BEA AquaLogic™ Enterprise Security

Introduction

Version 3.0
Revised: December 2007

Contents

4. Security Services

5. Security Administration

CHAPTER 1

Introduction and Roadmap

The following sections describe the content and organization of this document:

Audience and Scope

This document summarizes the features of the BEA AquaLogic™ Enterprise Security products and presents an overview of the architecture and capabilities of the security services. It provides a starting point for understanding the family of BEA AquaLogic Enterprise Security products.

The document is intended for all users of the ALES product family, including business analysts, security architects, security developers, application developers, and administrators.

Guide to This Document

This document is organized as follows:

- Chapter 2, "Overview of BEA AquaLogic Enterprise Security," describes ALES products, services, features, and functionality.
- Chapter 3, "Architecture Overview," describes the ALES product architecture and components.

- Chapter 4, “Security Services,” describes security services, including auditing, authentication, authorization and role mapping, and credential mapping.
- Chapter 5, “Security Administration,” describes policies and how you use the Administration Server to design policies.

Related Information

Additional ALES documentation includes:

- *WSDL Documentation for the Web Service Interfaces*—Provides reference information for the ALES Web Services Interfaces.
- *Policy Managers Guide*—Describes how to write and manage security policies for ALES.
- *SSM Installation and Colinfiguration Guide*—Describes how to install ALES Security Services Modules.
- *Developing Security Providers* —Describes how to develop custom security providers.
- *Javadocs for Security Service Provider Interfaces*—Provides reference information for the Security Service Provider Interfaces.
- *Programming Security for Java Applications*—Describes how to implement ALES security in Java applications.
- *Javadocs for Java API*—Provides reference information for the ALES Java Application Programming Interfaces.

CHAPTER 2

Overview of BEA AquaLogic Enterprise Security

This document summarizes the features of the BEA AquaLogic® Enterprise Security™ products and presents an overview of the architecture and capabilities of the security services.

This chapter covers the following topics:

What Is BEA AquaLogic Enterprise Security?

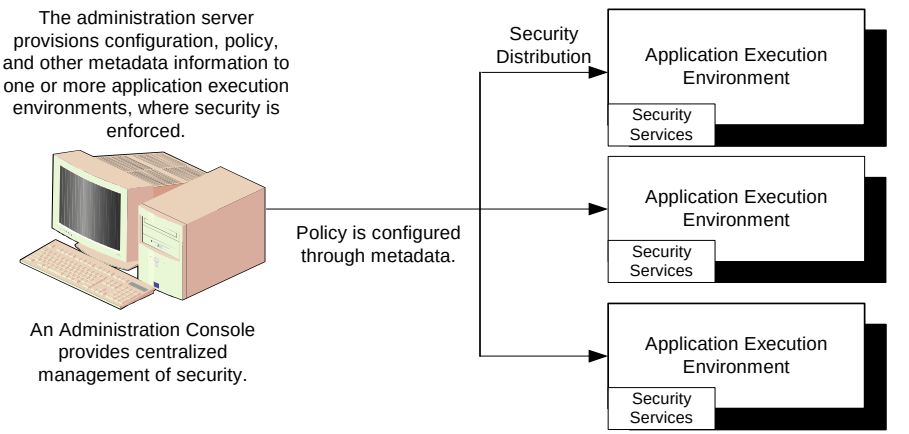
BEA AquaLogic Enterprise Security (ALES) is a fine-grained entitlements engine that allows the user to centrally define and manage policy to control access for both application software components (for example, URLs, EJBs, EJB methods, and so on.) as well as the application business objects (for example, accounts and patients records) that make up the application. Policy is evaluated and enforced locally in the application container so application context can be included as part of the access decision.

A major benefit of ALES as an entitlements engine is that it allows you to remove security logic from the application. Thus, you can take security decisions out of the hands of your developers and define access policy consistently across multiple applications. ALES is also a security

infrastructure product that provides a platform for creating a security service layer that can be used by multiple applications. It provides a standard set of security services including authentication, authorization, role mapping, auditing, and credential mapping. The product is distributed in that access decisions are made and enforced at runtime in the application container through a set of Security Service Modules (SSMs).

ALES consists of two major components—the Administrative Server and a set of SSMs. The Administrative Server can be accessed through a browser-based console that you can use to define your users, groups, and roles, the resources in your application, the policies that control access, and the security configurations of the SSMs (see Figure 2-1). ALES also includes Java and Web Services Interfaces that provide administration APIs. You can use these APIs to incorporate ALES administrative functionality into your own applications or even to write your own administrative console. The Administration Server supports delegated administrate in that allows you to delegate administrative functions to other users. Further, it provides a policy analysis function that enables you to analyze your policies to see which policies apply to specific resources, users, groups, and roles. To assist you in defining the access control policies, the Administration Server provides a resource discovery feature that you can use to automatically generate a list of application resources and to create a set of default access policies. Finally, to assist you in transporting your access policies from one Administration Server to another, the Administration Server provides policy exporting and importing tools.

Figure 2-1 Typical Application Execution Environment



Note: The BLM API supports all of the functionality supported by the Administration Console.

When a security configuration and/or policy is changed, you must distribute it so as to take effect throughout the enterprise, across multiple application execution environments. An open standards-based design allows customers, integrators and vendors to develop and incorporate their own custom security services. And, common security functions can be leveraged by applications throughout the enterprise.

The SSMs plug into the application container and intercept requests, enforce access policies, and make access control decisions in real time. SSMs get updated policy information through a real-time policy distribution mechanism that keeps policy sets up to date and consistent between SSMs. The security framework, the heart of the SSM, provides a set of security services including authentication, authorization, role mapping, auditing and credential mapping. ALES provides SSMs for the BEA WebLogic Platform products and for popular Web servers, such as Microsoft Internet Information Services (IIS) and Apache. Java and Web Services Interfaces are also provided that are language and infrastructure neutral. Applications can invoke the ALES security services directly through either the Java or Web Services APIs. In summary, the ALES SSMs provide the runtime enforcement of policy and a powerful security services layer and can be used as an integration platform for multiple security products in your infrastructure.

Key Features

The key features of the ALES product include:

- ***Support for securing the enterprise***, including Security Service Modules for: BEA WebLogic Platform 8.1, Internet Information Services (IIS), Apache, web services, and Java applications.
- ***Simplified infrastructure***, that improves information technology efficiency because the application security infrastructure can be leveraged by applications across the enterprise. This feature effectively reduces integration costs and provides stronger investment protection in third-party security technologies through a standards-based architecture.
- ***Centralized policy***, leads to lower administration costs. Security administrators can define and deploy access policies without the need for re-coding or re-deploying applications. Policies are all managed through a central Administration Console. In addition, policy inquiry functions allow the administrator to validate access control policy implementations prior to deployment.
- ***Delegation of administrative and user privileges***, allows privileges granted to one user to be assigned to another user under certain conditions, or constraints.
- ***BLM API for policy definition***, provides programmatic access to the ALES policy management infrastructure. This is a Java API that uses SOAP to communicate with the

central ALES management services. The API supports all of the functionality of the Administration Console including the creation and management of users, groups, roles, resources, resource authorization policies, and policy distribution and the security configuration and distribution. The BLM API examples (BEA_HOME\ales30-admin\examples\ policymgtwsapi) demonstrate how to use the BLM API for managing ALES roles and Policies.
For more information on the BLM API, see the BLM API Javadocs.

- ***Java API***, that allows security developers to develop environment interfaces or even integrate an application security infrastructure into an application. These Java interfaces support the most commonly required security functions and are organized into services that are logically grouped by functionality.
- ***Web Services API***, that allows security developers to write custom applications that invoke ALES 3.0 services through SOAP. These WSDL interfaces support the most commonly required security functions and are organized into services that are logically grouped by functionality.
- ***Role-based access control***, with integrated authentication, authorization and auditing. Administrators can design role-based access control (RBAC) policies to model their own business requirements, and then implement, test, and distribute them through a centralized Administration Server. You can include both resource and user attributes in your access control policies, along with other logical functions.
- ***A comprehensive and standards-based design***, to leverage Lightweight Directory Access Protocol (LDAP), Security Assertion Markup Language (SAML), Simple and Protected Negotiation mechanism (SPNEGO), eXtensible Access Control Markup Language (XACML), and Java. Additional support for standard security technologies, includes J2EE security technologies such as the Java Authentication and Authorization Service (JAAS), Java Secure Sockets Extensions (JSSE), and Java Cryptography Extensions (JCE).
- ***Maximum performance, scalability, reliability***, delivers a seamless, transparent experience for the user, while maintaining the necessary levels of application security through a distributed architecture.
- ***Support for customizing security services***, with step-by-step instructions, examples, and an open-standards model to help further reduce the cost of development. Security Service Provider Interfaces (SSPI) provide for rapid and simplified development of custom security services.

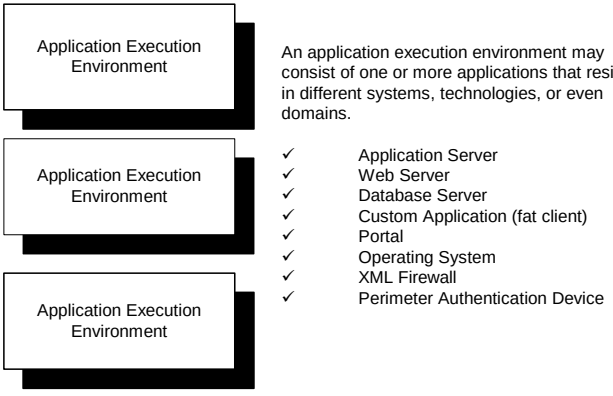
What is the Problem?

With the rush to build web-based applications and market services over the Internet, many developers had little comprehension of the security issues they may soon confront. Managing security is a huge challenge for any information technology organization that is providing new and expanded services to its employees, customers, and partners through both web-based and legacy applications. The advent of the Internet made protecting information and applications increasingly difficult to manage, monitor, and maintain. Financial transactions (ATM machines, bank transfers, credit card purchases and payments, stock market transactions), personal medical information (implementation of new Health Insurance Portability and Accountability Act or HIPPA regulations), Federal government facilities (Homeland Security affecting both military and civilian) provide only a few examples of areas where the concern for security has become essential and sometimes mandated by law.

Most applications require some form of security. As the complexity and volume of users and resources increases and with the rapid changes in business requirements that continue to evolve, the need for more stringent and robust security technologies becomes evident. To serve a worldwide network of users, an information technology organization must address the fundamental issues of maintaining the confidentiality, integrity and availability of the system and its data, providing the right information, to the right person, at the right time, across a diverse enterprise.

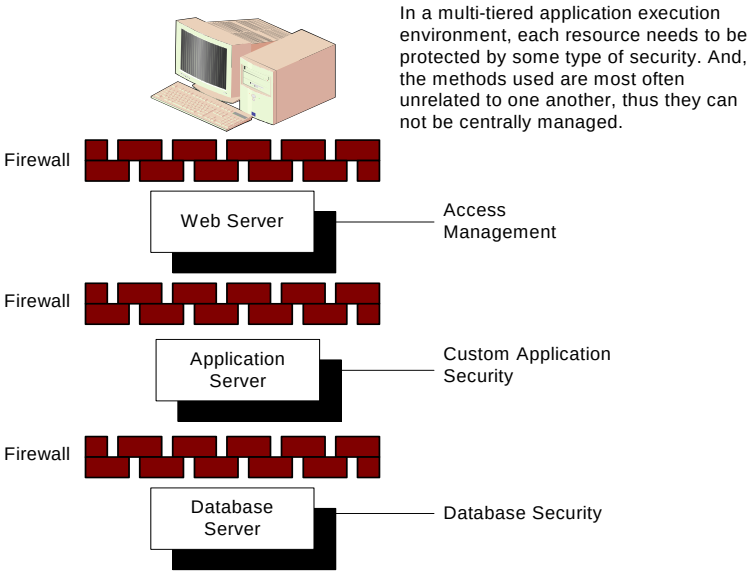
Because these applications often comprise a number of different components that may or may not reside on the same server or even in the same domain, policy management becomes extremely difficult and ensuring enterprise or regulatory compliance can prove impossible.

Figure 2-2 Application Execution Environment



A typical application execution environment is multi-tiered as shown in Figure 2-3 and may be distributed (vertically or horizontally) between multiple machines running on different operating system platforms. In this case, you must protect each tier or application component. The type of access control and technology for each one may be different and you need to be able to enforce security at each layer.

Figure 2-3 Multi-tiered Application Execution Environment



To address the multitude of potential breaches of security associated with multi-tiered environments, companies have had to purchase and integrate a variety of different and custom security technologies from a host of different vendors:

- Perimeter-based authentication for firewalls, with single sign-on, credential mapping, SAML, Virtual Private Networks (VPN), and web servers
- Digital certificates for server authentication
- Secure Sockets Layer (SSL) data encryption software or Hyper-Text Transfer Protocol (HTTPS) for secure transfer of data
- Access control and entitlement software

Integration of security technologies requires the application developer to embed these technologies and hard-code both integrated and unified security requirements within each application. Thus, as the number of applications increases, the expenses associated with application development and maintenance also increases. As a best practice, the application developer should *not* be responsible for developing, implementing, and managing security requirements.

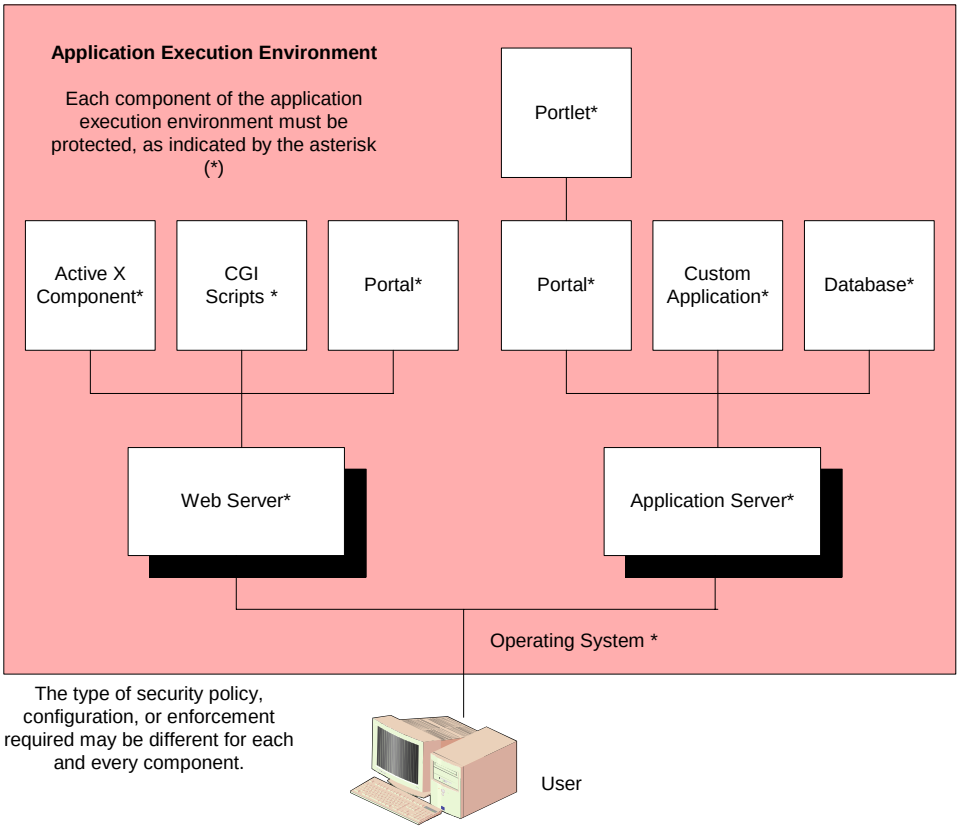
Early authorization implementations used static and inflexible approaches to define the different types of access granted or denied for a user. Because this type of implementation is extremely time-consuming (if only due to the number of users and the different types of user storage methods in use), it has become impractical for many implementations. Further, the cost of maintaining static first-generation security services can be exorbitant.

What is the Solution?

BEA has developed an application security infrastructure (ASI) that can be external to and isolated from the application itself. Using a services-oriented, policy-based architecture, you can replace the integrated security silo technologies and hard-coded policies. Figure 2-4 illustrates how a basic application execution environment can be protected using an integrated approach. Each component in the application requires protection, although the type of security typically varies.

A typical information technology environment consists of various types of servers–HTML, proxy, BEA WebLogic, Legacy, J2EE, and application–that access numerous LDAP and database servers containing information such as your user community (name, address, etc.). While the WebLogic platform servers provide application-level security for J2EE components, J2EE-based web services, portal and portlets (EJB, JSP/Servlet, JDNI, JDBC, JMS, MBeans), ALES provides application security for additional platforms and web servers.

Figure 2-4 Integrated Application Security

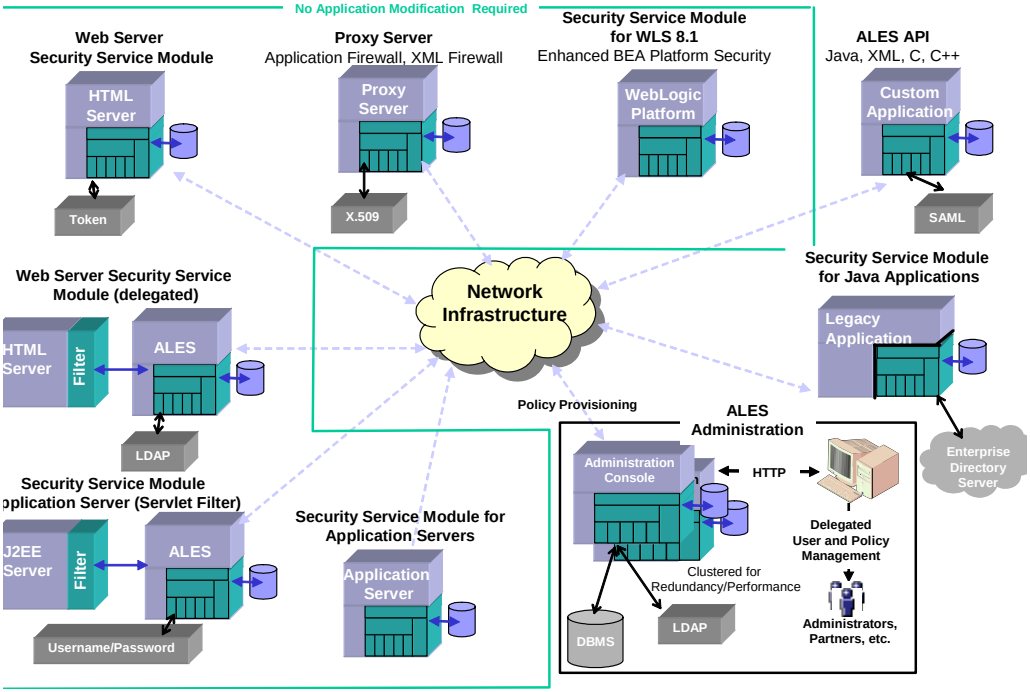


The open flexible architecture of the ALES products provides advantages to all levels of users and introduces an advanced design for securing your applications. With distributed computing, applications must be integrated across the network, as shown in Figure 2-5. ALES provides a distributed enterprise security solution that, together with clear and well-documented policies and procedures, can insure the confidentiality, integrity and availability of its applications and data.

Distributed Computing Security Infrastructure

Applications across the enterprise are built on a heterogeneous infrastructure with diverse resources. With an application security infrastructure as shown in Figure 2-5, the ALES products support a fully distributed architecture; integrating all applications across the network.

Figure 2-5 Distributed Computing Security Infrastructure Vision



The ALES products provide a variety of services that use the AquaLogic security framework, including enhanced policy-based authorization with role mapping, authentication with support for single-sign on and credential mapping, and customizable auditing features. A services-oriented strategy to application security infrastructure improves efficiency and strengthens security by providing a unified and consistent approach across the enterprise. BEA delivers security services that allow third-party security technologies to be exposed as reusable services, to further reduce integration time and costs, promote choice, and insure investment protection.

The type of security services you implement depends on the type of the application component you want to protect. You can use the set of security providers delivered with each Security Service Module to configure and enforce each security service or you can develop custom security providers.

The security services seek to provide ease of use, manageability for end users and administrators, and customizability for application developers and security developers. Administrators who configure and deploy applications can use the security providers included with the product that support most standard security functions or can create custom security providers. The product environments supported include WebLogic Server Versions 8.1 and 9.x/10.0, Internet Information Services (IIS) and Apache Web Servers, web services, and Java applications. ALES will expand this family of Security Service Modules in subsequent releases.

- WebLogic Server 9.x/10.0 Security Service Module

  The WebLogic Server 9.x/10.0 Security Service Module integrates ALES 3.0 with BEA WebLogic Server versions 9.1, 9.2, and 10.0.

- WebLogic Server 8.1 Security Service Module

  Supports BEA WebLogic Server, Version 8.1 and enhances the existing security services in the application server, providing customizable auditing, multi-domain standards-based single sign-on, database and Microsoft Window NT authentication, database credential mapping, and expanded policy expression capabilities for authorization and role assignments.

- IIS Web Server Security Service Module

  Supports the IIS Web Server. After installation, the security service module (SSM) binds with the web server through the web server application programming interface (ISAPI) so that the SSM can be used to protect web server application resources.

- Apache Web Server Security Service Module

  Supports the Apache Web Server. After installation, the SSM binds the web server through the web server filter so that the SSM can be used to protect web server application resources.

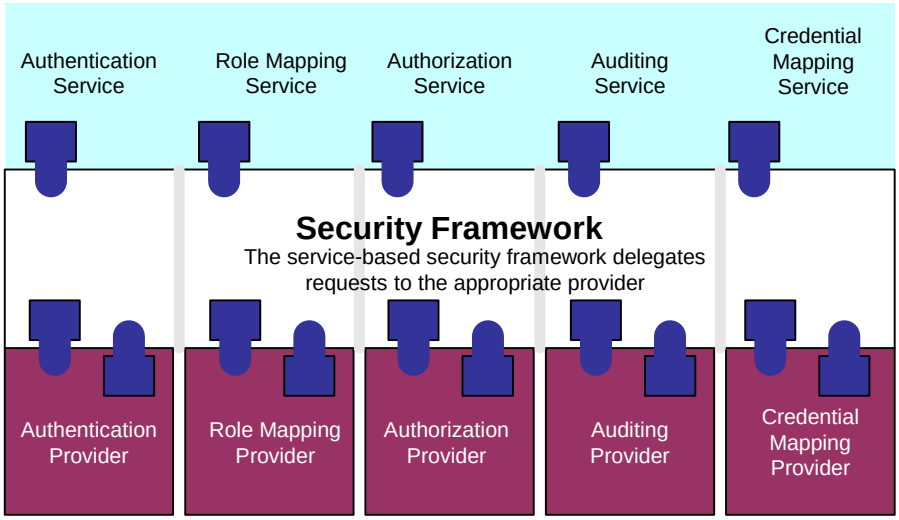
- Web Services Security Service Module

  An application programming interface (API) that allows security developers to write custom applications that invoke ALES 3.0 services through SOAP. These interfaces support the most commonly required security functions and are organized into services that are logically grouped by functionality.

- Java Security Service Module

  An application programming interface (API) that allows security developers to develop environment interfaces or even integrate an application security infrastructure into an application. These interfaces support the most commonly required security functions and are organized into services that are logically grouped by functionality.

Figure 2-6 Security Framework



Each Security Service Module is delivered with a full set of security providers. Table 2-1 lists the types of providers that are available for configuration.

Table 2-1 BEA AquaLogic Enterprise Security Service Providers

Provider	Description
Authentication Provider	Supports open-standard support for SAML, SPNEGO, and X.509 identity assertion, and authentication support for Microsoft Windows NT, Microsoft Active Directory, Netscape LDAP, Novell LDAP, relational database, and OpenLDAP login modules.
Credential Mapping Provider	Maps credentials used by a legacy or any remote system. The application then uses the appropriate credentials to log in to a remote system on behalf of a subject that already authenticated to support single sign on.

Table 2-1 BEA AquaLogic Enterprise Security Service Providers (Continued)

Provider	Description
Authorization Provider	Controls access to resources based on the role and policy assigned to the requested resource. An access decision is the part of the authorization provider that actually determines whether a user has permission to perform an operation on a resource. Authorization providers secure access to resources and transactions, enabling an organization with increasingly complex user communities to provide secure finely-grained access to resources. Access decisions, provided through a role-based authorization provider, incorporate relevant environmental, contextual, and transaction-specific information, allowing security policies to support business processes throughout the organization. In addition, an adjudication provider resolves authorization conflicts when you configure multiple authorization providers.
Role Mapping Provider	Supports dynamic role associations by obtaining a computed set of roles granted to a requestor for a resource.
Auditing Provider	Provides an electronic trail of all transaction activity and can include changes to system configuration parameters, policy changes, and transactions. For each audit item, the information can include who, what, when, where, and sometimes why.

How Our Solution Benefits You

The modular ALES service architecture provides specific benefits for:

- Application Developers
- Server and Application Administrators
- Security Developers
- Security Architects

Application Developers

Because most security for web applications and EJBs can be implemented by a system administrator, application developers do not need to be concerned about the details of securing the application, unless there are special considerations that must be addressed explicitly in the code. Security developers can also take advantage of BEA-supplied Application Programming Interfaces (APIs). Three sets of APIs are provided:

- The Java APIs are found in the `com.bea.security` package as described in Javadocs for Java APIs.
- The Web Service public interface APIs are found in the `SSM-SOAP Wsdldocs` package as described in Web Services API Wslddocs.
- The Security Provider SSPIs are found in the `weblogic.security` package as described in Javadocs for WebLogic Security Providers.

Server and Application Administrators

Administrators can use the security providers supplied as part of the product to implement an integrated solution. Administrators can use the Administration Server to define security roles and assign security policies to resources to create an authorization scheme that suites the needs of their business. In addition, the administrator can modify, test, and deploy the policy quickly and efficiently.

Security Developers

Third-party providers are integrating their products by using the Security Service Provider Interfaces (SSPI). As the underlying integration mechanism for security providers, the SSPI allows development of custom security providers. The SSPIs are available for Adjudication, Auditing, Authentication, Authorization, Credential Mapping, Identity Assertion, and Role Mapping. For information on the SSPIs, see Javadocs for Security Service Provider Interfaces.

This architecture allows security developers to provide integrated solutions that are easy to use. The result is a reduction in development requirements, which means an increased return on investment when implementing an enterprise security management solution. And, custom security services developed for WebLogic Platform 8.1 are compatible with the ALES services.

Security Architects

A dynamic role-based policy architecture eliminates the need for application developers to design and implement business policy and embed it within each and every instance of an application. More efficient policy administration enables an organization to adapt quickly to dynamic business processes as security policies are designed, tested, deployed, and distributed quickly by security administrators with no coding required.

Delegated administration allows for centralized control and delegated labor, enabling administrators more familiar with the needs of a particular user constituency to implement business policy.

It also allows the implementation of policies across a much larger, more complex, user community with standard policy (for example, consisting of employees, business partners, customers). If a change to a policy is required, it can be distributed throughout the enterprise and take effect whenever desired. With ALES products, if your application is already written to use some form of authentication or authorization schema, and the schema changes, no changes are required within the application.

Standards

ALES products adhere to the following standards.

Table 2-2 BEA AquaLogic Enterprise Security Standards

XML Standard	Used to
SAML	Participate in SAML-based single sign-on (SSO) environment.
WSDL 1.1	The Web Services Description Language (WSDL) is an XML-based specification that describes a web service. A WSDL document describes web service operations, input and output parameters, and how a client application connects to the web service.
Java Standards	**Used to**
CertPath	Retrieve X.509 digital certificates associated with infrastructure protection; available for customer direct use.
KeyStore	Retrieve RSA private keys associated with X.509 digital certificates associated with infrastructure protection; available for customer direct use.
JSSE	Protect infrastructure network connections for establishment of mutual trust.

Table 2-2 BEA AquaLogic Enterprise Security Standards (Continued)

XML Standard	Used to
JCE	Integrate cryptographic libraries.
JAAS	Provide authentication service implementations.
Miscellaneous Standards	**Used to**
XACML 2.0	XACML, the eXtensible Access Control Markup Language, is an OASIS standard. XACML is a standard language for expressing access control, or authorization, policy, and a standard format for expressing queries over these policies.
X.509	Validate the identity of infrastructure components through digital certificates; supported as proof of identity for customer use.
LDAP v3	Retrieve configuration information from the Service Control Manager, and user identity and user attributes from an LDAP v3 directory server.
ISAPI	Support compliant runtimes for authentication, SAML single sign-on, and protection of hosted web pages.
FIPS 140	Support certification of the embedded cryptographic libraries used for cryptographic protection and TLS protocol.
TLS v1 and SSL	Protect network communication between infrastructure components.
JDBC	Provide access to database stores using the database provider.

Examples

This release of ALES 3.0 includes the Administration Console and SSM examples shown inTable 2-1.

Table 2-3 AquaLogic Enterprise Security Examples

Example Location	Description
Admin Examples	
BEA_HOME\ales30-admin\DBSetupKit\README	Provides an example of installing and configuring a policy database for ALES. It also provides an example of installing an Oracle database.
BEA_HOME\ales30-admin\policy\aldsp_sample_policy\README.txt	Provides an example of importing the AquaLogic Data Services Platform sample policy.
BEA_HOME\ales30-admin\policy\alsb_sample_policy\README.txt	Provides an example of importing the AquaLogic Service Bus sample policy.
BEA_HOME\ales30-admin\policy\portal_sample_policy\README.txt	Provides an example of importing the WebLogic Portal sample policy.
BEA_HOME\ales30-admin\policymgtapi\readme.txt	Demonstrates how to use the ALES BLM JAVA API for managing ALES.
BEA_HOME\ales30-admin\policymgtwsapi\readme.txt	Demonstrates how to use the ALES BLM Web Service API for managing ALES roles and policies. This example demos how to establish a session with a remote BLM service, and how to do some simple operations using the BLM Web Service API.
BEA_HOME\ales30-admin\examples\EJBAppExample	Demonstrates the principles of an EJB application functioning under the ALES framework. The example shows how the additional layer provided by ALES can make an EJB application more secure by defining custom dynamic authorization policies without any changes in the application or the deployment descriptors.
Java SSM Examples	

Table 2-3 AquaLogic Enterprise Security Examples

Example Location	Description
BEA_HOME\ales30-ssm\java-ssm\examples\AttributeRetriever\README	This example shows how to implement a custom Attribute Retriever.
BEA_HOME\ales30-ssm\java-ssm\examples\JavaAPIExample\README	This example shows how to use Java SSM APIs to retrieve basic security services, and use them to do authentication, authorization, and auditing.
BEA_HOME\ales30-ssm\java-ssm\examples\QueryResources\README	This example shows how to get a list of the resources granted to a user. It also retrieves the list of resources denied to a user.
Web Services SSM Examples	
BEA_HOME\ales30-ssm\webservice-ssm\examples\JavaWebServiceClient\README	This example shows how to connect to a Web Service SSM, retrieve basic security services, and use them to do authentication, authorization, and auditing.
BEA_HOME\ales30-ssm\webservice-ssm\examples\SsmNet\README	This sample demonstrates how to access ALES Web Service SSM through its published WSDL in the .NET environment. This sample contains two parts: 1. An abstraction layer that encapsulates the stub code generated from the WSDL. 2. A simple ASP.NET web application that covers all the interfaces exposed by the Web Service SSM.
BEA_HOME\ales30-ssm\webservice-ssm\examples\SsmWorkshop\readme.txt	This example is a simple web application based on page flow, which uses a service control generated from ALES SSM web service WSDL file by Workshop to protect it's content.
BEA_HOME\ales30-ssm\webservice-ssm\examples\XACMLClient\README	XACML client is a simple client that does authorization to the Web Service SSM using the XACML protocol. It needs to first authenticate with the Web Service SSM and retrieve an identity assertion, then authorize with the identity assertion as subject.
WLS SSM Examples	

Table 2-3 AquaLogic Enterprise Security Examples

Example Location	Description
BEA_HOME\ales30-ssm\wls-ssm\examples\ALESEnabledWLP92Domain\README	This example shows how to protect a WebLogic Portal application using WLS 9.x/10.0 SSM.
WLS 8 SSM Examples	
BEA_HOME\ales30-ssm\wls8-ssm\examples\ALESEnabledWLPDomain\README	This example shows how to protect a Portal application using WLS SSM.
BEA_HOME\ales30-ssm\wls8-ssm\examples\ALESEnabledWLSCluster\README	ALES Enabled WLS Cluster Example.
BEA_HOME\ales30-ssm\wls8-ssm\examples\ResourceConverter	Resource Converter is a Java-based plug-in that acts as a provider extension to extend the capabilities of the ASI Authorization provider. The example implements a custom Resource Converter.
BEA_HOME\ales30-ssm\wls8-ssm\examples\SAMLServletExample	The SAML Servlet example demonstrates how an application can use the SAML servlet to use SAML Assertions to allow servers in different domains to operate in a federation of trusted servers based on successful Single Sign On (SSO) login to one of the servers by a user.
BEA_HOME\ales30-ssm\wls8-ssm\examples\taglib	This example shows how to use ALES with taglibs. This example includes a set of sample taglibs and a simple web application that uses those tags in its index.jsp.

CHAPTER 3

Architecture Overview

This section describes the general architecture of the ALES services, providers and service modules. Each Security Service Module comes with a set of security providers. Although applications can leverage the services offered through the existing security providers, the flexible infrastructure also allows security vendors, integrators, and customers to write their own security providers. The ALES providers and third-party security providers can be mixed and matched to create unique security solutions, allowing organizations to take advantage of new technology advances in some areas while retaining proven methods in others. The Administration Server allows you to configure and manage all your security providers and service modules through one unified management console.

The architecture comprises the following major components, which are discussed in the following sections:

Administration Server

The Administration Server gives you the enterprise-wide visibility you need to analyze security policies and ensure that applications and resources are properly protected. ALES also lets you delegate security administration to remote administrators who often better understand local users and business needs and who are better positioned to manage the security policies. By combining centralized control with delegated administration, you can define and manage overall policies

while specifying the management responsibilities to be handled by organizational administrators. For additional information on the Administration Server features, see Chapter 5, "Security Administration." The Administration Server consists of several components (as shown in Figure 3-1), including:

Policy Distributor—Ensures that the correct policies are provided to each Security Service Module and maintains policy synchronization.

Policy Database—Maintains policy data managed by the Administration Server in a relational database. The database management system provides the authoritative source of configuration and policy. Data from the policy database is distributed to the Security Service Modules by the Policy Distributor.

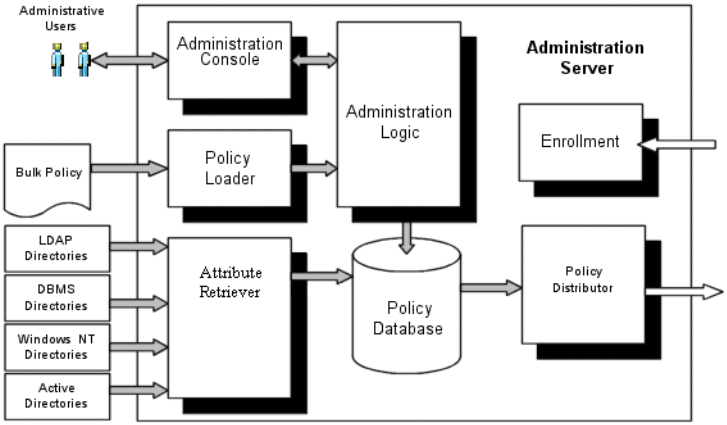
Policy Exporter—Exports policy data from the policy database for later use with the same Administration Server or another instance of the Administration Server.

Policy Loader—Imports policy data from an external file, generated in another system, exported from another instance of an Administration Server, or manually coded.

Administration Console and **Entitlements Management Tool—**Supports administrative policy security and administration delegation through browser-based user interfaces. Security configuration, policy configuration, user attributes (if required), resources, and rules are all managed through these console.

Administration Logic—Maintains the Policy Database used by both the administration consoles and the Policy Loader.

Figure 3-1 Administration Server Architecture



Service Control Manager

ALES employs a fully-distributed security enforcement architecture consisting of Security Service Modules embedded in the applications, application servers, and web servers being secured (see Figure 3-2). To facilitate the management of a potentially large number of distributed Security Service Modules, the Administration Server uses a remote administration mechanism to distribute appropriate configuration and policy data to each Security Service Module.

The Service Control Module (SCM) is an essential component of this remote administration mechanism. Each Service Control Module is responsible for storing and maintaining the configuration data for all Security Service Modules running its machine. Once started, a Security Service Module receives its configuration data from the local Service Control Module. When a change is made and distributed from the Administration Server, the Service Control Manager receives the change and updates the cached copy of the configuration. On restart, the Security Service Module receives updated configuration data from the Service Control Manager. Policy data does not require a restart, but is applied based on the desired provisioning characteristics.

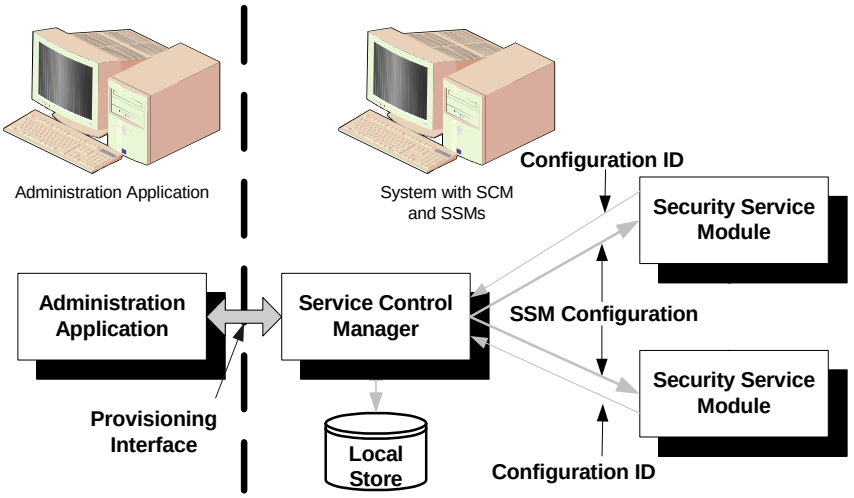
Note: It is possible to deploy an SSM without the SCM. You can use the PolicyIX tool, described in PolicyIX in the Administration Reference, to communicate directly with the BLM and retrieve configuration data. The PolicyIX tool allows you to export configuration data (configured either through the ALES Administration Console or directly via the BLM API) for a given SSM to an XML file, and use it with the configured

SSMs when the SCM is not available. See the SSM Installation and Configuration Guide for additional information.

The SCM is always installed on the ALES Administration server.

In addition to facilitating management, the Service Control Manager enables Security Service Modules to operate in the absence of the Administration Server. Because the Service Control Manager maintains a persistent copy of each configuration, new Security Service Modules can be started and existing Security Service Modules continue to function, even if the Administration Server goes down or is intentionally unavailable, such as in occasionally connected computing environments.

Figure 3-2 Service Control Manager



Security Service Modules

ALES supports a variety of Security Service Modules that you integrate with the security framework and provision as needed. The primary function of the security framework is to provide a simple application programming interface (API) that can be used by security and application developers to define security services. For a complete discussion of ALES services, see Chapter 4, "Security Services." You may incorporate as many Security Service Modules as you need to secure the enterprise, and configure and manage them directly through a central Administration Server as shown in Figure 3-2. The distributed nature of the architecture allows you to configure, manage and distribute policy throughout the enterprise.

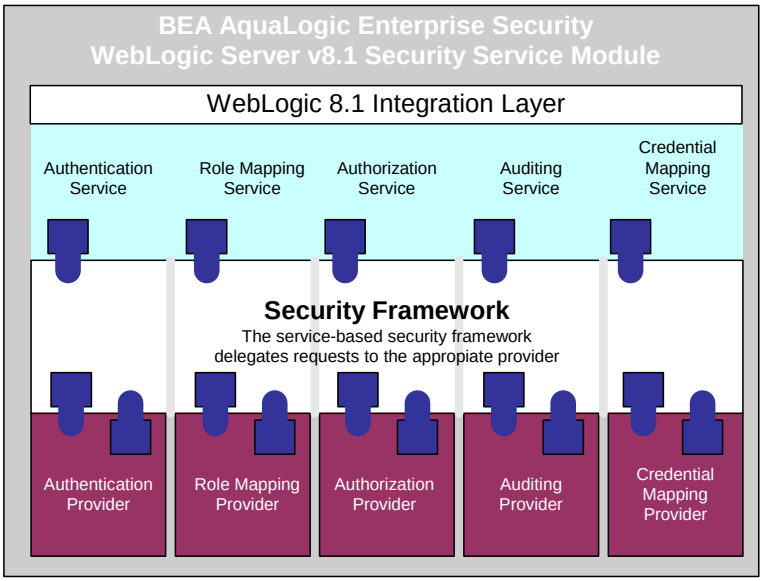
Configuration data for each Security Service Module is maintained within each machine and handled by a Service Control Manager. One additional benefit of this architecture is that even if the administration server goes down (either for maintenance or due to failure), there is no impact on the applications or security services provided by those Security Service Modules. At this time, the following Security Service Modules are available:

- "WebLogic Server 8.1 Security Service Module" on page 3-5
- "WebLogic Server 9.x/10.0 Security Service Module" on page 3-6
- "Web Server Security Service Module" on page 3-7
- "Web Services Security Service Module" on page 3-15
- "Java Security Service Module" on page 3-18

WebLogic Server 8.1 Security Service Module

The WebLogic Server 8.1 Security Service Module is a security enhancement product that supports BEA WebLogic Server, Version 8.1. Further, the Security Service Module ties the application server into the Administration Server so that all application server administrative security activities are performed through the Administration Server. The application server with the Security Service Module add-on supports enterprise-level security by making security for WebLogic Server host applications an integral part of the enterprise policy. All WebLogic Server security-related functions remain available, but those functions are provided through the Security Service Module. Figure 3-3 shows the major components of the WebLogic Server 8.1 Security Service Module.

Figure 3-3 WebLogic Server 8.1 Security Service Module Architecture



WebLogic Server 9.x/10.0 Security Service Module

The WebLogic Server 9.x/10.0 Security Service Module is a security enhancement product that supports BEA WebLogic Server, Version 9.x/10.0.

This SSM uses a different security framework from the one used in the WebLogic Server 8.1 SSM and the other AquaLogic Enterprise Security SSMs. When you install the WebLogic Server 9.x SSM, AquaLogic Enterprise Security uses the WebLogic Server 9.x security framework. As a consequence, when you use the WebLogic Server 9.x SSM, you configure security providers and other aspects of the SSM in the WebLogic Administration Console, rather than the AquaLogic Enterprise Security Administration Console.

You still use the AquaLogic Enterprise Security Administration Console to write security policies for all SSMs, and to configure SSMs other than the WLS 9.x SSM. You must also use the ALES Administration Console to configure the ASI Authorizer and ASI Role Mapper providers.

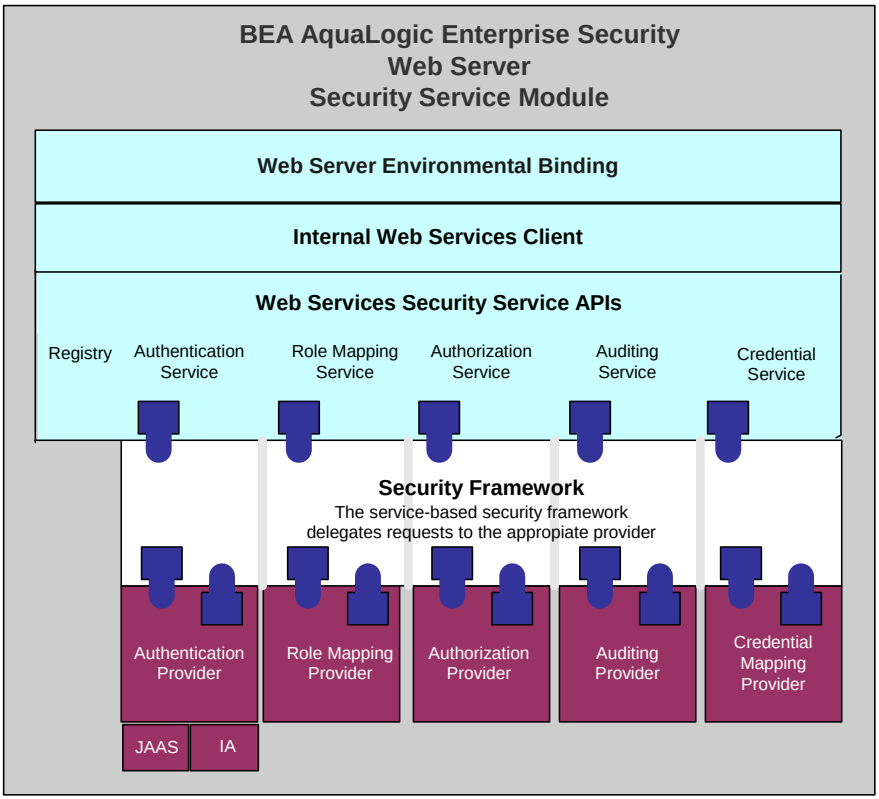
Web Server Security Service Module

The Web Server Security Service Module (SSM) provides an environmental binding between the ALES infrastructure and IIS and Apache web servers. The SSM consists of three components, a Web Server Environmental Binding, an internal Web Services Client, and the Web Services SSM (which includes the Security Service APIs, Security Framework and security providers) (See Figure 3-4). The ALES infrastructure provides six distinct services: Registry, Authentication, Authorization, Auditing, Role Mapping, and Credential Mapping. Each of these services is expressed in a way that is understandable to applications running within a web server that is protected by the ALES infrastructure. Therefore, the SSM can be used to configured and enforce security for web server applications and resources.

The Web Server SSM makes access control 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.

You can tailor the Web Server SSM to meet your 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. Additionally, the Web Server SSM enables security administrators and web developers to perform security tasks for applications running on a web server.

Figure 3-4 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.

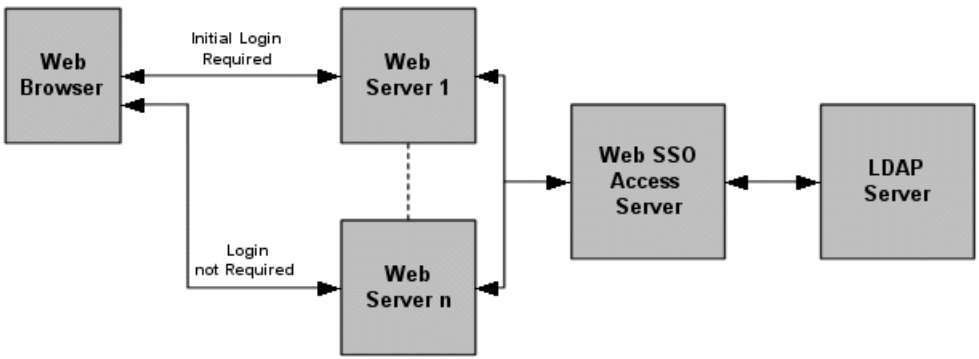
Web Single Sign-on Capabilities

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 3-5 shows the basic components of a web single sign-on service.

Figure 3-5 Web Single Sign-on



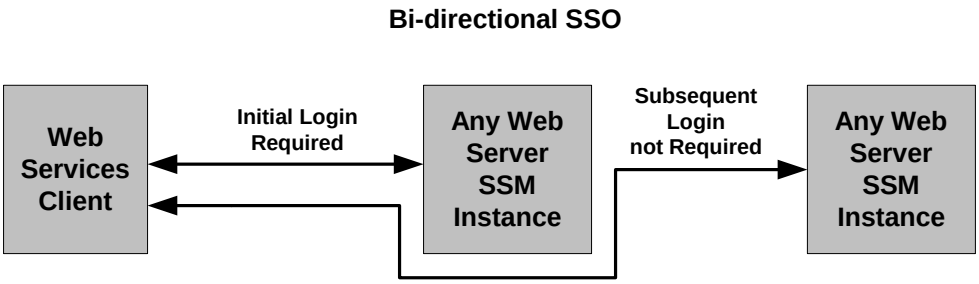
While web single sign-on facilitates access and ease of use, it does not improve security. In fact, security requirements should be considered when implementing a web single sign-on solution.

Single Sign-On Use Cases

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

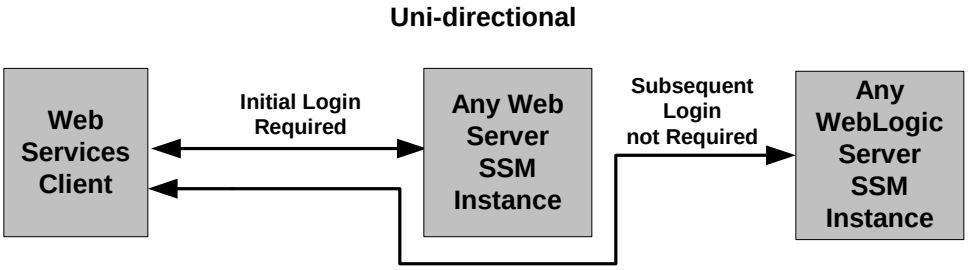
- **Bi-directional SSO among Web Server SSMs**—Once users are authenticated by a Web Server SSM, they are given an identity assertion token that they can use to access other Web Server SSM instances without being required to authenticate again (see Figure 3-6).

Figure 3-6 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 3-7).

Figure 3-7 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.

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 Services Security Service Module

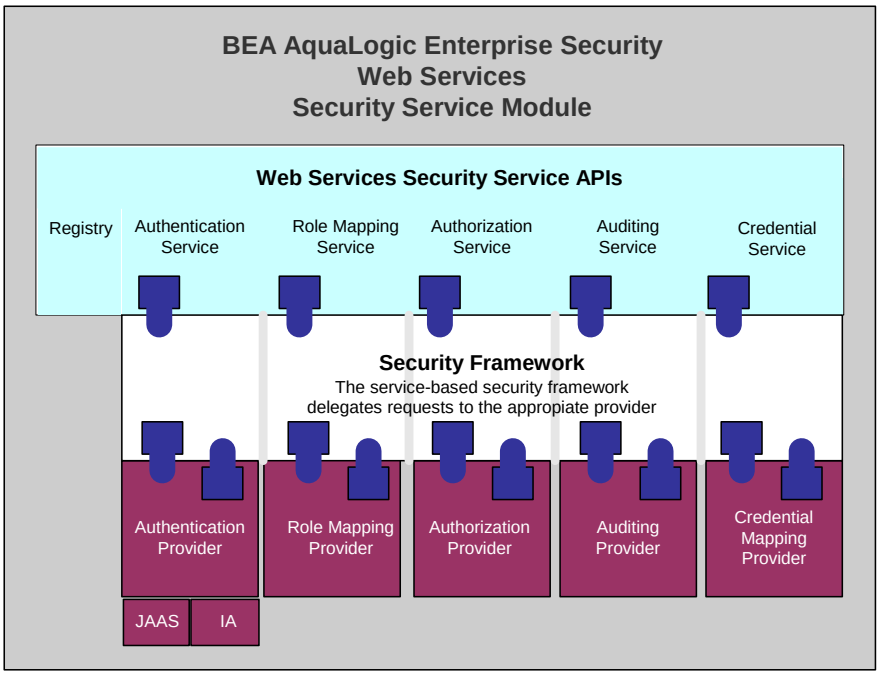
The Web Services SSM provides an application programming interface (API) that allows security developers to write custom application clients that invoke AquaLogic Enterprise Security services through SOAP. These interfaces support the most commonly required security functions and are organized into services that are logically grouped by functionality.

A Web Service 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 Security Service Module (SSM) provides five security services: Authentication, Authorization, Auditing, Role Mapping, and Credential Mapping (see Figure 3-8). The WSDL provided for these services can be used to developed web services clients to access the ALES infrastructure and use it to make access control decisions for custom applications.

Figure 3-8 shows the components of the Web Services SSM.

Figure 3-8 Web Services SSM Components



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/ales30-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 /10.0 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 3-17
- "Authorization Service" on page 3-18
- "Auditing Service" on page 3-18
- "Role Mapping Service" on page 3-18
- "Credential Service" on page 3-18

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.

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 SOAP request or response 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 SOAP request or response 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.

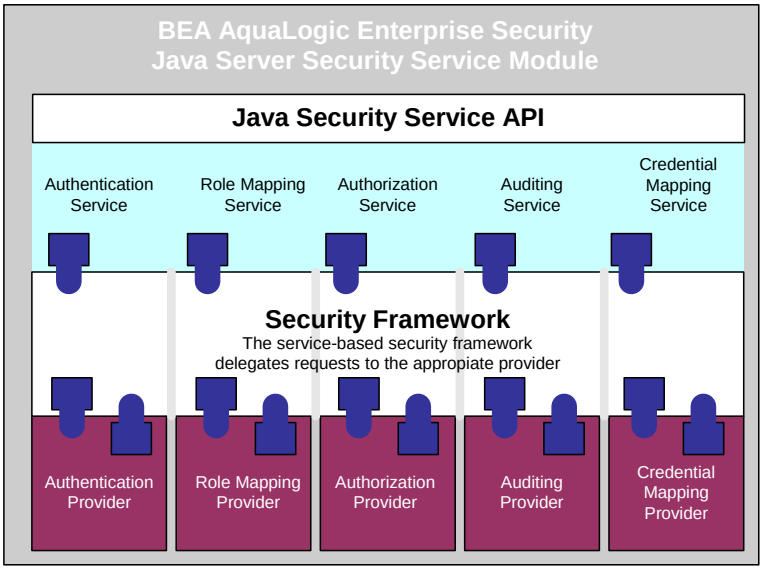
Java Security Service Module

The Java Security Service Module provides an application programming interface (API) that allows security developers to insert security into their applications. These interfaces support the most commonly required security functions and are organized into services that are logically grouped by functionality.

After you use the Java Security Service Module interfaces to implement security functions in your Java application, you can deploy and run your application on any instance of a Java Security

Service Module that supports the configuration requirements of your application. The Java Security Service Module offers five security services: Authentication Service, Authorization Service, Auditing Service, Role Service, and Credential Mapping Service. The name of each service indicates a type of function that can be implemented within a Java application. Each of these services is discussed in Chapter 4, "Security Services." Figure 3-9 shows the major components of the Java Security Service Module. The Java Security Service Module comprises the security service APIs, the security framework, and the security providers that you configure.

Figure 3-9 Java Security Service Module Architecture



CHAPTER 4

Security Services

The following sections describe the fundamentals of the ALES services:

Authentication and Identity

Authentication answers the question, *Who are you?* using credentials such as username and password combinations. An Authentication provider is used to prove the identity of users or system processes. The Authentication provider also stores, transports, and makes identity information available to various components of a system when needed.

Authentication is the process of verifying an identity claimed by or for a user or system process. An authentication process consists of two steps:

1. Identification—presenting an identifier to the security system.
2. Verification—presenting or generating authentication information that corroborates the binding between the entity and the identifier.

Authentication Service

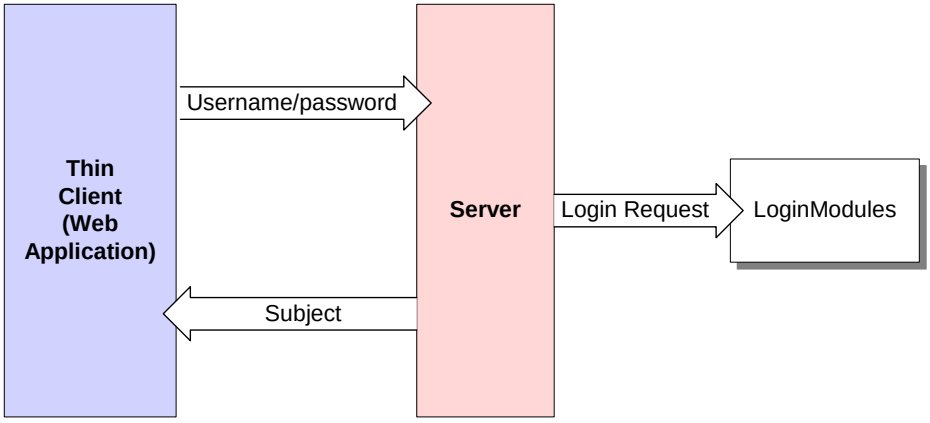
The Authentication providers support numerous methods of authentication services, including user username/password and identity assertion. In addition, a set of Authentication providers is supplied that access various types of user directories and identity tokens.

Figure 4-1 shows how the authentication process works for a web application that requires a username and password to perform the login. When a user attempts to log into a system using a username and password combination, the Authentication provider establishes trust by validating that username and password. This process also requires the use of a Login Module as described in "Identity Assertion" on page 4-5.

Note: Because the authentication service is implemented using the JAAS standard, developers of custom Authentication providers are involved with the JAAS process directly.

The authentication process requires a valid list of users and groups, through which their identity can be verified. ALES supports any number of LDAP servers or a database.

Figure 4-1 Authentication Service Example for Username and Password



Types of Authentication

Users must be authenticated whenever a request is made to access a protected resource. For this reason, each user is required to provide a credential (for example, a password). The following types of authentication are supported by the Authentication providers:

- "Username and Password Authentication" on page 4-3
- "Perimeter Authentication" on page 4-3

You can use one of the Authentication providers provided as part of ALES or you can create a custom authentication provider to perform a different type of authentication. Support for the following methods of user and password authentication include:

- Microsoft Windows NT
- OpenLDAP
- Novell
- Active Directory
- Database

Username and Password Authentication

In username and password authentication, a user ID and password are requested from the user and sent to the Security Framework. The Security Framework authenticates the user.

You can use Secure Sockets Layer (SSL) or Hyper-Text Transfer Protocol (HTTPS) to provide an additional level of security to username and password authentication. Because SSL encrypts the data transferred between the client and the Security Framework, the username and password are encrypted and not transferred in clear text. Therefore, ALES services can authenticate the user without compromising the confidentiality of the username and password.

Perimeter Authentication

Perimeter authentication is the process of authenticating the identity of a remote user outside of the application server domain. You can use an Identity Assertion provider to establish **perimeter authentication**—a special type of authentication that uses tokens. The Security Framework supports identity assertion providers that perform perimeter-based authentication and handle multiple security token types and protocols.

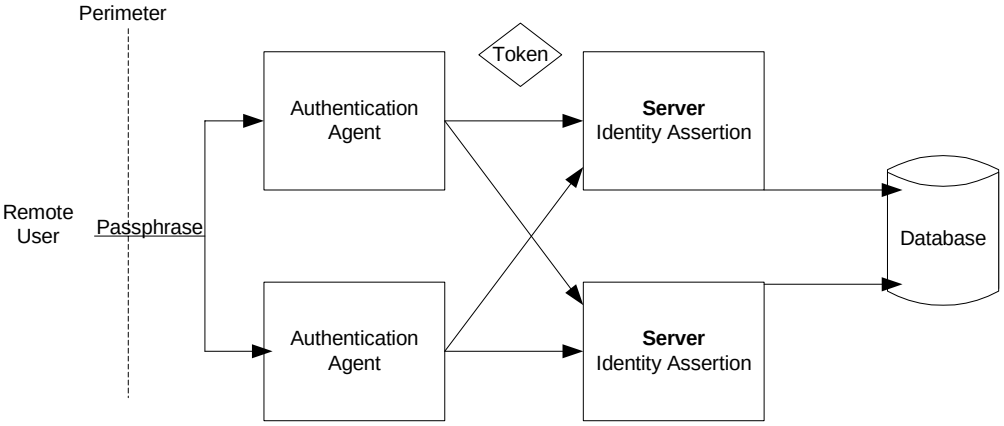
How is Perimeter Authentication Accomplished?

ALES services are designed to extend the single sign-on concept all the way to the perimeter by using identity assertion (see Figure 4-2). Provided as a critical piece of the Security Framework, the concept of identity assertion allows you to use the authentication mechanism provided by perimeter authentication schemes such as the Checkpoint OPSEC, the emerging Security Assertion Markup Language (SAML), Simple and Protected Negotiation Mechanism (SPNEGO), or enhancements to protocols such as Common Secure Interoperability (IIOP-CSIv2) to achieve this functionality.

Perimeter authentication is typically accomplished by the remote user specifying an asserted identity and some form of corresponding proof material, normally in the form of a passphrase, which is used to perform the verification. The **authentication agent**, the entity that actually vouches for the identity, can take many forms, including: Web Server, VPN, firewall, enterprise authentication service, or some other form of global identity service. Each of these forms of authentication has one common characteristic–they all perform an authentication process that results in an artifact or **token** presented to vouch for the authenticated user at a later time.

Currently, the format of the token varies from vendor to vendor, but there are efforts underway to define a standard token format. SAML is an emerging XML-based protocol for exchanging information securely. In addition, there is a current standard for identity attribute certificates that is based on the X.509 standard for digital certificates. But, if the applications and the infrastructure on which they are built are not designed to support this concept, enterprises are still forced to require remote users re-authenticate to the applications within the network. You can develop custom authentication providers that support different token types, including SAML.

Figure 4-2 Perimeter Authentication



Support for perimeter authentication requires the use of one or more identity assertion providers designed to support one or more token formats. You can implement multiple and different identity assertion providers. The tokens are transmitted as part of any normal business request, using the mechanism provided by each protocol supported. Once a request is received, the entity that handles the processing of the protocol message recognizes the existence of the token in the message. This information is used in a call to the Security Framework, which then calls the appropriate identity assertion provider to handle the verification of the token. It is the

responsibility of the identity assertion provider implementation to perform whatever actions are necessary to establish validity and trust in the token and to provide the identity of the user with a reasonable degree of assurance, without the need for the user to re-authenticate to the application.

The passphrase entered by the user is a string of characters (typically, much longer than a password) used to create a digitally-encrypted signature (or private or secret key). The passphrase adds an extra level of authentication security to ensure that you are who you say you are.

Identity Assertion

Identity Assertion providers support the mapping of a valid token to a user. You use an Identity Assertion provider to support the specific types of tokens that you use to assert the identities of users or system processes. You can develop an Identity Assertion provider to support multiple token types, but the administrator must configure the Identity Assertion provider so that it validates only one active token type. While you can have multiple Identity Assertion providers in a application domain with the ability to validate the same token type, only one Identity Assertion provider can actually perform this validation.

The security architecture supports Identity Assertion providers that perform perimeter-based authentication to Web Servers, firewalls, and VPNs. Identity Assertion providers support different token types, such as X.509 certificates (Kerberos), SAML and handles multiple security protocols (such as SOAP and IIOP-CSIv2). When used with an Authentication provider Login Module, Identity Assertion providers support single sign-on. For example, the Identity Assertion provider can generate a token from a digital certificate, and that token can be passed around the system so that users are not asked to sign on more than once. The Identity Assertion providers in ALES support the following token types:

- `Authenticated User` type
- CSIv2 principal name identity
- CSIv2 anonymous identity
- CSIv2 X.509 certificate chain identity
- CSIv2 distinguished name identity
- X.509 client certificate
- SAML
- Single Pass Negotiate Identity Asserter (SPNEGO)

Identity Assertion Service

When perimeter authentication is used (see Figure 4-3), a token from outside of the application domain is passed to an Identity Assertion provider responsible for validating the type of token configured as active. If the token is successfully validated, the Identity Assertion provider maps the token to a username, sends that username back, and the authentication process continues. Specifically, the username is sent through a JAAS Callback Handler and is passed to each configured Authentication provider Login Module, so that the Login Module can populate the subject with the appropriate principals. Perimeter-based authentication requires the same components as the authentication process, but includes an Identity Assertion provider.

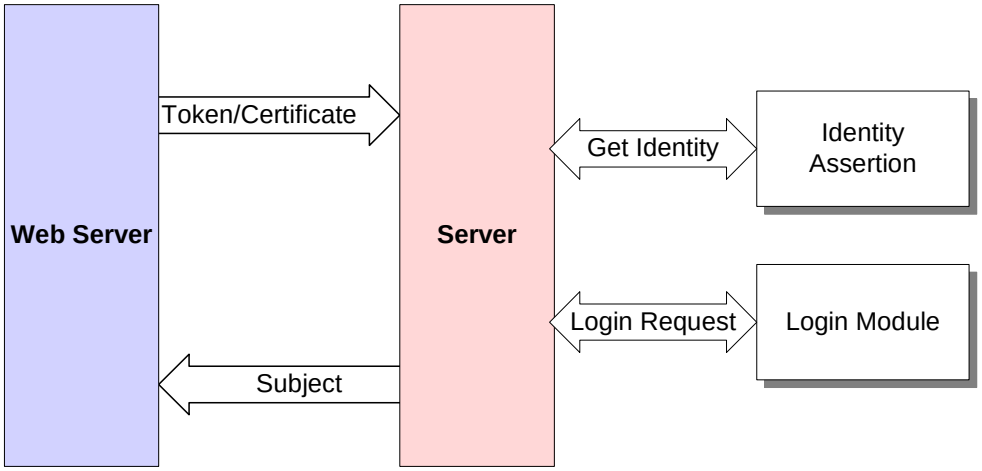
Identity Assertion providers are used as part of perimeter-based authentication process to validate the token type, and then map it to a username. It also specifies a list of trusted client principals to use for identity assertion. The wildcard character (*) can be used to verify that all principals are trusted. If a client is not listed as a trusted client principal, the identity assertion fails and the login is rejected. You can use an Identity Assertion provider in place of an Authentication provider if you create a Login Module for the Identity Assertion provider, or you can use an Identity Assertion provider in addition to an Authentication provider if you want to use the Authentication provider Login Module.

You can configure multiple Identity Assertion providers to support different types of tokens within the same application domain, but you do not need Identity Assertion provider unless you are supporting perimeter authentication. Two Identity Assertion providers are supplied: one supports X.509 and IIOP- CSIv2 and one supports SAML. A third Identity Assertion provider supports identity assertion using HTTP authentication tokens from the SPNEGO protocol, to enable single sign-on Cross-Platform authentication capabilities.

When used with a Login Module, Identity Assertion providers support external authentication of a user based on industry standard methods, for example, perimeter authentication or single sign-on. An Identity Assertion provider can generate a local subject. The Login Module that an Identity Assertion provider uses can be part of BEA’s authentication provider or one developed by a third party.

Unlike simple authentication, the Login Modules that Identity Assertion providers use *do not* verify proof material such as usernames and passwords; they simply verify that the user exists. The token provides the proof that the user is trusted.

Figure 4-3 Identity Assertion Example



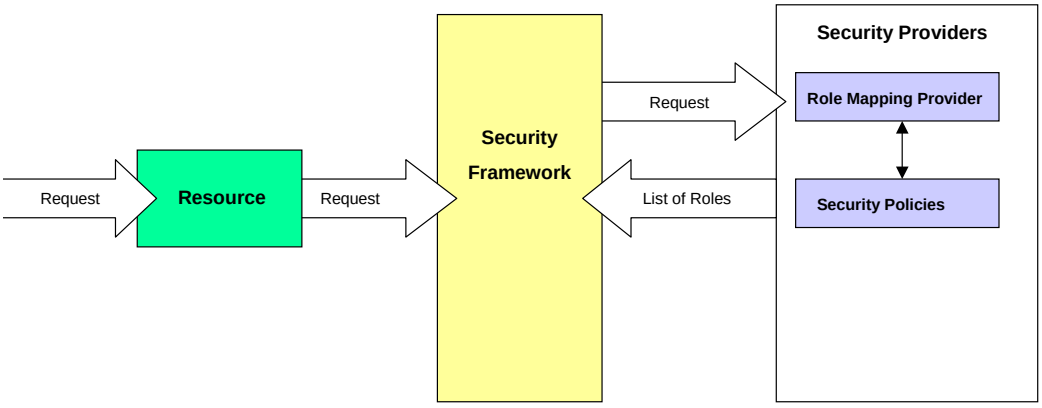
Role Mapping

Role mapping is the process by which the security service compares users or groups against a security role condition to determine whether a security role is granted. Role mapping occurs dynamically at runtime, just before an access decision is rendered for a protected resource.

Role Mapping Service

The Role Mapping provider determines dynamic roles for a specific user with respect to a specific protected resource. The Security Framework calls each Role Mapping provider that is configured as part of an authorization decision. It can also be used to determine the dynamic roles for the specific user to support personalization (for example, with BEA WebLogic Portal 8.1). Figure 4-4 shows how the Role Mapping provider interacts with the Security Framework to create dynamic role associations.

Figure 4-4 Role Mapping Service



The role mapping process is initiated when a user or system process requests a resource on which it attempts to perform an operation. The resource container that handles the type of resource requested receives the request (for example, the EJB container receives the request for an EJB resource). The resource container calls the Security Framework, and then passes in the request parameters, including information such as the subject (users and groups) of the request, the resource, and the action. The Security Framework calls each configured Role Mapping provider to obtain a list of the roles that apply. If an access control policy specifies that the requestor is entitled to a particular role, then the role is added to the list of roles that are applicable to the subject. This process continues until all security policies that apply to the resource or the resource container are evaluated. The list of roles is returned to the framework, where it can be used as part of other operations, such as access decisions.

The result of the dynamic role association performed by the Role Mapping provider is a set of roles that apply to the principals stored in a subject at the time of the request. These roles can then be used to make authorization decisions for protected resources, as well as a resource container and application code. For example, an EJB could use the Java 2 Enterprise Edition (J2EE) method to retrieve fields from a record in a database, without having knowledge of the business policies that determine whether access is allowed.

Authorization

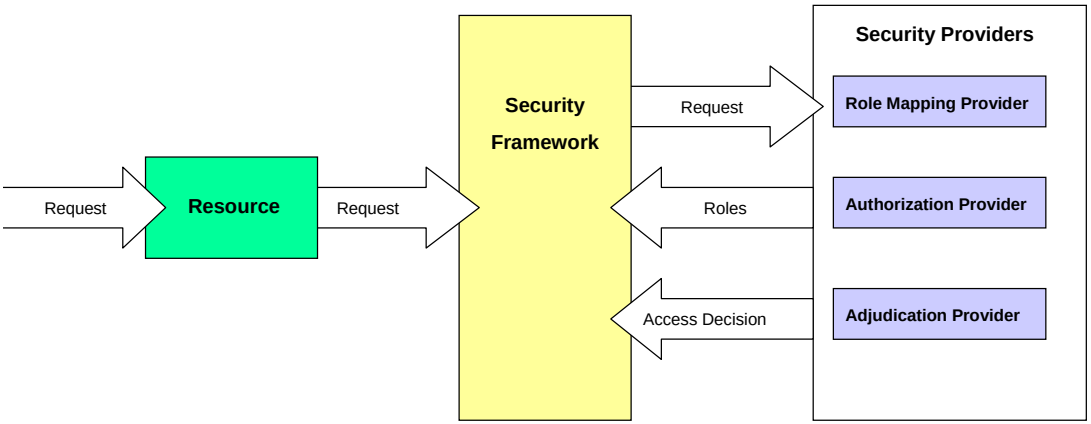
Authorization is the process whereby the interactions between users and resources are controlled, based on user identity or other information. In other words, authorization answers the

question, *Can this user access this resource under these conditions*? An Authorization provider is used to limit the interactions between users and resources to ensure integrity, confidentiality, and availability. You can optionally define a time constraint for an authorization policy to answer the question, *When can the user access the resource?*

Authorization Service

The Authorization service determines whether or not a user can access a resource based on the user's name, group, or role and the authorization policy assigned to the requested resource. Figure 4-5 illustrates how an Authorization provider, and the associated Adjudication and Role Mapping providers, interact with the Security Framework during the authorization process.

Figure 4-5 Authorization Service Example



The authorization process is initiated when a user request is made to perform an operation on a resource. The resource receives the request, calls the Security Framework, and then passes the request parameters to the security providers, including information such as the subject (users and/or groups) of the request, the resource requested, and the action. The Security Framework calls the configured Role Mapping provider using the request parameters to dynamically compute a list of roles to which the user making the request is entitled, and then passes the list of applicable roles back to the Security Framework.

The Authorization provider determines whether the user is entitled to perform the requested action on the resource, based on the user's profile and the supplied resource and the user's role, and makes an access control decision.

The following sections describe authorization concepts and functionality:

- User Directories
- Resources
- Authorization Policies and Role Mapping Policies
- ContextHandlers

User Directories

ALES products can access user directories from a variety of sources, including: application databases, Lightweight Directory Access Protocol (LDAP) directory servers, network databases, and others. The type of information or attributes collected—a method typically referred to as profiling—also varies and typically includes: name and address, phone, e-mail address, personal preferences, and more. The information contained in these directories is the core of any authentication service and is key to providing an effective identity management solution.

Users and group information, along with their attributes, are cached in a database table. User data is then available for access to create or enforce authorization policies.

ALES has no control over your user community; your method of storage is completely under your control. For administrative purposes, however, all users must be a member of at least one group.

Resources

A **resource** is a structured object used to represent an underlying entity that can be protected from unauthorized access using security roles and policies. Resources are hierarchical by nature. Therefore, the level at which you define these roles and policies is up to you. For example, you can define roles and policies on entire EAR, an EJB JAR containing multiple EJBs, a particular EJB within that JAR, or a single method within that EJB. Some typical resources you might want to protect include.

- An application, an application window, or a dialog box
- Specific business transactions, such as a money transfer or security trade
- Application controls, such as buttons and menu selections
- Database or directory server structures
- Web pages, servlets, and EJBs

- Products or services in a portal

Note: This ALES release includes an example that demonstrates the principles of an EJB application functioning under the ALES framework. The example shows how the additional layer provided by ALES can make an EJB application more secure by defining custom dynamic authorization policies without any changes in the application or the deployment descriptors. The example consists of an EJB application that runs on the server side, an EJB client, and the sets of resources, policies, and users that determine the access rights.

The example is installed in `BEA_HOME\ales30-ssm\wls-ssm\examples\EJBAppExample`.

Authorization Policies and Role Mapping Policies

A resource has no protection until it is protected by a policy. In early security systems, an access control list (ACL) was used to protect resources. ALES authorization policies replace ACLs through the use of role-based access control (RBAC). Like Login Modules for Authentication providers, an access decision is the component of an Authorization provider that actually answers the question, *Is access allowed?* Specifically, an access decision is asked whether a subject has permission to perform a given operation on a resource, with specific parameters in an application. Given this information, the access decision responds with a result of `PERMIT`, `DENY`, or `ABSTAIN`.

You create an authorization policy by establishing the permissions, or privileges, for access, based on the resource you want to protect and the role of the requestor; authorization policy focuses on the business process or role, rather than on a user or group. Using RBAC you separate the process of identifying the requestor (defined through a separate authentication process), from the process of authorization (defined by an access decision). The performance improvement gained by this separation is really quite significant. Once the user is authenticated, the access decision can be made more quickly because there is no need to re-authenticate. The importance and benefit of basing access decisions on roles becomes quite apparent as you can assign large groups of users, who's business functions are the same, to the same role. Because role changes less frequently than user or group status, this mechanism is easily scalable.

You assign authorization policies to any of your resources (for example, a URL, an EJB resource or a JNDI resource) or to attributes or operations of a particular instance of a resource (for example, an EJB method or a servlet within a web application). If you assign a authorization policy to a type of resource, all new instances of that resource inherit that policy.

To use a user or group to create an authorization policy, the user or group must be defined for the authorization provider that is configured for the service module to which the provider is associated. For efficiency, this is typically done through the use of roles rather than user or

groups. To use a role to create a role mapping policy, the role must be defined for the Role Mapping provider that is configured for that service module. All authorization and role mapping policy data is stored in the policy database. When the Authorization provider and Role Mapping provider are configured, their configurations are also stored in the policy database.

Note: Although you have the option of basing an authorization policy on a user or group, BEA recommends basing authorization policies on roles. Basing authorization policies on roles is a more efficient method of management.

Built-In Attribute Retrievers

This release of ALES provides built-in attribute retrievers that you can configure directly in the console. Attribute retrievers are used by ASI Authorization and ASI Role Mapping providers to retrieve attributes for use in policies. In prior releases of ALES, you needed to write code to create an attribute retriever, as described in Attribute Retriever.

Built-in attribute retrievers are typically configured with a set of connection properties and credentials to connect to the data source. The configured attribute retriever can have a set of attributes with queries defined to retrieve the attribute value during policy evaluation. The attribute retrievers take care of executing the query against the configured data source during policy evaluation, and return the result to ALES to make a decision.

Any changes to the configured attribute retrievers would need to be distributed to the SCMs, and the SSMs restarted for the new configuration to take effect.

If one of the preconfigured attribute retriever types does not meet your needs, you can still create a custom attribute retriever from code and then manage it from the console.

The following attribute retrievers types are provided. See the online help for information on configuring each specific attribute retriever type.

- RDBMSAttribute Retriever
- LDAPAttribute Retriever
- SDOAttribute Retriever
- ALESIdentity Attribute Retriever
- Custom Attribute Retriever

RDBMS Attribute Retrievers

RDBMS Attribute Retrievers retrieve attribute values from an RDBMS database. You configure the retriever with a database URL, driver, and credentials to access the database. The driver URL

can have a comma separated list of server URLs for failover. You then create attributes for these retrievers, with a query to retrieve attribute values. For example, to retriever the “user_type” from a database for a given user, you could use a query similar to `SELECT user_type FROM customer WHERE user_id=%sys_user%`. Queries can optionally contain ALES-defined system attributes, as in the example. Multiple attributes can be defined with comma-separated list of names for the attribute name field. If so defined, the query would have to contain multiple attributes in the same order as defined in the attribute name. For example, if the attribute name is defined as `id,postalcode`, the query could be similar to `SELECT id, postalcode FROM customer`. Optionally, attribute values can be cached based on the business requirement using the `use cache` and the TTL configuration parameters.

The created built-in attributes require an equivalent ALES dynamic attribute with the same name to be used as part of constraints for policy evaluation.

LDAP Attribute Retrievers

LDAP Attribute Retrievers retrieve attribute values from an LDAP database. You configure the connection properties, such as host name, port and credentials. Host name can be a comma-separated list of servers if failover is desired. You then create built-in attributes for the configured retriever with a query to locate the attribute. The query can be an entry of the form `uid=%sys_user%,cn=employees,ou=enterprisesecurity,ou=security,dc=amer,dc=bea,dc=com`. Optionally, configure the filter attribute if the search needs to be further narrowed down. For example `&(objectclass=*)(transactioncode= %sys_rule_obj_q%`. Note that the query and filters attributes can have optional ALES system attributes that are substituted with values before the query is run.

Make sure to create ALES dynamic attributes with the same name as that of the built-in attributes to be used as part of constraints.

Service Data Objects (SDO) Attribute Retrievers

Service Data Objects (SDO) is a joint BEA/IBM specification that identifies a unified framework for data application development. It works with data from multiple data sources in the form of physical or Logical data services from ALDSP. The ALDSP services can in turn depend on multiple data services to retrieve data.

To retriever attribute values from an ALDSP server, configure an SDO Attribute retriever. Configuration of the SDO retriever requires the ALDSP server lookup information and credentials. The service lookup entry would be of the form `ld:DataServices/CustomerManagement/CustomerProfile`. The servers can be a comma-separated list of servers for failover. You also configure the initial context factory

implementation to use, and make sure the jars are available in the classpath at runtime. For example, if the WebLogic initial context factory is used (`weblogic.jndi.WLInitialContextFactory`), make sure the WebLogic client jar is available in the CLASSPATH at runtime. Otherwise, you generate a ClassNotFoundException exception.

The application name entry is the name of the application to connect to on the hosted ALDSP server. Configure the login credentials to connect to the server.

Note: In this release SDO attribute retrievers are supported only when running on WLS SSMs. Also, there is no caching support for SDO Attributes with this release.

ALES Identity Attribute Retrievers

ALES Identity Attribute Retrievers are used to retrieve the value of an identity attribute from the ALES database. In releases of ALES prior to 3.0, metadirectory configuration was used to access identity attributes from ALES. This was done by configuring the "Use Metadirectory" option on the ASI Authorizers page. This functionality is now available via the ALES Identity Attribute Retriever.

Create the retriever with the driver URL, connection properties, and credentials to connect to the ALES database. There is no need to create attributes for the ALES Identity Retriever. You create ALES identity attributes using the ALES Administration Console and then use the attributes in the policy constraints as needed.

Make sure to associate and initialize the newly-created identity attributes to the respective identity directories.

Custom Attribute Retrievers

Custom Attribute Retrievers are used if the built-in attribute retrievers are not suitable for your needs. You could create a custom attribute retriever plug-in and configure a custom retriever using the console. I

In the prior release, custom retrievers were configured on the advanced settings of the ASI Authorization and Role Mapping provider. In the current release, this configuration is done by creating a new custom retriever for the ASI Authorization provider and then specifying the fully-qualified package (with classname) for the plug-in to be loaded at runtime.

Creating attributes for the custom retrievers is optional and should be done only when attribute-specific caching is required. To do this, create one or more dynamic attributes and use them with constraints for policy evaluation.

Upgrade of Metadirectory and Existing Attribute Retrievers at Installation

When upgrading from a prior version of ALES, the configuration related to Metadirectory and custom attribute retrievers is preserved:

- If there is a MetaDirectory configuration available in the prior versions, a new ALES Identity Retriever is added to the Attribute Retrievers tab.
- If there are any custom retrievers configured, they are shown as individual custom attributes with the same names as the plugins.

Configuration of Attribute Retriever on WLS 9.1/10.0 SSM is Through WebLogic Server

The WebLogic Server 9.x/10.0 SSM uses a different security framework from the one used in the WLS 8.1 SSM and the other ALES SSMs.

As a consequence, when you use the WLS 9.x/10.0 SSM you must use the WebLogic Administration Console to configure attribute retrievers and their related attributes. The configuration process is similar to that of the other SSMs types on the ALES Administration Console.

ContextHandlers

A ContextHandler is a high-performing class that obtains additional context and container-specific information from the resource container, and provides that information to security providers making access or role mapping decisions. The ContextHandler interface provides a way for an internal resource container to pass additional information to a Security Framework call, so that a security provider can obtain contextual information beyond what is provided by the arguments to a particular method. A ContextHandler is essentially a name/value list and, as such, it requires that a security provider know what names to look for. In other words, use of a ContextHandler requires close cooperation between the resource and the security provider. Each name/value pair in a ContextHandler is known as a context element and is represented by a Context Element object.

For example, with WebLogic Server 8.1, three types of resource containers pass Context Handlers to the Security Framework: the Servlet, EJB, and Web Service containers. Thus, URL (web), EJB, and Web Service resource types have different context elements whose values Authorization and Role Mapping providers can inspect. An implementation of the Audit Context interface (used when a security provider is implemented to post audit events) may also examine the values of context elements.

Adjudication

Adjudication involves resolving any authorization conflicts that may occur when more than one Authorization provider is configured in a application domain, by weighing the result of each Authorization provider's access decision. An Adjudication provider tallies the results that multiple access decisions return, and determines the final `PERMIT` or `DENY` decision. An Adjudication provider may also specify what to do when an answer of `ABSTAIN` is returned from an access decision from a single Authorization provider.

Adjudication Service

If there are multiple Authorization providers configured, the Security Framework delegates the job of reconciling any conflicts in the access decisions rendered by the Authorization providers to the Adjudication provider, which determines the ultimate outcome of the access decision. The Adjudication provider returns either a `TRUE` or `FALSE` verdict to the Authorization providers, and then forwards it to the resource through the Security Framework.

Auditing

In the ALES architecture, an Auditing provider is used to support auditing services, for example, using the Log4j Auditing Channel provider. An auditing service is used to collect all security event information and distribute that information as configured.

If configured, the Security Framework calls the Auditing provider before and after security operations are performed (such as authentication or authorization), enabling audit event recording. The decision to audit a particular event is made by the Auditing provider itself and can be based on specific audit criteria and severity levels. The records that contain the audit information may be written to output repositories such as an LDAP server, a database, or a file.

Auditing Service

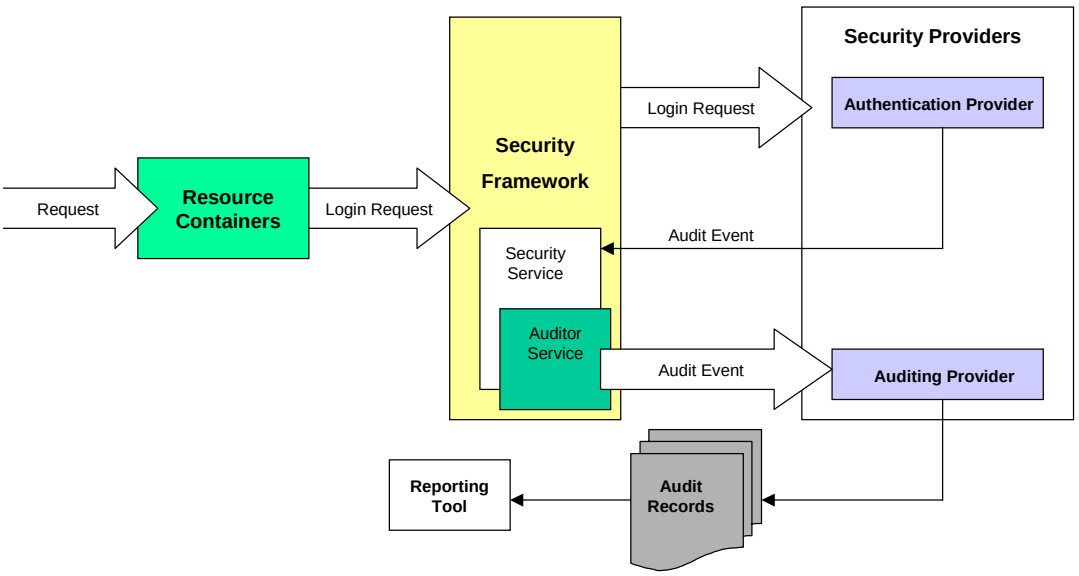
An auditing service is a process whereby information about service requests and the outcome of those requests are made available to configured auditing providers. Auditing provides an electronic trail of transaction activity, and can include changes to system configuration parameters, policy changes, transactions, and security breach attempts of any type. For each audit item, the information may include who, what, when, where, and sometimes why.

Records can be directed to a file, a database or other persistent storage medium, and then used to generate reports from any reporting generating software you choose. Figure 4-6 shows how the

Auditing provider interacts with the Security Framework and other types of security providers using a Authentication provider as an example.

The auditing process is initiated when a resource container passes authentication information for a user (for example, a username and password combination) to the Security Framework as part of a login request. The Security Framework passes the information associated with the login request to the configured Authentication provider. The Authentication provider passes the audit event back through the Security Framework, which then makes that audit event available to the configured Auditing provider.

Figure 4-6 Auditing Service Example for Authentication



Performance Statistics

The performance statistics feature enables the collection of data about authentication and authorization for purposes of troubleshooting and performance analysis. The performance statistic feature is controlled by an Auditing security provider, the PerfDBAuditor provider. Performance statistics are gathered for each Security Service Module in your AquaLogic Enterprise Security installation. In order to collect performance statistics for an SSM, you must enable and configure a PerfDBAuditor provider for that SSM.

See the *Performance Statistics* "how to" document for additional information.

Credential Mapping

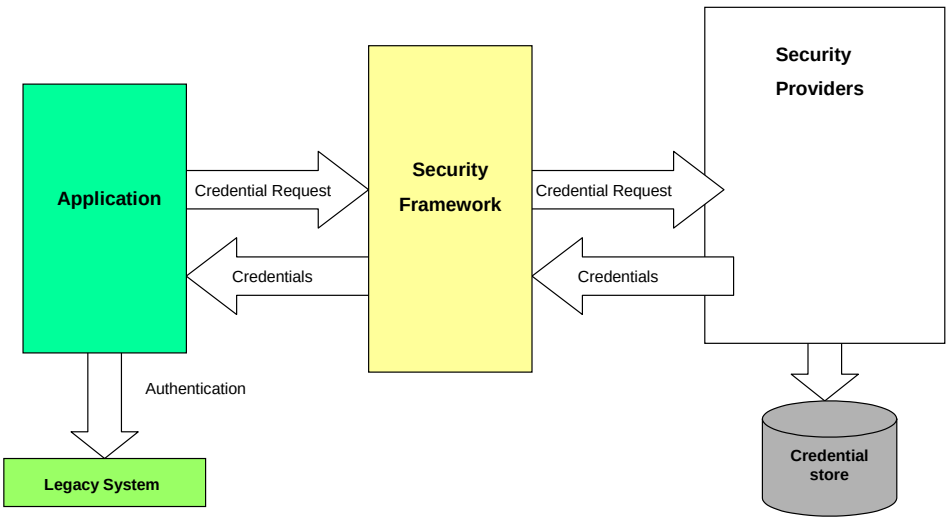
Credential mapping is the process of looking up the authentication credentials associated with a user for a given resource.

Credential Mapping Service

The Credential Mapping service provides authentication credentials for a user of another application, for example, some type of mainframe transaction processing, a database system, or a legacy application. Three Credential Mapping providers are supplied: one supports usernames and passwords stored in a relational database; one supports usernames and passwords stored in a WebLogic Server 8.1 embedded LDAP directory (this provides support for WebLogic Portal 8.1); and one produces SAML credentials.

Figure 4-7 illustrates how the Credential Mapping provider interacts with the Security Framework to support the credential mapping service.

Figure 4-7 Credential Mapping Service



Security Service Providers

Security services are based on a set of Security Service Provider Interfaces (SSPIs). Developers and third-party vendors can use the SSPIs to develop custom security providers for ALES. SSPIs

are available for Authorization, Role Mapping and Adjudication; Auditing; Authentication, and Identity Assertion; and Credential Mapping (for a description of providers, see Table 4-1). The SSPIs allow customers to use their security providers to secure resources. Customers can use the SSPIs to develop custom security providers or they can purchase customer security providers from third-party vendors.

Table 4-1 Security Providers

Provider Type	Provider Name	Description
AquaLogic Enterprise Security Providers		
Authentication	Novell Authenticator	Authenticates users using a Novell LDAP directory.
	Database Authenticator	Authenticates users using the ALES relational database provider.
	Open LDAPAuthenticatorr	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.
Identity Assertion	X.509 Identity Assertion Provider	Supports identity assertion through an X.509 digital certificate (ASN.1 encoding and decoding).
	SAML Identity Assertion Provider	Supports identity assertion through Security Assertion Markup Language v1.1.
	Single Pass Negotiate Identity Asserter	Supports identity assertion using HTTP authentication tokens from the SPNEGO protocol. The provider supports the identity assertion using the Kerberos tokens contained within the SPNEGO token.
	ALES Identity Assertion Provider	Supports identity assertion through an ALES cookie.

Table 4-1 Security Providers (Continued)

Provider Type	Provider Name	Description
Credential Mapping	Database Credential Mapping Provider	Provides authentication credentials for a user (username and password) from a database for use in another application.
	SAML Credential Mapping Provider	Provides authentication credentials for a user (a SAML assertion) for use in another application.
	ALES Credential Mapping Provider	Provides authentication credentials for a user (a ALES cookie) for use in another application.
Auditing	Log4j Audit Channel Provider	Provides auditing features that support the Apache Log4j capabilities.
Authorization	Authorization Provider	Provides enforcement of authorization for the policy with which it is used. The Authorization Provider returns an access decision that determines which resources are protected and if a particular user is allowed access to a resource.
	Adjudication Provider	Provides an authorization decision based on results from one or more authorization providers.
Role Mapping	Role Mapping Provider	Provides assignment to roles based on the access control policy with which it is used. The Role Mapping Provider returns a set or roles granted to a user on a protected resource.

Table 4-1 Security Providers (Continued)

Provider Type	Provider Name	Description
WebLogic Server 8.1 Providers	WebLogic Authentication Provider	Supports authentication through an embedded LDAP Directory for WebLogic Server.
	WebLogic Authorization Provider	Provides enforcement of authorization for the access control policy in WebLogic Server. The Authorization Provider returns an access decision that determines which resources are protected and if a particular user is allowed access to a resource.
	WebLogic Role Mapping Provider	Provides assignment to roles based on the access control policy defined for use with WebLogic Server 8.1. The WebLogic Role Provider returns a set or roles granted to a user on a protected resource.
	WebLogic Credential Mapping Provider	Provides authentication credentials for a user (username and password) for single sign-on use into another application or resource.

- Configuring WebLogic Security Providers
- Configuring Authentication Providers

In addition, you can use the following AquaLogic Enterprise Security 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: To assist customers in developing custom security providers, sample custom security providers are also available from the BEA dev2dev web site at http://dev2dev.bea.com.

For more information on developing custom security providers, see *Developing Security Providers for BEA AquaLogic Enterprise Security*.

CHAPTER 5

Security Administration

This section covers the following topics:

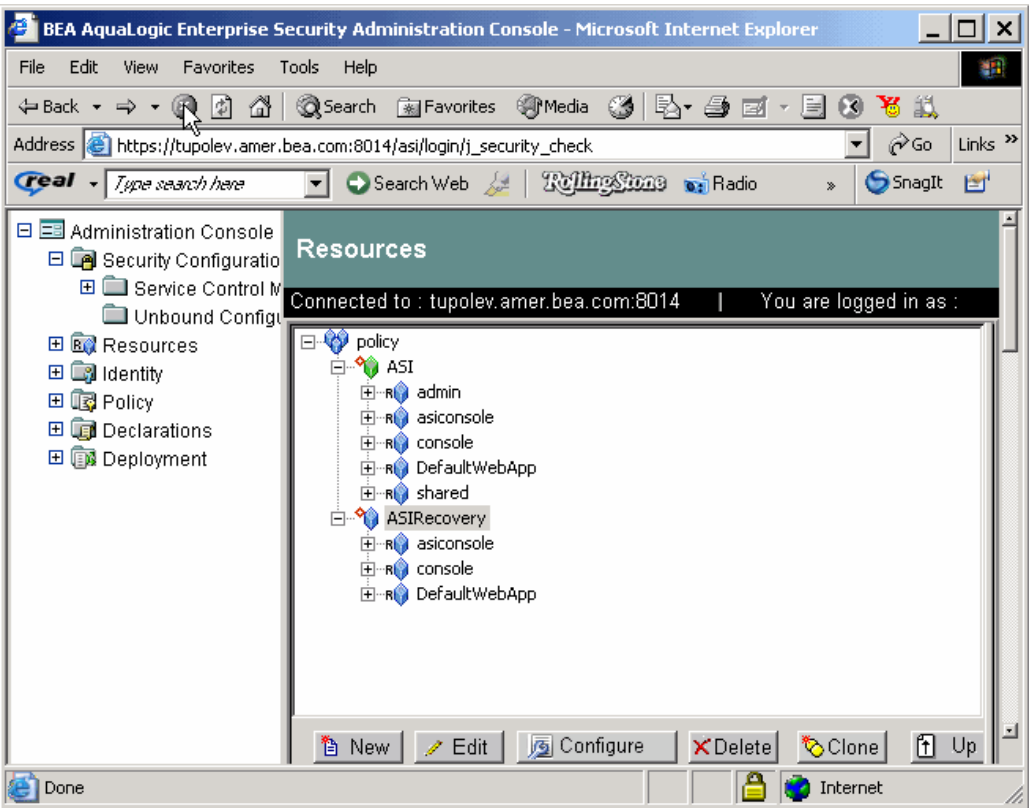
Managing Security

You manage security by using the Administration Console and Entitlements Management Tool to configure SSMs, and to design, edit, and distribute access control policy.

Security policies are built from policy elements: identity (users and groups with their associated attributes), privileges, roles, and the resources you want to protect. A policy can be applied to either a single resource or to many resources. An enterprise-wide policy represents the superset of all of your authorization and role mapping policies. The following sections describe security management concepts and how you implement security using the Administration Console:

Figure 5-1 shows how the Administration Console represents the various policy elements and resources. The resources shown here represent the initial policy settings used to protect the console.

Figure 5-1 Administration Console Resource Representation



Security Configuration

A Security Configuration is represented by the  icon in the Administration Console, and consists of one or more Service Control Managers, one or more Security Service Modules represented by the  icon, and the providers associated with each module. You can use the security providers that are provided, purchase custom security providers from third-party security vendors, or develop your own custom security providers. Table 5-1 shows how each provider type is represented in the Administration Console.

Table 5-1 Security Providers and their Representations

	Adjudication		Authorization
	Auditing		Credential Mapping
	Authentication		Role Mapping

To learn more about security providers and their services, see "Architecture Overview" on page 3-1. For information on how to develop custom security providers, see *Developing Security Providers for AquaLogic Enterprise Security*.

A configuration is a named collection of elements and security services over which a common and consistent set of policies are administered in a coordinated fashion. Each configuration has a unique Configuration ID and is bound to a Service Control Manager. Applications that use the same Configuration ID reference the same user communities, policy, and security configuration.

A configuration also defines the method of authentication and authorization used to provide access to a resource for one or more users and groups. The authorization policy protects the resource against unauthorized access. Each Security Service Module has a Configuration ID associated with it that defines the security providers, policy, and settings for that Security Service Module.

Resources

A resource is a general term that refers to any object that you can protect. Resources can include applications, data, or system components. Resources may also include background services with which the user has no direct interaction. For example, a bank might offer banking services on a web page that simulate the actions of a teller. The components (buttons, tables, data fields) displayed on that web page are all resources. The Administration Console displays resources in a tree structure called a resource hierarchy as shown in Figure 5-2. The resources shown here represent the administration services and policy data elements that are accessible to the Administration Server on startup.

A page generated from a JSP is also a resource. The page can call EJBs or COM resources to execute a transaction. The back office services that transfer money between accounts, issue a payment, or run a report are also resources, although they may not appear on the web page or execute on the application server.

Individual resources in the hierarchy are also called nodes and the type of node can convey additional information about the resource. Some nodes are organizational because they represent a logical grouping of resources. For example, an organizational node called accounting may represent all resources in the accounting department. A node representing a group of resources is represented by a blue resource icon  .

Another type of organizational node is an application node represented by a green resource icon  . This icon depicts a collection of resources that provide a specific set of services. Applications are considered organizational because they often consist of multiple, distributed components, such as an *n*-tier web application that may have resources on a web server, an application server, and a data server. However, an application node can also be used to represent an application with a single component, such as a desktop spreadsheet application.

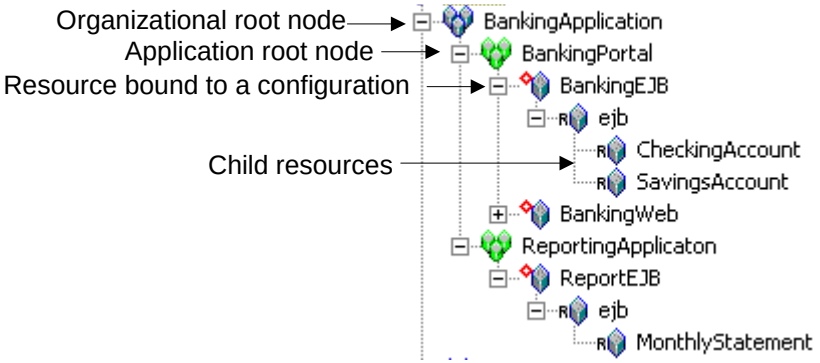
Individual resource nodes are represented by the  icon, and typically represent an individual resource that can be protected. An administrator may define as many resources and levels in the hierarchy as needed to represent data, services and system components within an application.

A special node, called a binding node, is depicted by a resource or application node with a small red diamond  . A binding node is used to associate (or bind) a subset of the resource tree to a particular configuration.

Any resource at or above a binding node can be marked as a policy distribution point. When a policy distribution is initiated, you can choose to distribute either all updates (by selecting the root node) or you can limit which updates are distributed by selecting resources from the existing distribution points. Only updates that were made at or below the selected distribution points are

distributed. Figure 5-2 shows how organizations, applications and resources are represented and bound to Security Service Modules in the Administration Console.

Figure 5-2 Organizations, Applications, and Resources form your Security Configuration



Some typical resources that you might want to secure, include:

- An application, an application window, or a dialog box
- Specific business transactions, such as a money transfer or security trade
- Application controls, such as buttons and menu selections
- Database or directory server structures
- Web pages (URLs), servlets, and Enterprise Java Beans (EJB)
- Products or services through BEA WebLogic Portal

The following topics provide more information:

- "Resource Attribute" on page 5-5
- "Privilege" on page 5-6
- "Privilege Group" on page 5-6

Resource Attribute

A resource attribute is represented by the  icon in the Administration Console. All resources can have attributes, which store information about the resources to which they belong. Common resource attributes might be a file type, resource owner, or the creation date.

Attributes are inherited by child resources from their parent. If a resource explicitly sets the value for an attribute, the inherited value overrides the resource value.

Privilege

A privilege is represented by the key  icon in the Administration Console and is an action that you can perform on a resource. For instance, execute is a typical application privilege; and read and write are typical file-system privileges.

You can use the privileges provided or you can create your own. Figure 5-3 shows how privileges appear in the Administration Console. Notice that each privilege refers to an action. A related collection of privileges may be organized into a privilege group for management purposes.

Figure 5-3 Privilege Representation in the Administration Console

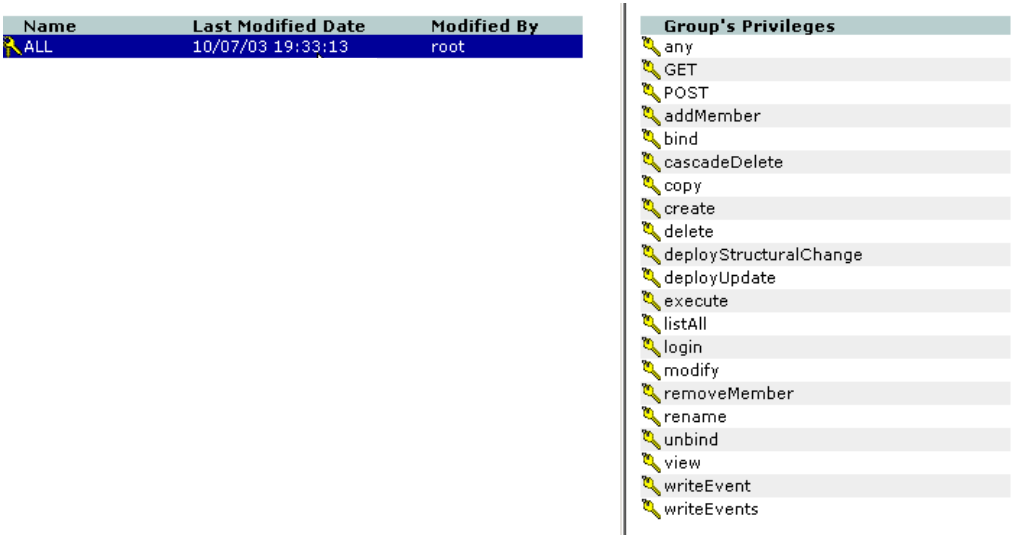
Name	Last Modified Date	Modified By
any	10/07/03 19:33:13	root
GET	10/07/03 19:37:04	//user/wles/system/
POST	10/07/03 19:37:04	//user/wles/system/
addMember	10/07/03 19:36:24	//user/wles/system/
bind	10/07/03 19:36:24	//user/wles/system/
cascadeDelete	10/07/03 19:36:24	//user/wles/system/
copy	10/07/03 19:36:24	//user/wles/system/
create	10/07/03 19:36:24	//user/wles/system/
delete	10/07/03 19:36:24	//user/wles/system/
deployStructuralChange	10/07/03 19:36:24	//user/wles/system/
deployUpdate	10/07/03 19:36:24	//user/wles/system/
execute	10/07/03 19:36:24	//user/wles/system/
listAll	10/07/03 19:36:24	//user/wles/system/
login	10/07/03 19:36:24	//user/wles/system/
modify	10/07/03 19:36:24	//user/wles/system/
removeMember	10/07/03 19:36:24	//user/wles/system/
rename	10/07/03 19:36:24	//user/wles/system/
unbind	10/07/03 19:36:24	//user/wles/system/
view	10/07/03 19:36:24	//user/wles/system/
writeEvent	10/07/03 19:37:04	//user/wles/system/
writeEvents	10/07/03 19:37:04	//user/wles/system/

Privilege Group

A privilege group is represented by the keys  icon in the Administration Console and allows you to organize privileges into logical groups for ease of management. For example, it is common to define a privilege group that applies to a particular application or set of transactions. Privilege

groups can be used as filters when constructing policies, although they cannot appear directly in the policy. Figure 5-4 shows an example of how privilege groups and their associated privileges appear in the Administration Console.

Figure 5-4 Privilege Group Representation in the Administration Console



Identity

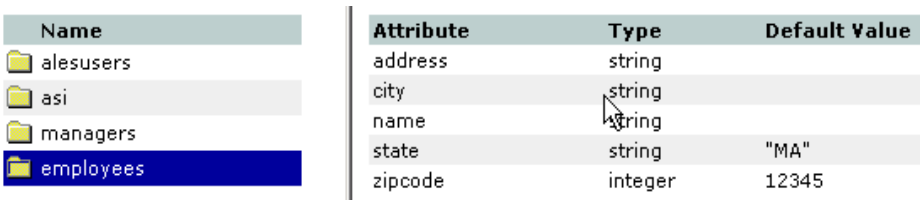
The Identity icon represents your directories and user communities in the Administration Console. Although ALES provides tools to manage users and groups locally, they are typically managed through an external repository, such as a Lightweight Directory Access Protocol (LDAP) directory server or a network database. The recommended approach is to use an Attribute Retriever, which can go directly to the data store, or multiple Attribute Retrievers if you need to access multiple data stores simultaneously for a unified view. You can also use AquaLogic Data Services Platform for data aggregation.

Note: Identity data are used to calculate roles used in your authorization policy and is not used for authentication purposes. Authentication is supported through your external repositories by configuring an authentication provider.

A directory typically represents groups of users of a particular application or resource, or users in a specific location. Each directory has an associated attribute schema. The schema defines the

attributes applied to members of the directory. Figure 5-5 shows how directories are represented in the Administration Console. In this example, there are two directories: ales and Employees. The `employees` directory shows the attributes that are stored for each member of the directory: zipcode, state, name, city and address. The zipcode and state attributes have default values set.

Figure 5-5 Directory Representation in the Administration Console



The number of directories you have depends on the level of granularity needed to separate your user community. You may want to have one global directory containing all users. In this case, you can populate a single directory using multiple external repositories. Or, you might want separate directories—one directory for customers, one for employees, and one for partners, where the method of authentication is different for each group. Having one directory requires a unique name for each user and group. If you are not able to guarantee this when you integrate your repositories, you should maintain separate directories.

The following topics provide more information about Identity:

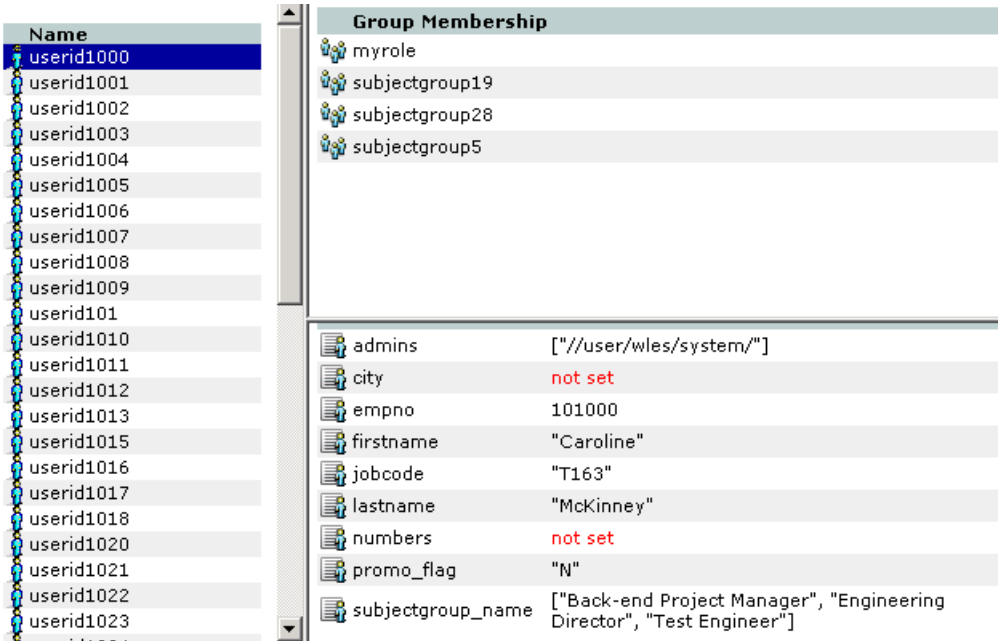
- "Users" on page 5-8
- "Groups" on page 5-9
- "Identity Attributes" on page 5-10
- "Roles" on page 5-11

Users

A user is represented by the  icon in the Administration Console and corresponds to an individual who makes a request to access a resource, although a user can be an automated process that accesses the system. Users are included in an authorization policy by assigning users to groups, and then assigning that group to a role. Each user within a directory must have a unique identity or user name.

Users can be associated with certain characteristics, referred to as identity attributes; these attributes store information about the user. The list of attributes that can be set for a user is dictated by the attribute schema of the directory to which the user belongs. Figure 5-6 shows an example of a user representation with identity attributes. In this example, `userid1000` belongs to four groups and there are nine attributes associated with the user.

Figure 5-6 User Representation in the Administration Console



Groups

A group is represented by the  icon in the Administration Console and is a logical collection of users that share some common characteristics, such as department, job function, or job title. For example, a company may separate its sales staff into two groups, Sales Representatives and Sales Managers, because they want their sales personnel to have different levels of access to resources depending on their job functions.

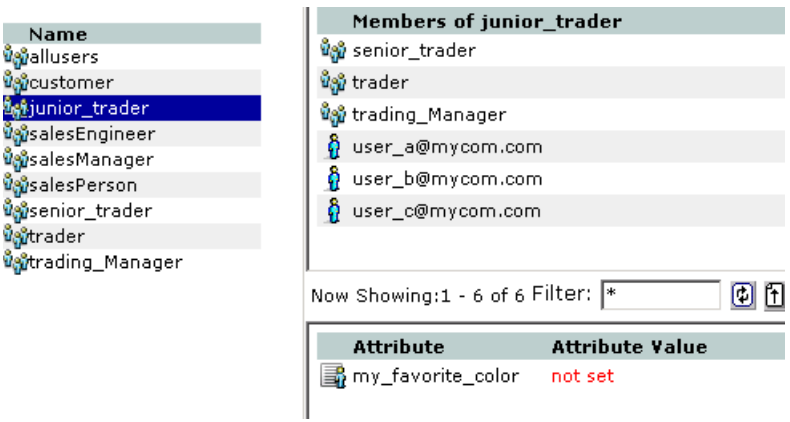
A group can contain either users or other groups; users who are assigned to a group are called group members. Nested memberships of groups within a group form a hierarchy. Group

membership can be assigned only from within the same directory. Groups have a static identity that an administrator assigns. All group names must be unique within a application domain.

Managing groups is more efficient than managing large numbers of users individually. By using groups, you do not need to define policy for each and every user. Instead, each user in the group inherits the policies applied to the group; this rule also applies to nested groups. Granting a permission or role to a group is the same as giving that permission or role to each user who is a member of the group. For example, an administrator can specify roles for 50 users at one time by placing the users in a group, and then granting that group the role on a given resource.

Figure 5-7 shows how groups are represented in the Administration Console. In this illustration, the junior_trader group contains three group members, three user members, and one attribute (my_favorite_color).

Figure 5-7 Group Representation in the Administration Console



Identity Attributes

Identity attributes are represented by the  icon in the Administration Console (as shown in Figure 5-7). Each user and group can have different characteristics defined as identity attributes. The type of information or attributes collected—a method typically referred to as profiling—also varies and typically includes information such as name and address, phone, e-mail address, personal preferences, and so forth. Identity attributes can be extracted from the external data source.

An identity attribute is declared specifically to contain identity information. An attribute value can be used in policies to set limits for that user. Attributes provide a very powerful way to refer to users and groups indirectly in policies, which results in a more dynamic and versatile access control policies.

Roles

A role is represented by the  icon in the Administration Console. A role is a dynamic alias used to associate users and groups to policy-based functional responsibilities. A role represents a collection of privileges on a resource. Roles are computed and granted to users or groups dynamically, based on conditions, such as user name, group membership, identity attributes, or dynamic data, such as the time of day. Roles membership can apply to only specific resources within a single application or can be applied globally across the enterprise, based on how you distribute the policy. A role can also be delegated from one user to another user.

Granting a role to a user or a group confers the defined access privileges to that user or group, as long as the user or group is a member of the role. Multiple users or groups can be granted a single security role.

Policy

Authorization policy controls what actions a user can perform on a resource. Authorization policies are typically written to grant specific privileges upon resources to users within a given role under a defined set of conditions.

The following topics provide more information:

- “Role Mapping Policies” on page 5-11
- “Authorization Policies” on page 5-12
- “Role Mapping Policy Reports” on page 5-12
- “Authorization Policy Reports” on page 5-12

Role Mapping Policies

A role mapping policy is represented by the  icon in the Administration Console and displays the policies that determine the membership for each role you have created, to which a user or group can be assigned. BEA recommends defining roles, and then assigning users and groups to

those roles. A delegation policy assigns a role or privilege on a resource from one user or group (delegator) to another user or group (delegate). To delegate, the delegator must first have the role assigned to them. Role mapping policies are used to assign users or groups to membership in roles. The membership can be limited based on a number of items including the resource hierarchy, identity, contextual, and resource attributes. Roles may also be delegated from one user to another using delegation policies.

Authorization Policies

Authorization policies are represented by the  icon in the Administration Console and displays the policies that determine what actions can be performed on a resource. Authorization policies are typically written to grant specific privileges upon specific resources to a role with a defined set of constraints. An authorization policy can define privileges, resources, policy subjects (users, groups, and roles), constraints, and delegators. For ease of policy management, BEA recommends that authorization policies be written to grant privileges to roles, rather then granting them directly to users or groups. A delegation policy that delegates a privilege assigns the privileges granted to one user (the delegator) on a resource to another user, group, or role.

Role Mapping Policy Reports

Role mapping policy reports are represented by the  icon in the Administration Console Using this function you can create a role mapping policy inquiry and use it generate a report that you can use for analysis. Role mapping policy inquiries search for role-based, access-control policies written against specific resources that match specified characteristics exactly. You can define inquiries that include a policy subject list (user and group), a role list, a resource list, and a delegator list. Role mapping policy inquiries ask the question, What role can do what to what resource?

Authorization Policy Reports

Authorization policy reports are represented by the  icon in the Administration Console. Using this function you can create an authorization policy inquiry and use it generate a report that you can use for analysis. Authorization policy inquiries search for privilege-based policies that match specified characteristics exactly. You can define inquiries that include a policy subject list (user, group and role), a privilege list, a resource list, and a delegator list. Authorization policy inquiries ask the question, Who can do what to what resource?

Declarations

Declarations are represented by the  icon in the Administration Console. A declaration is a variable that represents either a predefined value (for example, days of the week) or a value that is dynamically defined at runtime (the date). To help you design efficient policies, various built-in declarations are provided for your use. There are four types of declarations:

- **enumerated type**—A type that consists of a predefined list of ordered values from which you create constants and attributes. The system comes with a number of predefined enumerated types and you can define your own. For example, you could define the enumerated type "color" with the values of "red", "green", or "blue".
- **constant**—A named, predefined, static value, or set of values that you can reference in a policy for a value that does not change at runtime.
- **attribute**—Represents characteristics that define dynamic values, users, groups, resources and configurations. An attribute has an associated type which may either be a built-in type (such as string, integer, date) or an enumerated type.
- **evaluation function**—A named function that you can use in a policy condition to perform more advanced operations. Each function may have a number of parameters and returns a Boolean result. There are a number of built-in evaluation functions and you can declare and use your own custom evaluation functions. Each custom evaluation function must be registered as a plug-in with the authorization and role mapping engine (ARME) using it.

Deployment

Deployment is represented by the  icon in the Administration Console.

Once you have designed your access control policies, you must deploy it before it can take effect in your environment. Before you deploy policy, choose the distribution point for the policy. The distribution point identifies what portions of the policy updates are made active in your environment. After the distribution, you can view the results of the policy distribution. You also distribute security configuration data. After a configuration update, you must restart the effected modules to make use of the new configuration; this means you must restart the Security Service Module component and your application.

The Deployment Status page allows you to gather information regarding policy and configuration deployments. The page shows if any components are out of sync with the central Administration Server.