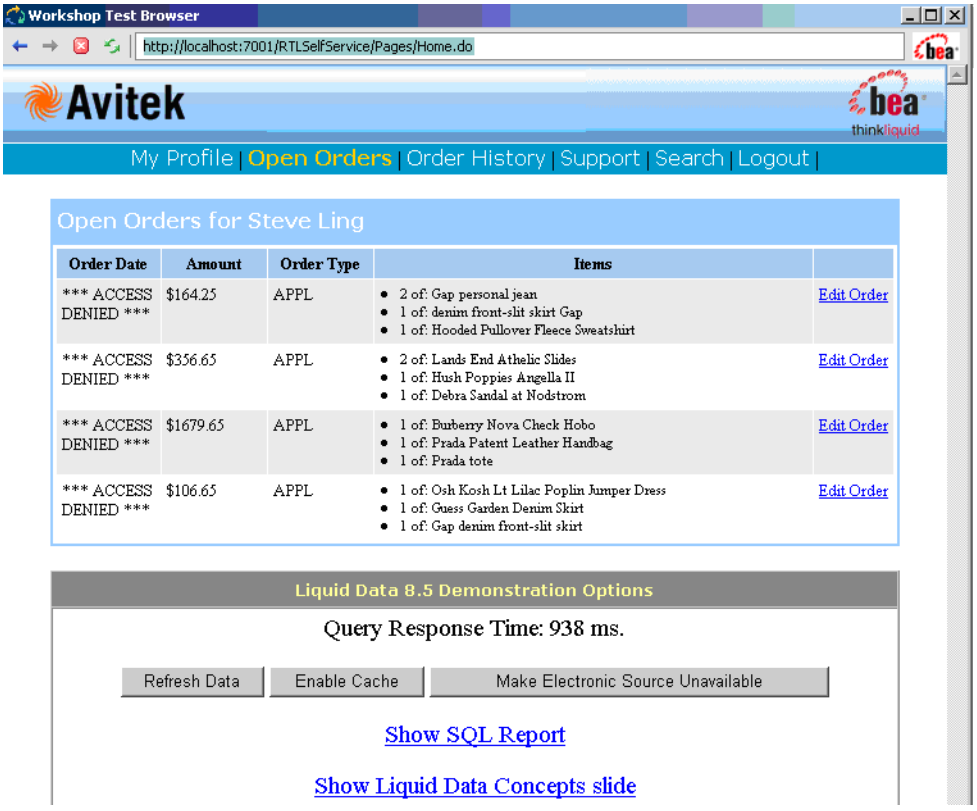
2. Set username as Steve by dragging over link, then click Login button. Your client should be granted access to the Profile Page (see Figure 12-5).

Figure 12-5 Profile Page



3. Select Open Orders Page from top menu. Open orders should be visible (see Figure 12-6). Order Data should have "ACCESS DENIED".

Figure 12-6 Open Orders Page



Pre- and Post-Processing Data Redaction Solutions

ALES 2.6 provides mechanisms to redact ALDSP data both before and after the ALDSP Engine processes a client query request.

Redacting ALDSP data before the ALDSP Engine processes a client query request is called **pre-processing**. Redacting ALDSP data after the ALDSP Engine processes a client query request is called **post-processing**. This section describes both solutions. This section begins with an overview of the two solutions.

Overview of Pre- and Post-Processing Data Redaction Solutions

In the pre-processing data redaction solution, the result of the data-retrieving request is protected by adding security constraints before the XQuery function is executed. That is, before the ALDSP Engine processes a client query request, an ALES security constraint is created and inserted into the client query. Pre-processing may return either predicates to modify the original query, or provide a substitute function name.

ALDSP uses the WLS security framework to control access to the data services. When a user calls a function of a data service, the request is first authorized by WLS. ALDSP provides a plug-in mechanism to get additional **obligations** from the authorization response that is returned from the WLS security framework. These obligations are returned only when the authorization decision is is true, and are used to modify the original request to add additional restraints before it calls to the data services.

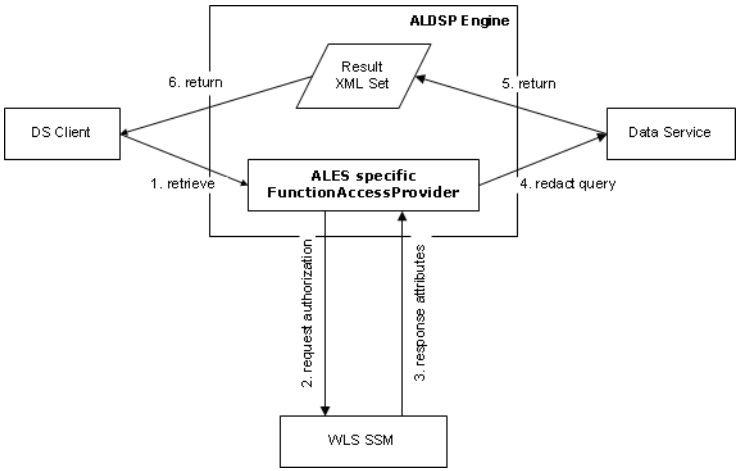
Because this approach is performed before pushing the query to the data sources, it is called pre-processing.

The pre-processing data redaction solution provides an ALES-specific plug-in. This plug-in calls the Java API to authorize the user request, then gets the response attributes from the authorization response and returns them as obligations.

Consider the sequence of events shown in Figure 12-7:

1. The ALDSP engine receives a data service client request and invokes the ALES plug-in.
2. The ALES plug-in calls the ALES Java API for authorization. The authorization decision may return additional predicates as responses.
3. The ALES plug-in returns the authorization decision and responses to the ALDSP engine.
4. The ALDSP engine adds the predicates, or a function name, or both to the original query, and calls the data service
5. Return the result to the ALDSP engine.
6. Return the result to the data service client.

Figure 12-7 Overview of Pre-Processing Solution



In the post-processing data redaction solution, the ALDSP engine retrieves the data from the data service and then invokes the relevant security XQuery function. The decision about what data to retrieve is already made, and the issue is whether to grant access and return the data. Because this approach processes the result-set after querying the data, it is called post-processing.

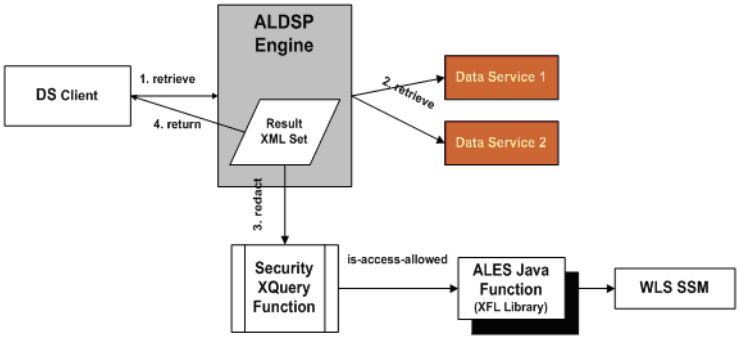
This approach may not be suitable for fast operations or very large data sets.

Consider the sequence of events shown in Figure 12-8:

1. The data service client sends a data retrieving request to ALDSP.
2. The ALDSP engine retrieves the data from the data service (i.e. Data Service 1, Data Service 2).
3. Before returning the data, ALDSP invokes the relevant security XQuery function for the data element.
4. The security XQuery function invokes one of the ALES Java methods, passing in the resource name, one or more attribute names, and values for those attributes.
5. The ALES Java method invokes the WLS SSM and gets the result of authorization and optional responses defined in the queries.
6. The ALES Java method returns authorization decisions and a set of responses to the security XQuery function.

7. The XQuery function returns the decision. The function can use the responses to make the decision. For example, the policy decision may be true, but based on the responses the function returns false.

Figure 12-8 Overview of the Post-Processing Solution



How the Pre-Processing Solution Integrates with ALDSP

The ALES-specific function access provider plug-in `com.bea.security.ales.aldsp.ALESFunctionAccessProvider` calls the Java SSM to do authorization, gets the response attributes, and then returns the response attributes as obligations to ALDSP engine. The ALES policy file needs to include policies with the response attributes appropriately defined.

For example, consider a policy such as the following:

```
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 permits user *system* to perform `ALDSP_QUERY` privilege on the data service `OrderView`'s function `getOrders`, and returns the attribute `aldsp_replacement_function`, with a value `getOrdersThatAmountLessThan1000` if the authorization decision is true.

When a data service client calls the `getOrders` function, ALDSP will call the ALES-specific implementations to do authorization. This policy will be evaluated and return an attribute `aldsp_replacement_function` with a value `getOrdersThatAmountLessThan1000` if the authorization decision is true.

The ALES plug-in returns this attribute and value as obligations to the ALDSP engine.

Finally, the ALDSP engine calls the replacement function `getOrdersThatAmountLessThan1000` instead of the original. This new function should have exactly the same signature; no additional verifications are performed at runtime.

Types of Pre-processing Obligations

ALDSP supports two types of obligations. You can use them alone or together.

Replacement Function

The replacement function is called instead of the original user-requested function. For example, assume that the user requests to call a function named "getOrders". After authorization, a replacement function "getOrdersThatAmountLessThan1000" is returned, and this function is called to return a lesser result than the original.

XQuery expression

The returned XQuery expression is applied to the invoked function, or its replacement, as a predicate. There may be multiple predicates returned, and they will all be applied to the function.

For example, assume that the user requests to call a function named "getOrders". After authorization, an XQuery expression "./order/amount < 1000" is returned. This expression is applied to the requested function "getOrders", so that only a subset of orders matching the criteria provided in the obligation will be returned.

Predefined ALES Privilege is Required

ALES defines the privilege `ALDSP_QUERY` for use in policies. You must use this privilege for any policy for which you want to apply pre-processing.

Predefined ALES Response Attribute Names Are Required

ALES defines three response attribute names for you to use with a replacement function or XQuery expression. The attribute names are all in lower case.

These attribute names can be returned by the ALES evaluation functions `report` and `report_as`, and can also can be returned by user-defined evaluation functions.

aldsp_replacement_function

Provide a value for this attribute. The value will be used as a function name that ALDSP calls to 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. For example, you can define expressions such as `./order/amount < 1000`, or `./order/amount < 1000 OR ./order/customerid eq 'CUSTOMER4'`.

These expressions will be used as a filter in ALDSP. The value of the attribute can be a list. In this case, ALDSP will apply the AND operator to the list values.

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

aldsp_namespace_binding

This attribute defines the namespace mapping for prefixes, and is used in XQuery expressions. For example, if the following xquery expression were returned: `./f1:region eq 'west'`, the namespace map should contain the following entry mapping the string prefix to the string namespace (for example, `{"f1","http://com.bea.security"}`):

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

How to Use the Pre-Processing Data Redaction Solution

Before you can use the pre-processing data redaction solution, you must integrate ALES with ALDSP, as described in TBS.

After you have done this, then:

- "Modify set-wls-env Script to Enable Pre-Processing Solution" on page 12-25
- "Write Replacement Function" on page 12-25
- "Define Policies for Replacement Function" on page 12-25
- "Define Policies for XQuery Expression" on page 12-26
- "Define Namespace Bindings" on page 12-26

These tasks are described in the sections that follow.

Modify set-wls-env Script to Enable Pre-Processing Solution

You need to modify the file `set-wls-env`, which is in the WLS SSM instance directory. To do this:

1. Navigate to the WLS SSM instance directory, and edit `set-wls-env.bat`.
2. Add pre-processing jar file (`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`.
4. Save your changes.

Write Replacement Function

If you are going to use replacement functions to protect data services, implement the replacement function on the target data service. This replacement function must have the same signature (return type, and number and type of parameters) as the replaced function.

For example, if you are going to restrict `OrderView.getOrders` to return only the orders that total less than $1000, write an XQuery function (named something such as `getOrdersThatAmountLessThan1000`) to implement the restriction. This function must have the same return type, and same number and type of parameters as `getOrders`.

Define Policies for Replacement Function

If you are going to use a replacement function to protect data services, define a policy that allows someone to access the target data service's function. (If you have an existing policy, you only need modify it.) Return a named attribute `aldsp_replacement_function`, the value of which is the replacement function. There are no additional access control checks performed for the replaced function.

For example, to use the function `getOrdersThatAmountLessThan100` to replace the function `getOrders` when an anonymous user calls `getOrders`, define a policy such as:

```
grant (

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

```
) if
report_as("aldsp_replacement_function","getOrdersThatAmountLessThan1000")
```

Define Policies for XQuery Expression

If you are going to use an XQuery expression to protect data services, define a policy that allows someone to access the target data service's function. Return a named attribute `aldsp_xquery_expression`, the value of which is the desired XQuery expression.

For example, to restrict the function `getOrders` to return orders that amount less than $1000 if an anonymous user calls it, define a policy such as:

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

Define Namespace Bindings

If namespaces are used in an XQuery expression, you must define namespace bindings. (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,
 //user/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")
```

How the Post-Processing Solution Integrates With ALDSP

The ALDSP security XQuery functions, which are described in Creating a Security XQuery Function, enable you to specify custom security policies that can be applied to data elements. In particular, security XQuery functions are useful for creating data-driven policies (policies based on data values). For example, you can block access to an element if the order amount exceeds a given threshold.

ALES 2.6 provides two ALES Java methods that you can call from within your security XQuery function. (The ALES Java methods are used as XFL functions.) These ALES Java methods then invoke the WLS SSM, which determines whether the access request should be granted.

The first ALES Java function returns a Boolean value to indicate the result of the access permission decision. The second ALES Java function returns a string array that contains the result of the access permission decision, and response attributes. You can then use the response attributes to implement some specific logic in your security XQuery function.

You can use the ALES Java methods in place of, or in addition to, the following ALDSP `access control function extensions` to the BEA implementation of XQuery:

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

For example, consider the typical security XQuery function usage shown in Listing 12-1. In this instance, you might replace the function `fn-bea:is-access-allowed` with an ALES Java method.

Listing 12-1 `Typical Security XQuery Function Usage`

```
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, it is not XML schema-neutral. Therefore, you create the security XQuery function based on the custom data service schema.

One limitation of the custom security XQuery function is that it must be created in Workshop, instead of in ALDSP console. A custom XQuery security function cannot be created from the ALDSP console, because the console compiler does not access the custom functions used in it.

ALES Java Methods

ALES 2.6 includes the following two Java methods, both of which are in the class named `com.bea.security.ales.aldsp.AccessController`.

```
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)
```

ALES Java Method Parameter Format

The ALES Java methods have three parameters:

- The first parameter is the resource name. The resource name must match a resource name defined in an ALES policy. That is, the resource name is defined in ALES and indicates the secured ALDSP data element.
- The other two parameters are two string arrays, one for attribute names and one for values.

  The indexed attribute names in the first parameter correspond to the indexed attribute values in the second.

  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`.

ALES Java Method Return Values

If you use the ALES `is_access_allowed` method, the boolean return value represents the access permission, either `true` or `false`. You can return this boolean value directly to the security XQuery function, or do some additional operation based on the result.

If you use the `is_access_allowed_with_response_attributes` method, the first element of the array is the access permission decision, either `true` or `false`.

The other array elements represent 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`.

In addition, you can use the response attribute value to implement additional logic, as described in "How to Write Policies That Return Response Attributes as ALDSP Obligations" on page 12-29.

How to Write Policies That Return Response Attributes as ALDSP Obligations

There is nothing unique about the ALES policy that you create to protect an ALDSP resource, beyond the general requirement that the resource name, attribute names, and attribute values you specify in the ALES Java method must correlate with the policy. The general steps you perform to create a policy are described in "How to Integrate the ALES Java Methods" on page 12-31.

However, 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")`.

Then, in your security XQuery function, you can test the return response attributes. Remember that the first element of the array is the access permission decision, either `true` or `false`. For example, consider Figure 12-9:

Figure 12-9 Testing Return Response Attributes

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

The `report_as` function loads a named response attribute with a specified value. The value may be an attribute, a constant or a string literal. You can specify multiple values, in which case the response attribute is returned as a list.

While the `report()` and `report_as()` functions are run when the policy is evaluated, they are not really constraints of the policy. Data reported by the functions are returned only if the adjudicated authorization decision agrees with the policy. This means the attributes returned from GRANT policies are not passed to the caller unless the overall access decision is PERMIT.

How to Write and Configure the Security XQuery Function

Use Workshop, not the ALDSP console, to add a new XQuery function as the security XQuery function in the ALDSP application, as follows:

1. Import the ALES Java method via an XFL library in the current Workshop application, as described in "How to Integrate the ALES Java Methods" on page 12-31.

   The XQuery Function Library (XFL) is a facility for providing auxiliary functions across multiple data services. The ALES Java method is an XFL function. You import it from a Java class shipped by ALES.

2. In order to get the elements of the data service, 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. Inside of the XQuery Function, specify all of the attributes used in the ALES policies that protect the resource, and two string arrays for names and values. A detailed example is provided in "ALES Security XQuery Function Integration Example" on page 12-33.

   Note: 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.

How to Integrate 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 "Integrating with AquaLogic Data Services Platform: Main Steps" on page 12-4.

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 the jar files `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, select the node of the data service project, right click and select Import Source Metadata.

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

   d. Locate the jar file `alesxfl.jar`, expand it, 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. Add a new XQuery function as the security XQuery function in the ALDSP application by using Workshop, not the ALDSP console:

   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. Inside of 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 Integration Example" on page 12-33.

4. Specify which element of the data service is protected in the ALDSP configuration file:

   a. Open the configuration file of the ALDSP application.

   The file location is `<ALDSP_DOMAIN>/liquiddata`.

The file name is *<ALDSP_APPLICATION_NAME>*`LDConfig.xml`.

b. Find the element (for example *<con:DSConfiguration>*)whose id is the data service name to be protected.

c. Under the element you found in Step b., 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 Integration Example

This example is based on the RTLApp that is shipped by ALDSP.

In this example, 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.

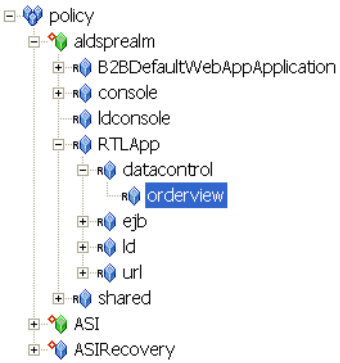
The integration example follows these steps:

1. Configure and distribute the policy in ALES:

   a. Log in to the ALES Administration console (for example, `https://localhost:7010/asi`).

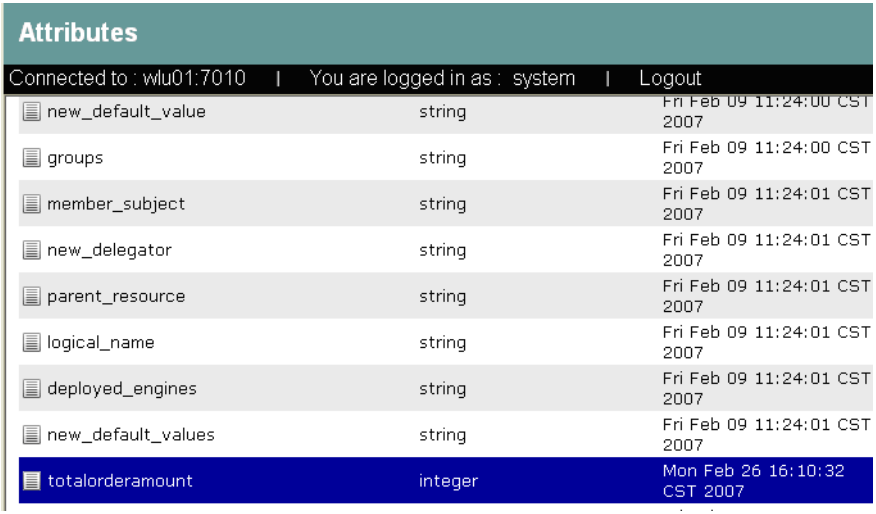
   b. Define a resource named datacontrol/orderview under the resource node RTLApp, as shown in Figure 12-10.

Figure 12-10 ALDSP Resource Tree



   c. Define a dynamic attribute named totalorderamount whose type is integer, as shown in Figure 12-11.

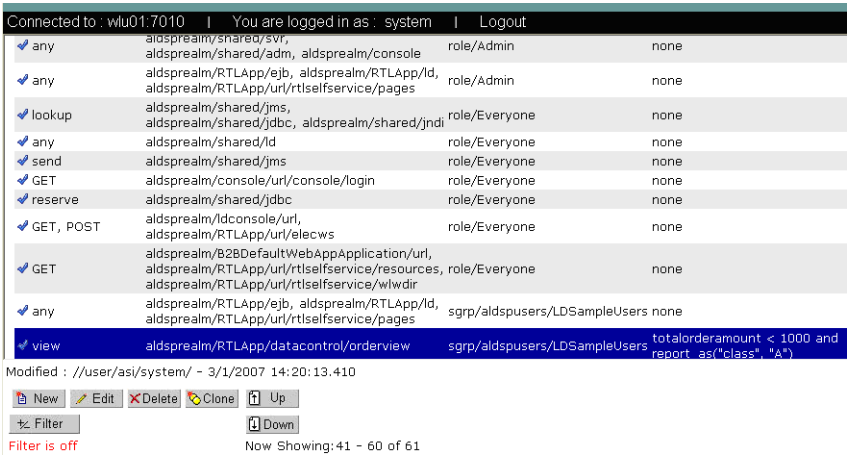
Figure 12-11 ALDSP Dynamic Attribute



d. Define an authorization policy for the resource as shown in Figure 12-12.

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

Figure 12-12 ALDSP Authorization Policy



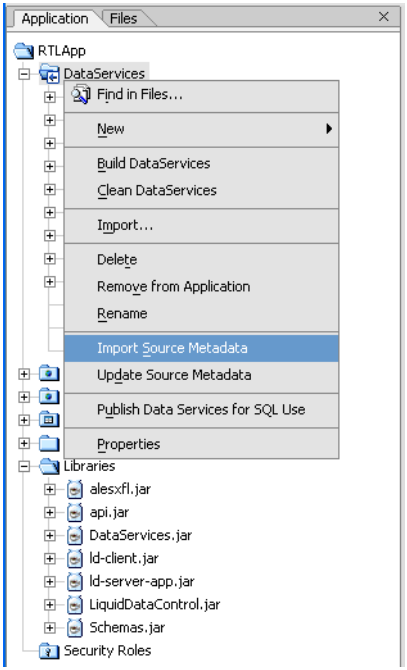
e. Distribute the policy change.

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

   a. Copy the jar files `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 on the DataServices folder and select Import Source Metadata from the pop-up menu, as shown in Figure 12-13.

Figure 12-13 Import ALES Java Function



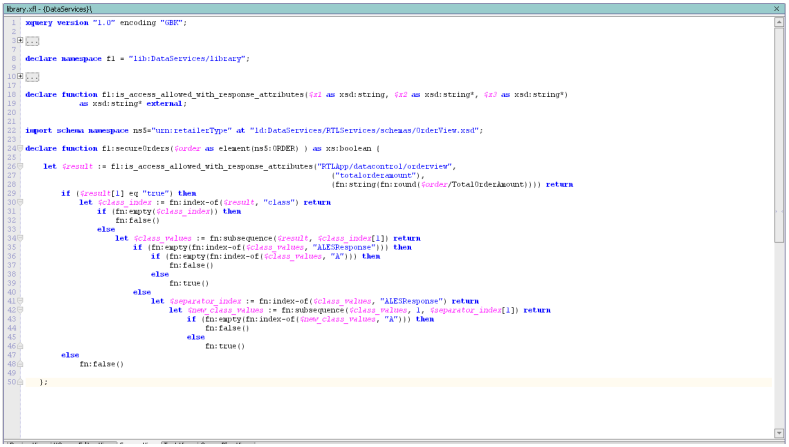
   c. Select Java Function as the Data Source Type, and click Next.

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

   e. In the next page, select the method `is_access_allowed_with_response_attributes`, and finish the wizard.

3. Add a security XQuery function in the XFL file `library.xfl`, as shown in Figure 12-14. 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 atttibute `class` is not found, return false.
   - Line 33 to 46: Check if the response attribute `class` contains the value `A`.

Figure 12-14 Security XQuery Function



4. Specify which element of the data service is protected in the ALDSP configuration file:
   a. Open the configuration file of the ALDSP application.

The file location is *BEA_HOME*/weblogic81/samples/domains/ldplatform/liquiddata.

The file name is RTLAppLDConfig.xml.

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>
```

5. Redeploy the application in Weblogic Console:

   a. Log in to the Weblogic Server Administration Console.

   b. Expand the node Deployments|Applications, and select the RTLApp application.

   c. In the right tab, select the Deploy tab.

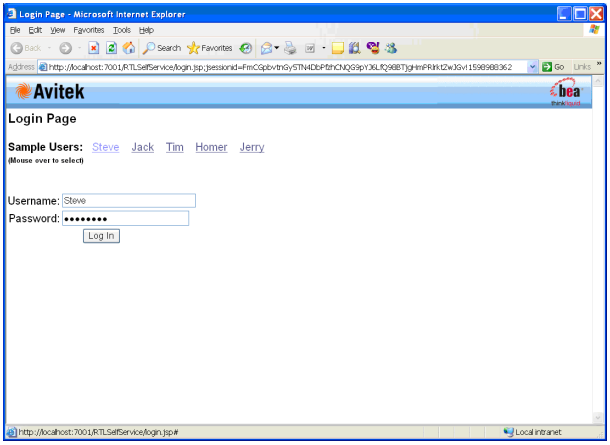
   d. Click the Redeploy Application button.

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

6. Restart the Weblogic Server.

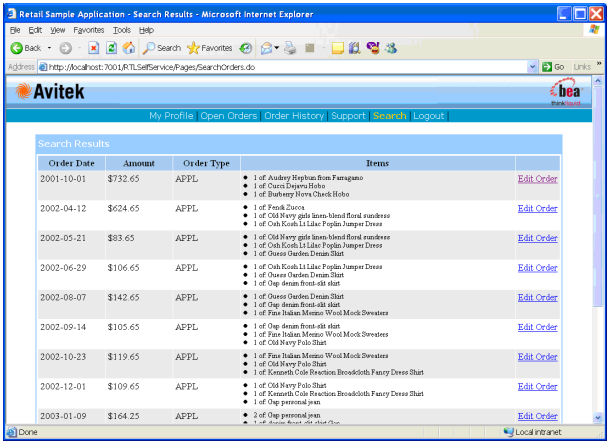
7. Open the web application RTLSelfService (for example, http://localhost:7001/RTLSelfService), and select the user Steve to log in, as shown in Figure 12-15.

Figure 12-15 Avitek Login Page



8. 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 12-16.

Figure 12-16 Search Results with ALES Protection



CHAPTER 13

Integrating with AquaLogic Enterprise Repository

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

Introduction

AquaLogic Enterprise Repository (ALER) manages the metadata for any type of software asset, from business processes and Web services to patterns, frameworks, applications, and components. You can use AquaLogic Enterprise Repository to manage AquaLogic Enterprise Security policy data as ALER software assets. By integrating ALER with ALES, you can:

- More easily 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.

See the AquaLogic Enterprise Repository documentation at http://edocs.bea.com/aler/docs26/index.html.

Setting Up ALER to Manage ALES Assets

The main steps in setting up ALER to manage ALES assets are:

1. Install and configure ALER and ALES. For information about installing ALER, see the *ALER Installation Guide*. For information about installing ALES, see *Installing the Administration Server*.
2. Configure ALER to enable import and export, as described in "Setting ALER System Properties for Import and Export" on page 13-2.
3. Import the ALES Policy Asset Type into ALER, as described in "Importing the ALES Policy Asset Type into ALER" on page 13-2.

Setting ALER System Properties for Import and Export

In order to use the ALES policyIX utility to import and export ALES assets to ALER, set to true the following system properties in ALER:

- `cmee.importexport.enabled` (enables the REX Open API on the ALER server)
- `cmee.extframework.enabled` (enables exporting assets to and importing assets from ALER)

To set these system properties:

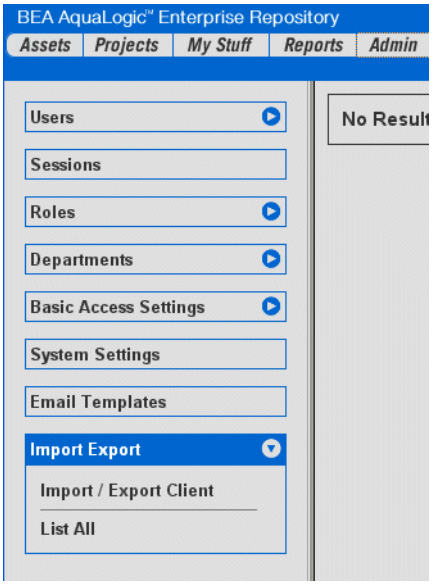
1. In the ALER console, open the **Admin** tab.
2. Select **System Settings** from the left column.
3. Under **Import / Export and Introspection > Import / Export**, set **Import/Export Client** cmee.importexport.enabled to True.
4. Under **Open API > Common**, set **Open API Enabled** cmee.extframework.enabled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

Importing the ALES Policy Asset Type into ALER

Import the ALES Policy Asset Type into ALER:

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

Figure 13-1 Starting the ALER Import/Export Client

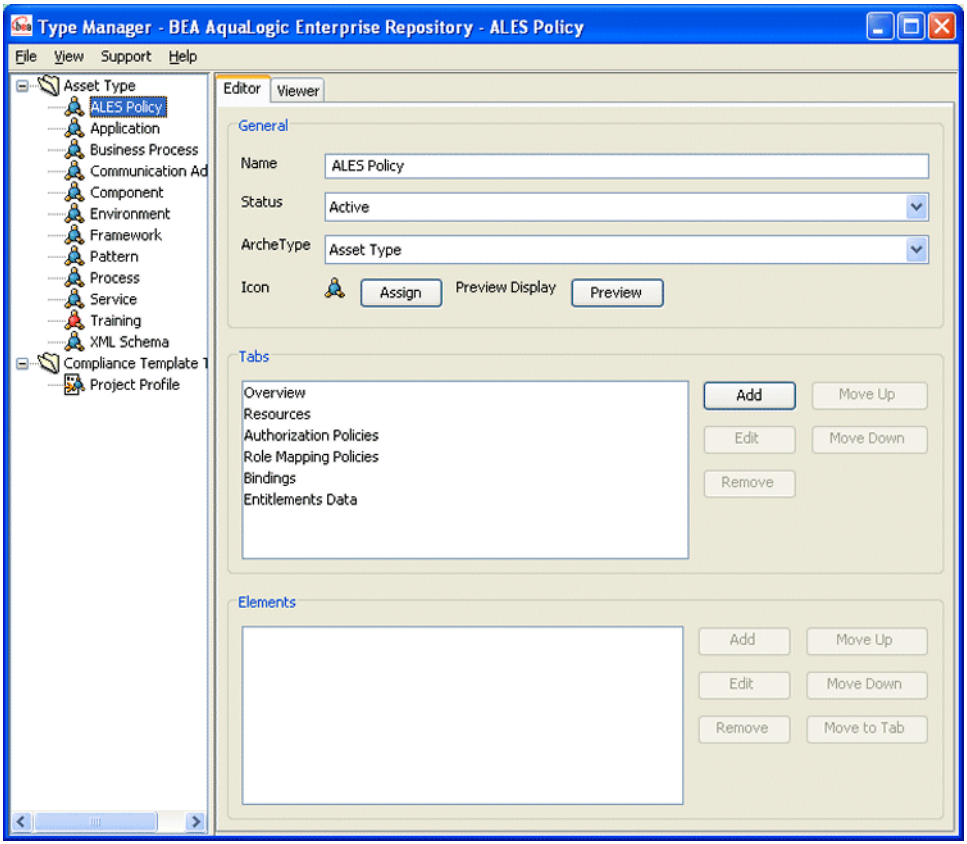


3. In the Select file to import field of the Import tab, locate the ALES Policy Asset Type definition. It can be found at *ALES_ADMIN_HOME*`/lib/alesAssetSchemaObjects.jar`. Click **Next** twice.
4. Close the ALER Import/Export Client.

Verify that the ALES Policy Asset Type was created:

1. In the ALER console, open the **Assets** tab and click **Edit/Manage Assets**. The Asset Editor opens.
2. In the Asset Editor, select **Actions > Manage Types**. The ALES Policy asset should appear in the Type Manager.

Figure 13-2 ALER Type Manager



Using ALER to Manage ALES Assets

Once you have set up the ALES Policy Asset Type in ALER, you can use ALER to manage workflow, policy approval, and policy versioning. The ALER console also gives you the ability to directly modify data in an ALES Policy Asset. This is not recommended, however. Instead, changes to ALES Policy Asset data should be made in ALES and imported into ALER. See "Importing and Exporting with policyIX" on page 13-7.

ALES Policy Asset Type

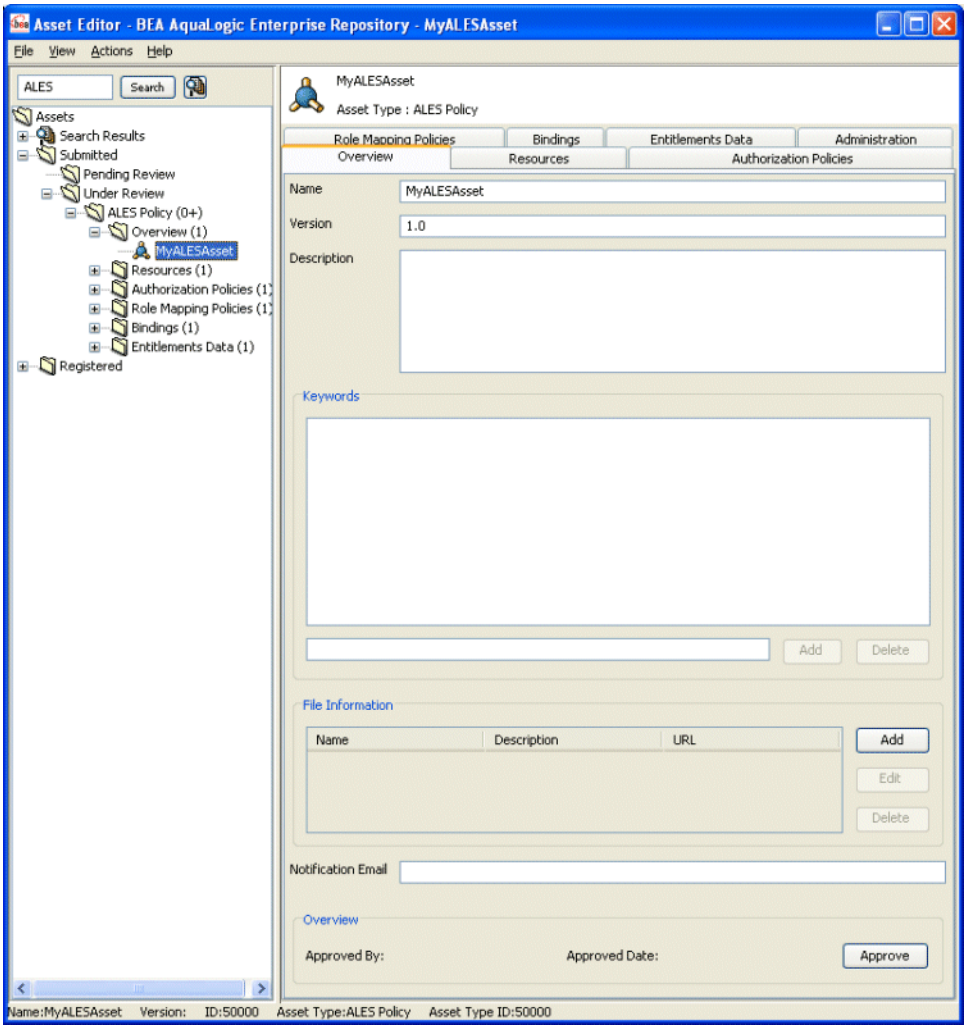
In ALER, the term *asset* is used to describe an object that contains metadata. Before an asset can be created, there must be an asset type by which to categorize the asset. An asset type contains the structure of the asset. The ALER asset type named ALES Policy is a defined asset type in ALER that contains ALES metadata, such as privileges, policy, resources, and resource attributes. An ALES Policy Asset can be considered a container of ALES policy information.

Viewing ALES Policy Assets in the ALER Console

The ALER Asset Editor displays ALES Policy Assets in the following tabs (see Figure 13-3):

- Overview - General information such as asset name and location, and notification e-mail (the e-mail address of the user to be notified if the asset changes).
- Resources - Data for resources and resource attributes. The Resources tab displays columns for each of the following:
  - Resource Name - The name of the resource
  - Resource Type - The type of the resource (a dropdown menu containing only valid resource types)
  - Description - Description of the resource
  - Virtual Resource - Indicates whether the resource is a virtual resource
  - Distribution Point- Indicates whether the resource is a Distribution Point
  - Attributes - The resource attributes that are associated with the resource. This column contains a table of attributes that belong to the resource.
- Authorization Policies - Allows you to add, edit, or delete Authorization policies, Privileges and Privilege Groups.
- Role Mapping Policies - Allows you to add, edit, or delete role mapping policies.
- Bindings - The SCM name, SSM name, and Binding Resources of the asset.
- Entitlements Data - Information related to Identities, Identity Schema, Groups, Group attributes and Group memberships that are created by the Entitlements UI and RBAC API.
- Administration - ALER administration information, with information about the creation, submission, acceptance, assignment, notification, and registration of the asset.

Figure 13-3 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 you import 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 is incremented by 1. You can also modify the version number of an asset within ALER.

Importing and Exporting with policyIX

AquaLogic Enterprise Security includes a utility named policyIX that you can use to import and export policy data from ALER. You can use policyIX to import or export directly between ALES and ALER, or you can import or export from a policy file. Importing or exporting from a file does not require you to be able to connect to a running ALES Administration Server.

Exporting to ALER from ALES

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

```
policyIX -exportToALER <config-file>
```

To export policy data from a policy file to ALER, run policyIX with the `-exportToALER` option and specify a destination file:

```
policyIX -exportToALER <config-file> <policy-file>
```

Importing to ALES from ALER

To import policy data to ALES from ALER, run policyIX with the `-importFromALER` option:

```
policyIX -importFromALER <config-file>
```

To import policy data from ALER to a policy file, run policyIX with the `-importFromALER` option and specify a destination file:

```
policyIX -importFromALER <config-file> <policy-file>
```

You can subsequently import the policy data from the policy file into ALES with a command like:

```
policyIX -import <config-file> <policy-file>
```

For more information, see PolicyIX in the *ALES Adminstration Reference*

Configuration File for ALER Importing and Exporting

The policyIX utility uses a configuration file, the location of which is specified as an argument. A sample policyIX configuration file can be found at *`ALES_ADMIN_HOME`*`/config/policyIX_config.xml`. See PolicyIX: config.xml in the *ALES Adminstration Reference* or the comments in the sample `policyIX_config.xml` file for information about the values to include in your configuration file. In your configuration file, specify the following ALER-related elements:

aler_configuration

A parent element that contains all configuration data needed to connect to ALER, and import and export data from ALER. It contains one or more `aler_property` elements.

aler_property

Specifies a name/value pair of ALER properties. The following property names can be specified in an `aler_property` element:

- `server_url` - ALER connection URL
- `username` - user name to use to connect to ALER
- `userPassword` - user password to connect to ALER
- `assetDescription` - a description of the asset
- `assetName` - name of the asset to export or import
- `importAssetVersion` - Asset version to import; valid only if the `-importFromALER` policyIX option is used.

For example:

Listing 13-1 Configuration for ALER Import and Export

```
<aler_configuration>
    <!-- ALER Connection URL -->
  <aler_property name="server_url"
        value=http://localhost:7101/aler/services/FlashlineRegistry/>
  <!-- User name and password to user to connect to ALER-->
  <aler_property name="userName" value="admin"/>
  <aler_property name="userPassword" value="admin"/>
  <!-- Name of the ALER asset to export/import -->
  <aler_property name="assetName" value="MyALESPolicy"/>
  <aler_property name="assetDescription"
                value="This is an ALES Policy asset"/>

<!-- Asset version to import, only valid if the -importFromAler switch is
used -->
  <aler_property name="importAssetVersion" value="2"/>
</aler_configuration>
```

CHAPTER 14

Integrating with AquaLogic Service Bus

This section covers the following topics:

Introduction

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 ALES WebLogic Server 9.x Security Service Module.

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.

Integrating with AquaLogic Service Bus: Main Steps

This section describes how to integrate AquaLogic Enterprise Security with the AquaLogic Service Bus. Once integrated, you can use the AquaLogic Enterprise Security Administration

Console to write and deploy a set of authorization and role mapping policies to protect ALSB runtime resources.

To integrate AquaLogic Enterprise Security with AquaLogic Service Bus, perform the following tasks:

1. Make sure you have fulfilled all the installation prerequisites described in "Integration Pre-Requisites" on page 14-2.
2. Create, configure, and enroll an instance of the WebLogic Server SSM, as described in "Creating the WebLogic Server SSM Configuration" on page 14-3.
3. Create in the ALES console resources that correspond to the AquaLogic Service Bus resources you want to protect, as described in "Configuring ALSB Resources" on page 14-9.
4. Configure security policies for ALSB, as described in "Configuring ALSB Policies" on page 14-13.
5. Optionally, you can use the ALES PerfDB Auditing provider to verify your configuration, as described in "Verify the Configuration Using the Performance Auditing Provider" on page 14-17.

Integration Pre-Requisites

Before you begin, you must ensure that the following pre-requisites are satisfied:

1. Install WebLogic Server 9.1 or 9.2 in accordance with the *BEA Installation Guide*. This section assumes that you have installed WebLogic Server in `c:\bea920`, the default BEA_HOME directory for WebLogic Server 9.2 on Windows.
2. Install the AquaLogic Enterprise Security Administration Server, as described in *Installing the Administration Server*. This section assumes you have installed the ALES Administration Server in `c:\bea920\ales-22-admin`, the default for ALES 2.5 and WebLogic Server 9.2 on Windows.
3. Install the WebLogic Server 9.x Security Service Module in your WebLogic Server directory, as described in *Installing Security Service Modules*. This section assumes you have installed the WebLogic Server 9.x Security Service Module in `c:\bea920\ales-22-ssm`.
4. Install AquaLogic Service Bus 2.5 in your WebLogic Server directory in accordance with the *AquaLogic Service Bus Installation Guide*.
5. You must have access to an Administration Console that is running on the AquaLogic Enterprise Security 2.5 Administration Server on either the local machine or a remote

machine. This section assumes you are accessing the ALES Administration Console at `https://localhost:7010/asi` and the ALSB Console at `https://localhost:7021/sbconsole`. Replace these URLs with the actual hostnames and port numbers on which you access these consoles.

Creating the WebLogic Server SSM Configuration

Securing ALSB with ALES employs the WebLogic Server 9.x SSM. Integration of ALES with ALSB is not supported for versions of WebLogic Server prior to WebLogic Server 9.1. Install the WebLogic Server 9.x SSM on the machines on which you have installed ALSB, as described in *Installing Security Service Modules*. Configure the WebLogic Server 9.x SSM, as described in the following sections:

Create an Instance of the Security Service Module

Before starting a WebLogic Server Security Service Module, you must first create an instance of the WebLogic Server Security Service Module using the Create New Instance Wizard:

1. Start the ALES Administration Server:
   - On Windows, Start > All Programs > BEA AquaLogic Enterprise Security > Administration Server > Start Server
   - On UNIX, run `ALES-SSM/bin/WLESAdmin.sh start`

2. Run the ALES SSM New Instance Wizard:
   - On Windows, Start > All Programs > BEA AquaLogic Enterprise Security > WebLogic Server 9.x Security Service Module > Create New Instance
   - On UNIX, run `ALES-SSM/wls9-ssm/adm/instancewizard.sh`
3. In the Create New Instance Wizard, enter values for the following:
   - Instance name (default: myrealm)
   - Authorization engine port (default: 8000)
   - Configuration ID (default: myrealm)
   - Enterprise domain (default: asi)

   Click Next.

For more information about creating an instance of a WebLogic Server Security Service Module, see Creating an Instance of a Security Service Module in *Installing Security Service Modules*.

Enroll the Instance of the Security Service Module

After you create the WebLogic Server Security Service Module instance, enroll it with the SCM. You must have the ALES Administration Server running prior to enrolling the Security Service Module. To enroll the WebLogic Server Security Service Module run enroll.bat or enroll.sh. The enroll scripts are found in `ALES-SSM/wls9-ssm/adm/instance`.

For more information about enrolling a security service module, see Enrolling the Instance of the Security Service Module in *Installing Security Service Modules*.

Enable the Console Extension for Security Providers in the WLS 9.x Console

ALES includes an extension to the WebLogic Server 9.x Administration Console. Install the console extension in order for the ALES security providers to be visible in the WebLogic Server 9.x Administration Console.

To install the ALES security provider console extension, copy `ales_security_provider_ext.jar` from `BEA_HOME/ales26-ssm/wls9-ssm/lib` to the `BEA_HOME/WLS_HOME/domains/servicebus/console-ext` directory.

Modify the startWebLogic File

Copy and modify the `startWebLogic.cmd` files present in `BEA-HOME/weblogic92/samples/domains/servicebus` and `BEA-HOME/weblogic92/samples/domains/servicebus/bin`, as described in "Modifying the startWebLogic File" on page 8-12. The files `set-wls-env.bat` and `set-wls-env.sh` are located in the directory `ALES-SSM/wls9-ssm/instance/myrealm/bin`.

Configure ALES Security Providers in the WebLogic Administration Console

An SSM configuration defines the set of security providers to use for adjudication, authentication, auditing, authorization, role mapping, and credential mapping services. This section describes how to use the WebLogic Server Administration Console to configure a set of security providers for AquaLogic Service Bus. After you have completed the steps described in "Integration Pre-Requisites" on page 14-2 and the preceding sections:

1. Start WebLogic Server, using the startWebLogic script you modified in accordance with the instructions in "Modify the startWebLogic File" on page 14-5.
2. Access the WebLogic Server Administration Console, using a browser. The default URL for the console is https://localhost:7001/console .
3. Use the WebLogic Server Administration Console to:
   - "Configure the Security Realm" on page 14-5
   - "Configure a Database Authenticator" on page 14-6
   - "Configure an ASI Authorization Provider" on page 14-6
   - "Replace the Default Adjudicator with the ASI Adjudicator" on page 14-7
   - "Configure an ASI Role Mapper" on page 14-7
   - "Activate Changes" on page 14-7

For more information about creating a SSM configuration, see Configuring and Binding a Security Service Module in *Installing Security Service Modules* and the Console Help. See also Chapter 8, "Configuring the WebLogic Server 9.x SSM."

Configure the Security Realm

1. In the Change Center of the WebLogic Server Administration Console, click Lock & Edit.

2. In left pane of the WebLogic Server Administration Console, click Security Realms and then in the right pane click myrealm.
3. On the Configuration page for the myrealm security realm:
   a. Set Security Model Default to Advanced.
   b. Uncheck Combined Role Mapping Enabled.
   c. Click Save.
   d. Click Advanced.
   e. Set Check Role and Policies to All Web applications and EJBs.
   f. Click Save.

Configure a Database Authenticator

1. In the WebLogic Server Administration Console, select Security Realms > myrealm > Providers > Authentication.
2. Click New.
3. In the Name field, enter `DatabaseAuthenticator`.
4. Select DatabaseAuthenticator from the Type pull down menu and click OK.
5. Select REQUIRED from pulldown menu and click on save.
6. On the Configuration: Provider-Specific page for the DatabaseAuthenticator 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.

Configure an ASI Authorization Provider

1. In the WebLogic Server Administration Console, select Security Realms > myrealm > Providers > Authorization.
2. Click New.
3. In the Name field, enter `ASIAuthorizer`.

4. Select ASIAuthorizationProvider from the Type pull down menu and click OK.
5. On the Configuration: Provider-Specific page for the ASIAuthorizationProvider security provider, set the Application Deployment Parent field to `//app/policy/myrealm` and click Save.

Replace the Default Adjudicator with the ASI Adjudicator

1. In the WebLogic Server Administration Console, select Security Realms > myrealm > Providers > Adjudication.
2. Click Replace.
3. In the Name field, enter `ASIAdjudicator`.
4. Select ASIAdjudicator from the Type pull down menu and click Save.
5. On the Configuration: Provider-Specific page for the ASIAdjudicator security provider, uncheck Require Unanimous Permit and click Save.

Configure an ASI Role Mapper

1. In the WebLogic Server Administration Console, select Security Realms > myrealm > Providers > Role Mapping.
2. Click New.
3. In the Name field, enter `ASIRoleMapperProvider`.
4. Select ASIRoleMapperProvider from the Type pull down menu and click Save.
5. On the Configuration: Provider-Specific page for the ASIRoleMapperProvider security provider, set the Application Deployment Parent field to `//app/policy/myrealm` and click Save.

Activate Changes

To activate the changes to the security realm:

1. In the Change Center of the WebLogic Server Administration Console, click Activate Changes.
2. Shut down the AquaLogic Service Bus domain.

Configure ALES Security Providers in the ALES Administration Console

After you configure your security providers in the WebLogic Server Administration Console, you need to take some additional steps to configure them using the AquaLogic Enterprise Security Administration Console. The console's default URL is `https://localhost:7010/asi` and its default user name and password are `system` and `weblogic`. Use the ALES console to:

- "Create the weblogic User" on page 14-8
- "Create a New SSM Configuration" on page 14-8
- "Bind the Configuration to the SCM" on page 14-9

Create the weblogic User

Create a user named `weblogic`.

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

Create a New SSM Configuration

In the ALES Administration Console, create a new configuration named myrealm, including the ASI Authorization provider and the ASI Role Mapper:

1. In the left pane, click Unbound Configuration.
2. Click Create a new Security Service Module Configuration.
3. In the Configuration ID field, enter `myrealm` and click Create.
4. Select Providers > Authorizers.
5. Click Configure a new ASI Authorization Provider and click Create.
6. In the Details tab for the ASI Authorization Provider, set the Application Deployment Parent field to `//app/policy/myrealm` and click Apply.

7. Select Unbound Configurations > myrealm > Role Mapping.
8. Click Configure a new ASI Role Mapper Provider and click Create.
9. In the Details tab for the ASI Authorization Provider, set the Application Deployment Parent field to `//app/policy/myrealm` and click Apply.

Bind the Configuration to the SCM

In the ALES Administration Console, bind the new security configuration to the Service Control Manager:

1. Select Security Configuration > Service Control Managers > adminconfig.
2. Click adminconfig in the left pane and click the Bindings tab in the right pane.
3. Select `myrealm` from the dropdown menu and click Bind.

Configuring Resources and Policies for ALSB

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.

This section covers the following topics:

Configuring ALSB Resources

This section describes how to use the ALES Administration Console to define the application resources that you will protect using ALES.

Creating a 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

Creating a Virtual Resource

To create a virtual resource named `xyz`:

1. Create a resource as described in "Creating a Regular Resource" on page 14-9.
2. Right click the resource `xyz` and select Configure Resource.
3. Check the Allow Virtual Resources box and click OK.

Creating the 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
: The full name of the proxy service, for example: path=project/folder1/folder2

proxy
: The 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 the 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 14-1 describes how ALSB Proxy Service reference elements map to ALES resource and privilege elements

Table 14-1 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`.

Creating a 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 drop-down list box, select Binding.
4. Check the Distribution Point box.
5. After the resource is created, right click the resource and select Configure Resource.
6. Select Binding application from the pull-down menu and click OK.

Creating a Resource Tree

Select Resources on the left pane and create a resource tree as shown in Listing 14-1:

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

Note: Pay extra attention to entering the resource names correctly, any mistake will result in incorrect configuration

Listing 14-1 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*.

Configuring ALSB Policies

The ALES Administration Server installation includes a set of sample polices for BEA AquaLogic Service Bus, located at `BEA_HOME/ales26-admin/examples/policy/alsb_sample_policy`. You can import these sample policies and use them as a starting point for developing a full set of policies for your applications. 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 Policy Examples" on page 14-13
- "Role Mapping Policy Examples" on page 14-15

Authorization Policy Examples

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

```
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 the top of the window.
3. Click any from the list and click Add.
4. Click Resources tab and expand myrealm > shared.
5. Select `adm` and click Add, then select `svr` and click Add.
6. Click the Policy Subjects tab, click Admin and then click Add.

7. Make sure that the data is correct and click OK.

The following policy 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;
```

To add this policy:

1. Select Policy > Authorization Policies and click New.
2. Check grant the top of the window.
3. Click any from the list and click Add.
4. Click Resources tab and expand myrealm > shared.
5. Select in turn each of eis, ejb, jdbc, jms, jndi, url, webservices and workcontext and click Add.
6. Click the Policy Subjects tab, click Everyone and then click Add.
7. Make sure that the data is correct and click OK.

The following policy grants all users access to the ProxyServices resource:

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

To add this policy:

1. Select Policy > Authorization Policies and click New.
2. Check grant the top of the window.
3. Click access from the list and click Add.

4. Click Resources tab and expand myrealm > shared > ProxyService > MortgageBroker.
5. Select ProxyServices and click Add.
6. Click the Policy Subjects tab, click Everyone and then click Add.
7. Make sure that the data is correct and click OK.

Role Mapping Policy Examples

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.
2. Click New
3. In the Available Roles list, click Admin.
4. Click Add.
5. Click the Resources tab.
6. In the Available Resources list, click myrealm.
7. Click Add.
8. Click the Policy Subjects tab
9. In the Select Policy Subjects from: pulldown menu, select Users.
10. Select weblogic.
11. Click Add.
12. Make sure that the data is correct and click OK.

The following policy grants the user `anonymous` the role `Anonymous` over the resource `myrealm`:

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

To add this policy:

1. Select Policy > Authorization Policies.

2. Click New
3. In the Available Roles list, click Anonymous.
4. Click Add.
5. Click the Resources tab.
6. In the Available Resources list, click myrealm.
7. Click Add.
8. Click the Policy Subjects tab
9. In the Select Policy Subjects from: pulldown menu, select Users.
10. Select anonymous.
11. Click Add.
12. Make sure that the data is correct and click OK.

The following policy grants the group of all users the role `Everyone` over the resource `myrealm`:

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

To add this policy:

1. Select Policy > Authorization Policies.
2. Click New
3. In the Available Roles list, click Everyone.
4. Click Add.
5. Click the Resources tab.
6. In the Available Resources list, click myrealm.
7. Click Add.
8. Click the Policy Subjects tab
9. In the Select Policy Subjects from: pulldown menu, select Groups.
10. Select anonymous.
11. Click Add.

12. Make sure that the data is correct and click OK.

Distribute Changes

After you have made changes to the configuration and policies in the ALES console, distribute the changes:

1. Click Deployment in the left pane.
2. Click Configuration in the right pane.
3. Select Security Configurations and click Distribute Configuration Changes.

Once the distribution of the Security Configurations reaches 100% complete, distribute the policy changes:

1. Click Deployment in the left pane.
2. Select Policy and click Distribute Policy.

Once the distribution of the policy reaches 100% complete:

1. Start the myrealm ARME instance that is used to protect the ALSB domain. On Windows, select Start > All Programs > BEA AquaLogic Enterprise Security > Security Service Module > Weblogic 9.x Server Securiity Service Module > myrealm > Start ARME(console mode)
2. Start the ALSB domain:

   ```
   BEA_HOME\weblogic92\samples\domains\servicebus\startWebLogicALES.cmd
   ```

   This step can take several minutes to complete.

Now the AquaLogic Service Bus domain is protected by the AquaLogic Enterprise Security WebLogic 9.x SSM.

Verify the Configuration Using the Performance Auditing Provider

This step is optional. If you like, you use the ALES performance auditing provider to verify that the AquaLogic Enterprise Security SSM has been properly configured to protect your ALSB installation.

The PerfDBAuditor is an ALES audit provider which collects statistics about requests routed through ALES. After you configure a PerfDBAuditor in your ALSB security realm, you can

examine the database tables. For more information about the PerfDBAuditor provider, see Performance Statistics in the *Administration and Deployment Guide*.

To use the PerfDBAuditor to verify your configuration, follow the procedures in the following sections:

- "Configure the PerfDBAudit Provider" on page 14-18
- "Restart the Domain" on page 14-18
- "Generate Data" on page 14-19

Configure the PerfDBAudit Provider

Using the WebLogic Server Administration Console, configure the ALES Performance DB Audit provider:

1. In the WebLogic Server Administration Console, select Security Realms > myrealm > Providers > Auditing
2. Click New.
3. In the Name field, enter `PerfDBAuditor`.
4. Select PerfDBAuditor from the Type pull down menu and click OK.
5. 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).

   Click on save
6. Click Activate Changes.

Restart the Domain

Stop the server by running
`BEA_HOME/weblogic92/samples/domains/servicebus/bin/stopWebLogic.sh`

Restart the server by running
`BEA_HOME/weblogic92/samples/domains/servicebus/startWebLogicALES.cmd`

Generate Data

Generate some performance data and check it:

1. In Internet Explorer, open the ALSB example application at `http://localhost:7021/examplesWebApp/index.jsp`.
2. Click Reload the examples.
3. Under Run the AquaLogic Service Bus Examples, click Run the Example.
4. Click Submit Loan Application.

After a few minutes, check the database table PERF_ATZ_STAT, which is populated with authorization statistics. You should see a non-zero value under TOTALREQ. This indicates that access to the ALSB example application is protected by the AquaLogic Enterprise Security SSM.

CHAPTER 15

Enabling SAML-based Single Sign-On

The ALES provides support for the producing and consuming SAML 1.1 assertions, and for sending/receiving them using the Browser/POST Profile.

This section covers the following topics:

Overview

The use of SAML assertions allows servers in different domains to operate in a federation of trusted servers and grant access to users based on a single login to one of the servers. In a given federation, there are SAML 'producers' and SAML 'consumers'. The SAML providers authenticate users and generate assertions attesting to the user's identity. The SAML assertion can then be included in user requests to other servers in the federation, making additional logins unnecessary.

ALES SSMs allow a Microsoft IIS or Apache HTTP server to operate as a SAML producer or a SAML consumer (or both) and send/receive SAML assertions using the Browser POST Profile. Note that Browser Artifact Profile is not currently supported.

When set up as a SAML consumer, the SSM running on an IIS or Apache HTTP server will accept requests containing assertions and then use its SAML Identity Asserter to validate the assertions.

Configuring ALES as a SAML Assertion Consumer

When serving as a SAML consumer, the SSM receives requests specifying a protected resource and SAML assertion attesting to the user's validity. The SSM's SAML Identity Asserter accepts the SAML token and returns the corresponding user. ALES will grant access to the resource based on any policies associated with the resource and/or users role.

To configure a IIS or Apache SSM as a SAML consumer:

Note: It is assumed that the necessary Resources and Policies governing access to protected resources have been established.

1. Using the Administration Console, create or use an existing SSM configuration defining a SAML Identity Asserter. The SAML Identity Asserter consumes SAML assertions and returns the corresponding authenticated subjects.

Note: The trusted keystore configured for the SAML Identity Asserter must contain the certificate used to sign the Assertion and the certificate that signed that certificate up to the trust anchor. If the trust anchor is a well known CA such as Verisign, the keystore does not have to contain the trust anchor certificate.

2. Install the ALES SCM and SSM on the IIS or Apache web server.

3. Create instances of the Web Service SSM and the Web Server SSM on the web server and configure the web server to call the SSM. Set the SSMs to use the certificate authority keystore. Set the password for the SSM to use when logging in to the ASI database.

4. Set up a file to serve as the target POST URL. The file can be copied to the web server or referred to using a virtual server. It serves as a placeholder that alerts the SSM to receive a SAML assertion. It must be a valid HTML file, but requires nothing more than empty `<HTML>` and `<BODY>` tags. The SSM provides a template file named `SAMLIn.acc` (IIS) or `SAMLIn.html` (Apache) in the `templates` directory.

5. In SSM's `default.properties` file, enable the `set saml.incoming.enable` parameter to 'true'.

   Example: `set saml.incoming.enable=true`

6. In SSM's `default.properties` file, set the `saml.incoming.url` parameter to the POST URL you established on the server (see step 4). Make sure you create a policy that allows POST to the SAML consumer URL.

   Example: `saml.incoming.url=http://<server>/<dir>/SAMLIn.acc`

Configuring ALES as a SAML Assertion Producer

When operating a SAML producer, the SSM will receive requests for a SAML assertion. The SSM's Authentication Provider authenticates the user and its SAML Credential Mapper returns a SAML assertion. The SSM then sends a response contain the SAML assertion using the Browser POST Profile.

By default, SAML assertions produced by ALES are 64-base encoded tokens identifying the principals. They include an XML Signature proving that the assertion has not been tampered with in transit from the sender to the provider, and will contain group information about the user if that information is available. Note that these assertions do *not* contain the certificate chain used for signing the assertion. It is up to the SAML consumer to notify the recipient of the certificate that can be used to verify the XML signature.

To configure a IIS or Apache SSM as a SAML Producer:

1. Using the Administration Console, create a SSM configuration defining an a Authentication Provider and a SAML Credential Mapper.
2. Install the ALES SCM and SSM on the IIS or Apache web server.
3. Create instances of the Web Service SSM and Web Server SSM on the web server. Set the SSMs to use the certificate authority keystore. Set the password for the SSM to use when logging in to the ASI database.
4. Configure the IIS or Apache web server to integrate with the SSM.
5. Configure the script for handling the Browser POST to the SAML consumer.

Notes: The SSM's `template` directory contains a file named `SAMLXfer.acc` (IIS) or `SAMLXfer.dhtml` (Apache) that can be used.

Make sure you create a policy allowing Everyone access to the script file.

CHAPTER 16

Enabling SPNEGO-based Single Sign-on

This section covers the following topics:

Configuring Single Sign-On with Microsoft Clients

Using a Single Pass Negotiate Identity Asserter shipped with AquaLogic Enterprise Security, you can achieve cross-platform authentication, single sign-on (SSO), integration with Microsoft Internet Explorer browser clients and Microsoft .NET web services.

Cross-platform authentication is achieved by emulating the negotiate behavior of native Windows-to-Windows authentication services. The servlet container in this release was modified to handle the necessary header manipulation required by the Windows negotiate protocol, also known as Simple and Protected Negotiate (SPNEGO).

The negotiate identity asserter is an implementation of the Security Service Provider Interface (SSPI) as defined by the WebLogic Security Framework and provides the necessary logic to authenticate a client based on the client's SPNEGO token.

For more information, see the following topics:

Requirements

The environment for cross-platform authentication requires the following components.

- Domain controller back-end system
  - Windows 2000 or greater Active Directory
  - Service accounts for mapping Kerberos services
  - Service Principal Names (SPNs) properly configured
  - Keytab files created and inserted, based on platform environment of your server system
- AquaLogic Enterprise Security application server
  - AquaLogic Enterprise Security installed
  - WebLogic Security Service Module installed and an instance created
  - WebLogic application server configured to use Active Directory realm
  - Keytab import and configuration
  - MIT Kerberos V5 Generic Security Service API (GSS-API)
- Client systems
  - .NET Framework 1.1
  - Windows 2000 Professional SP2 or greater
  - Internet Explorer 5.01, Internet Explorer v5.5 SP2, Internet Explorer 6.0 SP1
  - Proper configuration of the Internet Explorer browser
  - Proper configuration of web services clients

Note: Client users must be logged on to a Windows 2000 domain or realm, having acquired Kerberos credentials from the Active Directory domain. Local logons do not work.

The following sections describe how to configure the SPNEGO provider and how to set up the necessary components.

- "Enabling a Web Service or Web Application" on page 16-3
- "Configure the Client .NET Web Service" on page 16-7
- "Configure the Internet Explorer Client Browser" on page 16-8

Enabling a Web Service or Web Application

To enable a particular web service, web application, or other protected resource for single sign-on, you must use the Single Pass Negotiate Identity Asserter provider in conjunction with client certification set as the login configuration in your standard J2EE `web.xml` descriptor file.

For configuration instructions, see the following topics:

- "Configuring the SPNEGO Security Provider" on page 16-3
- "Editing the Descriptor File" on page 16-3

Configuring the SPNEGO Security Provider

Configure the Single Pass Negotiate Identity Assertion provider using the Administration Console. To configure the provider, create an instance of the SPNEGO provider for the WebLogic Server 8.1 Security Service Module.

Editing the Descriptor File

Listing 16-1 shows a sample `web.xml file` for a protected WebLogic Web Service resource (Conversation) with the login configuration set to `CLIENT-CERT`.

Listing 16-1 Sample Web.xml File

```
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application
2.3//EN" "http://java.sun.com/dtd/web-app_2_3.dtd">

<web-app>
<security-constraint>
       <display-name>Security Constraint on Conversation</display-name>
       <web-resource-collection>
           <web-resource-name>Conversation web service</web-resource-name>
            <description>Only those granted the ConversationUsers role may
             access the Conversation web service.</description>
            <url-pattern>/async/Conversation.jws/*</url-pattern>
            <http-method>GET</http-method>
            <http-method>POST</http-method>
        </web-resource-collection>
        <auth-constraint>
            <role-name>ConversationUsers</role-name>
```

```
        </auth-constraint>
    </security-constraint>

    <login-config>
      <auth-method>CLIENT-CERT</auth-method>
    </login-config>

    <security-role>
        <description>Role description</description>
        <role-name>ConversationUsers</role-name>
    </security-role>
</web-app>
```

You can use any role to protect your web resource collection. You want to make the corresponding security and run-as role assignments in your `weblogic.xml` descriptor as needed. Continuing the example, Listing 16-2 shows a sample `weblogic.xml` file.

Listing 16-2 Sample weblogic.xml File

```
<?xml version="1.0" encoding="UTF-8"?>

  <!DOCTYPE weblogic-web-app
    PUBLIC "-//BEA Systems, Inc.//DTD Web Application 7.0//EN"
    "http://www.bea.com/servers/wls700/dtd/weblogic700-web-jar.dtd" >

<weblogic-web-app>
    <security-role-assignment>
        <role-name>ConversationUsers</role-name>
        <principal-name>weblogic</principal-name>
    </security-role-assignment>
</weblogic-web-app>
```

For more information on configuring protected resources for WebLogic Server, see Securing WebLogic Resources in the BEA WebLogic Server 8.1 documentation set available on the Web at `http://e-docs.bea.com/wls/docs81/`.

Configuring Active Directory Authentication

To configure Active Directory authentication, perform the following tasks:

Utility Requirements

This procedure requires the use of the following Active Directory utilities:

- `setspn` – found on the Windows 2000 Resource Kit
- `ktpass` – found on the Windows 2000 distribution CD in `\Program Files\Support Tools`

Configuring and Verifying Active Directive Authentication

The first three steps of this procedure assume that you have two domains: one represents the WebLogic application server domain (`bea.com`) and one controlled by Active Directory (`magellan.corp`).

1. Create a user account for the hostname of the web server machine in Active Directory, by using the Active Directory Users and Computers Snap-in.

   Click Start->Programs->Administrative Tools->Active Directory Users and Computers.

   Use the simple name of the WebLogic server host. For example, if the host you are running the WebLogic application on is called `myhost.bea.com`, create a new user in Active Directory called `myhost`. Do not select "User must change password at next logon." Make a note of the password for use in step 3.

2. Create the Service Principal Names (SPNs) for this account:

   ```
   setspn -A host/myhost.bea.com myhost
   setspn -A HTTP/myhost.bea.com myhost
   ```

3. Create your user mapping and export the keytab files using the `ktpass` utility:

```
ktpass -princ host/myhost@MAGELLAN.CORP -pass <password> -mapuser myhost
-out c:\temp\myhost.host.keytab

ktpass -princ HTTP/myhost@MAGELLAN.CORP -pass <password> -mapuser myhost
-out c:\temp\myhost.HTTP.keytab
```

Note: If you generated the keytab files for a WebLogic server on a UNIX host, copy the keytab files securely to the UNIX host. Login as root and then merge them into a single keytab using the ktutil utility:

```
ktutil: "rkt myhost.host.keytab"
ktutil: "rkt myhost.HTTP.keytab"
ktutil: "wkt mykeytab"
ktutil: "q"
```

If your WebLogic Server instance is running on a Windows platform, generate the keytab from that machine using the ktab.

4. Verify that authentication works.
   - To verify that Kerberos authentication is working on the UNIX system, run the `kinit` utility:

     ```
     kinit -t mykeytab myhost
     ```

     You are prompted for the password and if authentication succeeds, the command prompt returns without an error message.
   - To verify that Kerberos authentication is working on a Windows system, use the `ktab` utility locally on the WebLogic server host to create the keytab file in the WebLogic server domain directory:

     ```
     setEnv
     ktab -k mykeytab -a myhost@MAGELLAN.CORP <password>
     ```

5. Create a JAAS login configuration file.

   For either a Windows or a UNIX server host, you need a JAAS login configuration file. You also need to set some system properties to direct WebLogic server to allow the proper Kerberos authentication to occur. A sample login configuration file called `krb5Login.conf` looks like this:

   ```
   com.sun.security.jgss.initiate
   {
   com.sun.security.auth.module.Krb5LoginModule required principal=
   "myhost@MAGELLAN.CORP" useKeyTab=true keyTab=mykeytab storeKey=true;
   };
   com.sun.security.jgss.accept
   {
   com.sun.security.auth.module.Krb5LoginModule required principal=
   ```

```
"myhost@MAGELLAN.CORP" useKeyTab=true keyTab=mykeytab storeKey=true;
};
```

6. Add the following system properties to the start line of your WebLogic server:

```
-Djava.security.krb5.realm=MAGELLAN.CORP
-Djava.security.krb5.kdc=ADhostname
-Djava.security.auth.login.config=krb5Login.conf
-Djavax.security.auth.useSubjectCredsOnly=false
```

For a web service client, complete the steps described in "Configure the Client .NET Web Service" on page 16-7. For Internet Explorer configuration, complete the steps described in "Configure the Internet Explorer Client Browser" on page 16-8.

Configure the Active Directory Authentication Provider

To populate groups properly in the authenticated subject and to use the keystores, you must configure the Active Directory Authentication Provider.

Configure the Client .NET Web Service

If you are configuring the client .NET web service, perform the following steps:

1. Open the `web.config` file for the client web service.
2. Set the authentication mode to Windows for IIS and ASP.NET. This is usually the default.

   ```
   <authentication mode="Windows" />
   ```

3. Add the statement needed for the web services client to pass to the proxy web service object so that the credentials are sent through SOAP.

   For example, if you have a web service client for the conversation web service represented by the proxy object `conv`, then setting the web services client credentials in C# looks like this:

   ```
   /*
   * Explicitly pass credentials to the Web Service
   */
   conv.Credentials = System.Net.CredentialCache.DefaultCredentials;
   ```

Configure the Internet Explorer Client Browser

If you are configuring Internet Explorer, perform the following steps:

Configure the Sites

To configure the sites:

1. In Internet Explorer, click Tools, and then click Internet Options.
2. Click the Security tab.
3. Click Local intranet.
4. Click Sites.
5. Ensure that the Include all sites that bypass the proxy server check box is checked, and then click Advanced.
6. In the Local intranet (Advanced) dialog box, enter all relative domain names that will be used on the intranet (e.g. myhost.bea.com).
7. Click OK to close the dialog boxes.

Configure Intranet Authentication

To configure Intranet Authentication:

1. Click the Security tab, click Local intranet, and then click Custom Level.
2. In the Security Settings dialog box, scroll down to the User Authentication section of the list.
3. Select Automatic logon only in Intranet zone. This setting prevents users from having to re-enter logon credentials; a key piece to this solution.
4. Click OK to close the Security Settings dialog box.

Verify the Proxy Settings

To verify the Proxy Settings:

1. In Internet Explorer, click Tools, and then click Internet Options.
2. Click the Connections tab.
3. Click LAN Settings.
4. Verify that the proxy server address and port number are correct.
5. Click Advanced.
6. In the Proxy Settings dialog box, ensure that all desired domain names are entered in the Exceptions field.
7. Click OK to close the Proxy Settings dialog box.

Set the Internet Explorer 6.0 Configuration Settings

In addition to the previous settings, one additional setting is required if you are running Internet Explorer 6.0.

1. In Internet Explorer, click Tools, and then click Internet Options.
2. Click the Advanced tab.
3. Scroll down to the Security section.
4. Make sure that Enable Integrated Windows Authentication (requires restart) is checked, and then click OK.
5. If this box was not checked, restart the browser.

CHAPTER 17

Authorization Caching

Authorization caching allows the ASI Authorization and ASI Role Mapper providers to cache the result of an authorization call, and use that result if future calls are made by the same caller. The authorization cache automatically invalidates itself if there is a policy or user profile change. This section covers the following topics:

AquaLogic Enterprise Security also supports use of two other caches:

- Client-side authorization caching in Web Services clients. For more information, see Using the Web Services Client Authorization Cache in *Programming Security for Web Services*.
- Server-side identity caching in the Web Services SSM. For more information, see Using the Web Services SSM Identity Cache in *Programming Security for Web Services*

Authorization Cache Operation

The entry keys for elements in the authorization cache are constructed from these elements:

- Subject
- Resource
- Privilege (for authorization only)

- Roles (for authorization only)
- Context

The key construction algorithms are different for the authorization cache and the role mapping cache. The key-value pairs for entries in the authorization cache are:

```
Authorization: Map (key =  Subject + Resource + Privilege + Roles + Contexts,
value = Result + ResponseAttributes + Roles)
```

The key-value pairs for entries in the role mapping cache are:

```
Role mapping: Map (key =  Subject + Resource + Contexts, value = Roles)
```

Note that `Contexts` and `ResponseContexts` can be single values or lists.

Note: ALES versions prior to 2.5 used a Pre Load Attributes configuration parameter. You may need to use this option if you use the latest version of ALES to manage ALES version 2.2 or 2.1 SSMs. For more details on using the Pre Load Attribute caching option see Authorization Caching in the ALES 2.2 documentation.

Authorization performance of the ASI Authorization Provider can be improved further by setting the "Lazy Role Provider" switch (which can be found in the ALES Administration Console on the Performance tab of ASI Role Mapper Provider). If the switch is set, the authorization service will not make an extra call to acquire the list of roles that corresponds to a particular combination of subject, resource and context; instead, it will delegate this function to the ARME native process in a single call when a request for making the authorization decision is made.

The authorization cache is a per session object. This means that a separate cache object is created every time a client establishes the connection. If two Java-SSM clients have established connections, a separate cache object is created for each of them, even if the same credentials and SSM instance were used.

Configuring Authorization Caching

Authorization caching is on by default. It may be configured from within the Administration Console through the ASI Authorization and ASI Role Mapper provider configuration. Table 17-1 lists the switches affect the authorization cache.

Table 17-1 Authorization Caching

Setting	Default Value	Description
AccessAllowedCaching	true	Enables/disables caching of authorization decisions.
GetRolesCaching	true	Enables/disables caching of role mapping decisions.
SessionExpiration	60	Specifies the number of seconds that authorization decisions for a user will be cached in memory. Upon expiration, the cached information is cleared and then updated if the user makes a subsequent request. While increasing this value can improve performance, it may also reduce security by making authorization decisions based on outdated information.
SubjectDataCacheExpiration	60	Defines how long user profile data will be cached. Cached authorization decisions are reset each time this data cache expires. You can increase this value to improve performance.

Table 17-1 Authorization Caching (Continued)

Setting	Default Value	Description
Pre Load Attributes	adaptive -private	Not used as of ALES version 2.5. Determines whether the provider loads ContextHandler data before starting to evaluate policy or waits for a callback to ask for specific items. Pre-loading attributes can dramatically improve performance in policies that use contextual attributes. **Note:** ALES versions prior to 2.5 used the Pre Load Attributes configuration parameter. You may need to use this option if you use the latest version of ALES to manage ALES version 2.2 or 2.1 SSMs. For more details on using the Pre Load Attribute caching option see Authorization Caching in the ALES 2.2 documentation.
Lazy Role Provider	true	When a request for list of roles is made, determines indicates whether a call should be made to the Role Mapping Cache / ARME or whether the call may be postponed until the returned list of roles is in fact used. Setting this true provides significant performance improvement when used in combination with the ASI Authorization provider since the provider can request the roles and the authorization decision in a single call.

The properties listed in Table 17-2 can be entered as advanced configuration properties to further tune the cache.

Table 17-2 Advanced Configuration Properties

Setting	Default Value	Description
ASI.AuthorizationCacheLimit	1000	Determines the maximum number of cached decisions per user session. Once exceeded, old cached values are overwritten.
ASI.AuthorizationCacheDynamicAttributeLimit	10	Determines the maximum number of context attributes a decision may use and still be stored in the cache.
ASI.PolicyCacheInvalidatorPollingInterval	1000	Determines how often the cache checks for policy distributions. The value is in milliseconds

Authorization Caching Expiration Functions

There is a small subset of data that may change without the knowledge of the cache. This includes internally computed time values, as well as custom evaluation plug-ins. Because the cache is not aware of changes in these values, it does not automatically invalidate a cached decision when they change. For this reason a series of evaluation functions is provided to control the period of cache validity. These functions are only needed in policies that make explicit use of internally computed time values or custom evaluation plug-ins.

Table 17-3 lists the internally computed time values. If these values are referenced in a policy, you should also explicitly set the cache validity for the policy.

Table 17-3 Time Values Used with Expiration Functions

Credential	Value	Range or Format
time24	integer	0–2359
time24gmt	integer	0–2359
dayofweek	Dayofweek_type	Sunday–Saturday
dayofweekgmt	Dayofweek_type	Sunday–Saturday
dayofmonth	integer	1–31

Table 17-3 Time Values Used with Expiration Functions (Continued)

Credential	Value	Range or Format
dayofmonthgmt	integer	1–31
dayofyear	integer	1–366
dayofyeargmt	integer	1–366
daysinmonth	integer	28–31
daysinyear	integer	365–366
minute	integer	0–59
minutegmt	integer	0–59
month	month_type	January–December
monthgmt	month_type	January–December
year	integer	0–9999
yeargmt	integer	0–9999
timeofday	time	HH:MMAM" or "HH:MMPM"
timeofdaygmt	time	HH:MMAM" or "HH:MMPM"
hour	integer	0–23
hourgmt	integer	0–23
date	Date	MM/DD/YYYY"
dategmt	Date	MM/DD/YYYY"

Table 17-4 lists the expiration functions for the authorization cache. You can use these functions to set an expiration time for the decision. This way you can instruct the cache to only hold the value for a given period of time, or to not hold it at all. These functions correspond roughly to each of the internally computed time types.

Table 17-4 Expiration Functions

Function	Argument	Description
valid_for_mseconds	integer	Valid for a given number of milliseconds
valid_for_seconds	integer	Valid for a given number of seconds
valid_for_minutes	integer	Valid for a given number of minutes
valid_for_hours	integer	Valid for a given number of hours
valid_until_timeofday	time	Valid until the specified time on the date the evaluation is performed
valid_until_time24	integer	Valid until the specified time on the date the evaluation is performed
valid_until_hour	integer	Valid until the specified hour on the date the evaluation is performed
valid_until_minute	integer	Valid until the specified minute of the hour the evaluation is performed
valid_until_date	Date	Valid until the specified date
valid_until_year	integer	Valid until the specified year
valid_until_month	month_type	Valid until the specified month of the year the evaluation is performed
valid_until_dayofyear	integer	Valid until the specified day of the year the evaluation is performed
valid_until_dayofmonth	integer	Valid until the specified day of the month the evaluation is performed
valid_until_dayofweek	Dayofweek_ type	Valid until the specified day of the week the evaluation is performed
valid_until_timeofday_gmt	time	Valid until the specified time on the date the evaluation is performed in GMT time.
valid_until_time24_gmt	integer	Valid until the specified time on the date the evaluation is performed in GMT time.

Table 17-4 Expiration Functions (Continued)

Function	Argument	Description
valid_until_hour_gmt	integer	Valid until the specified minute of the hour the evaluation is performed in GMT time
valid_until_minute_gmt	integer	Valid until the specified minute of the hour the evaluation is performed in GMT time.
valid_until_date_gmt	Date	Valid until the specified date in GMT time.
valid_until_year_gmt	integer	Valid until the specified year in GMT time.
valid_until_month_gmt	month_type	Valid until the specified month of the year the evaluation is performed in GMT time.
valid_until_dayofyear_gmt	integer	Valid until the specified day of the year the evaluation is performed in GMT time.
valid_until_dayofmonth_gmt	integer	Valid until the specified day of the month the evaluation is performed in GMT time.
valid_until_dayofweek_gmt	Dayofweek_ type	Valid until the specified day of the week the evaluation is performed in GMT time.

For example, if you had the following policy:

```
GRANT(//priv/order,//app/resturant/breakfast,//sgrp/customers/allusers/)
if hour < 11;
```

When authorization caching is enabled, you write the policy as:

```
GRANT(//priv/order,//app/resturant/breakfast,//sgrp/customers/allusers/)
if hour < 11 and valid_until_hour(11);
```

With authorization caching, the result of this policy is cached in the provider until 11:00 AM, at which time, it expires. Not calling `valid_until_hour` argument results in this policy being cached until the next policy distribution. Therefore, if you are using authorization caching, it is important to update your time dependent policies appropriately.

APPENDIX A

AquaLogic Enterprise Security Adapter for Sun Identity Manager

The AquaLogic Enterprise Security Adapter is a plug-in to the Sun Identity Manager that enables the propagation of users and their attributes between Sun Identity Manager and AquaLogic Enterprise Security in a bi-directional way. The adapter is available with AquaLogic Enterprise Security 2.2 CP4 or higher.

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 AquaLogic Enterprise Security, and the user operations in the AquaLogic Enterprise Security Administration console will be synced into Sun Identity Manager. The sync interval from AquaLogic Enterprise Security 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 `ales26-admin` to `idm/WEB-INF/lib`:
   - `ales26-admin/lib/asi_classes.jar`
   - `ales26-admin/lib/asitools.jar`
   - `ales26-admin/lib/jsafeJCE.jar (WLS 8.x) or jsafeJCEFIPS.jar (WLS 9.x)`

- ales26-admin/lib/log4j.jar
- ales26-admin/lib/ssladapter.jar
- ales26-admin/lib/sslplus.jar
- ales26-admin/lib/webservice.jar
- ales26-admin/lib/webserviceclient.jar
- ales26-admin/lib/providers/ojdbc14_g.jar
- ales26-admin/lib/providers/jconn2.jar
- ales26-admin/lib/providers/jconn3.jar
- ales26-admin/data/SunIMAdapter/lib/ALESResourceAdapter.jar

3. Copy `ales26-admin/data/SunIMAdapter/forms/*` to `idm/sample/forms`.
4. Copy `ales26-admin/data/SunIMAdapter/images/ALES.gif` to `idm/applet/image`.
5. Add execute permission for the following scripts on UNIX platforms:
   - ales26-admin/bin/install_user_change_schema_oracle.sh
   - ales26-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:

   ales26-admin/bin/install_user_change_schema_oracle.bat|sh

   For Sybase, run:

   ales26-admin/bin/install_user_change_schema_sybase.bat|sh

   You need to supply your ALES database credentials in order for the scripts to make changes to the ALES database.
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 every user-related operation in the ALES system to the ALES database. 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 database 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.x SSM

1. Start the ALES Administration servers.
2. Log in to the ALES admin console, https://hostname:port/asi.
3. Create UserChangeDBAuditor. There should be no more than one User Change DB Auditor in one ALES domain.
   a. Click on Security Configuration.
   b. Click Security Control Managers.
   c. Click asiadmin node under adminconfig.
   d. On the Providers tab of the asiadmin configuration, click the Auditors tab.
   e. Click Configure a new User Change DBAuditor.
   f. Accept the default name and click Create.
   g. Click on the Details tab and enter the JDBC parameters. The values should equal those of the ALES database configuration.
   h. Click Apply.
4. On the tree menu on the left side of console, Click on Deployment.
5. Click on the Configuration tab.
6. Check the Security Configuration changes check box, and then click Distribution Configuration changes.
7. Click Refresh until this distribution is 100% complete.
8. Restart the ALES Administration server to make the UserChangeDBAuditor take effect.

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.