BEA WebLogic Server™

Introduction to WebLogic Security

Version 8.1
Revised: August 2005

Copyright

Restricted Rights Legend

Information in this document is subject to change without notice and does not represent a commitment on the part of BEA Systems. THE SOFTWARE AND DOCUMENTATION ARE PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. FURTHER, BEA Systems DOES NOT WARRANT, GUARANTEE, OR MAKE ANY REPRESENTATIONS REGARDING THE USE, OR THE RESULTS OF THE USE, OF THE SOFTWARE OR WRITTEN MATERIAL IN TERMS OF CORRECTNESS, ACCURACY, RELIABILITY, OR OTHERWISE.

Trademarks or Service Marks

Contents

About This Document

1. Overview of the WebLogic Security Service

2. Security Fundamentals

3. Security Realms

4. WebLogic Security Service Architecture

About This Document

The document summarizes the features of the new WebLogic® Security Service and presents an overview of the architecture and capabilities of the WebLogic Security Service. It is the starting point for understanding the WebLogic Security Service.

This document is organized as follows:

- Chapter 1, "Overview of the WebLogic Security Service" introduces the WebLogic Security Service, describes the audiences of this document, lists its key features, and gives a brief list what has changed in this release.
- Chapter 2, "Security Fundamentals" describes security concepts as they relate to BEA WebLogic Server™ security. This section includes discussions of auditing, authentication, authorization, Secure Sockets Layer (SSL), firewalls, and the relationship between J2EE and WebLogic security.
- Chapter 3, "Security Realms," describes security realms, which are used to protect WebLogic resources.
- Chapter 4, "WebLogic Security Service Architecture," describes the WebLogic Server Security architecture. This section includes discussions of the WebLogic Security Framework, the Security Service Provider Interfaces (SSPIs), and the WebLogic security providers that are included as part of the product.
- Chapter 5, "Terminology," defines key terms that you will encounter throughout the WebLogic Server security documentation.

Audience for This Guide

This document is intended for the following audiences:

- Application Architects—Architects who, in addition to setting security goals and designing the overall security architecture for their organizations, evaluate WebLogic Server security features and determine how to best implement them. Application Architects have in-depth knowledge of Java programming, Java security, and network security, as well as knowledge of security systems and leading-edge security technologies and tools.

- Security Developers—Developers who focus on defining the system architecture and infrastructure for security products that integrate into WebLogic Server and on developing custom security providers for use with WebLogic Server. They work with Application Architects to ensure that the security architecture is implemented according to design and that no security holes are introduced, and work with Server Administrators to ensure that security is properly configured. Security Developers have a solid understanding of security concepts, including authentication, authorization, auditing (AAA), in-depth knowledge of Java (including Java Management eXtensions (JMX), and working knowledge of WebLogic Server and security provider functionality.

- Application Developers—Developers who are Java programmers that focus on developing client applications, adding security to Web applications and Enterprise JavaBeans (EJBs), and working with other engineering, quality assurance (QA), and database teams to implement security features. Application Developers have in-depth/working knowledge of Java (including J2EE components such as servlets/JSPs and JSEE) and Java security.

- Server Administrators—Administrators work closely with Application Architects to design a security scheme for the server and the applications running on the server, to identify potential security risks, and to propose configurations that prevent security problems. Related responsibilities may include maintaining critical production systems, configuring and managing security realms, implementing authentication and authorization schemes for server and application resources, upgrading security features, and maintaining security provider databases. Server Administrators have in-depth knowledge of the Java security architecture, including Web application and EJB security, Public Key security, and SSL.

- Application Administrators—Administrators who work with Server Administrators to implement and maintain security configurations and authentication and authorization schemes, and to set up and maintain access to deployed application resources in defined security realms. Application Administrators have general knowledge of security concepts and the Java Security architecture. They understand Java, XML, deployment descriptors, and can identify security events in server and audit logs.

e-docs Web Site

BEA® product documentation is available on the BEA corporate Web site. From the BEA Home page, click on Product Documentation.

How to Print the Document

You can print a copy of this document from a Web browser, one file at a time, by using the File—>Print option on your Web browser.

A PDF version of this document is available on the WebLogic Server documentation Home page on the e-docs Web site (and also on the documentation CD). You can open the PDF in Adobe Acrobat Reader and print the entire document (or a portion of it) in book format. To access the PDFs, open the WebLogic Server documentation Home page, click Download Documentation, and select the document you want to print.

Adobe Acrobat Reader is available at no charge from the Adobe Web site at http://www.adobe.com.

Related Information

The following BEA WebLogic Server documents contain information that is relevant to the WebLogic Security Service:

- *Managing WebLogic Security*—This document explains how to configure security for WebLogic Server and how to use Compatibility security.
- *Developing Security Providers for WebLogic Server*—This document provides security vendors and application developers with the information needed to develop custom security providers that can be used with WebLogic Server.
- *Securing a Production Environment—*This document highlights essential security measures for you to consider before you deploy WebLogic Server into a production environment.
- *Securing WebLogic Resources*—This document introduces the various types of WebLogic resources, and provides information that allows you to secure these resources using WebLogic Server. The current version of this document primarily focuses on securing URL (Web) and Enterprise JavaBean (EJB) resources.
- *WebLogic Server 8.1 Upgrade Guide*—This document provides procedures and other information you need to upgrade 6.x and earlier versions of WebLogic Server to WebLogic Server 8.1. It also provides information about moving applications from a 6.x or earlier

version of WebLogic Server to 8.1. For specific information on upgrading WebLogic Server security, see *Security* in the *WebLogic Server 8.1 Upgrade Guide*.

- *Security FAQ*—This document gives answers to frequently asked questions about WebLogic Server security.
- *Javadocs for WebLogic Classes*—This document provides reference documentation for the WebLogic security packages that are provided with and supported by this release of WebLogic Server.

Contact Us!

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

In your e-mail message, please indicate the software name and version you are using, as well as the title and document date of your documentation.

If you have any questions about this version of BEA WebLogic Server, or if you have problems installing and running BEA WebLogic Server, contact BEA Customer Support through BEA WebSupport at http://www.bea.com. You can also contact Customer Support by using the contact information provided on the Customer Support Card, which is included in the product package.

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

- Your name, e-mail address, phone number, and fax number
- Your company name and company address
- Your machine type and authorization codes
- The name and version of the product you are using
- A description of the problem and the content of pertinent error messages

Documentation Conventions

The following documentation conventions are used throughout this document.

Convention	Item
Ctrl+Tab	Keys you press simultaneously.
italics	Emphasis or book titles.

`monospace text`	Code samples, commands and their options, data structures and their members, data types, directories, and file names and their extensions. Monospace text also indicates text that you must enter from the keyboard.
	Examples:
	`#import java.util.Enumeration;`
	`chmod u+w *`
	`\tux\data\ap`
	`.java`
	`config.xml`
	`float`

`monospace boldface text`	Identifies significant words in code. *Example*: `void` **`commit`** `( )`
`monospace italic text`	Variables in code. *Example*: `String` *`expr`*
UPPERCASE TEXT	Indicates device names, environment variables, and logical operators. *Example*s: LPT1 BEA_HOME OR
`{ }`	A set of choices in a syntax line.
`[ ]`	Optional items in a syntax line. The brackets themselves should never be typed. *Example*: `java [options] weblogic.Server [args...]`
|	Separates mutually exclusive choices in a syntax line.

Convention	Item
`...`	Indicates one of the following in a command line: • An argument can be repeated several times in a command line • The statement omits additional optional arguments • You can enter additional parameters, values, or other information The ellipsis itself should never be typed. *Example*: `java [options] weblogic.Server [args...]`
. . .	Indicates the omission of items from a code example or from a syntax line. The vertical ellipsis itself should never be typed.

Overview of the WebLogic Security Service

While other security documents in the BEA WebLogic Server™ documentation set guide users through specific tasks—such as programming WebLogic® security, developing a custom security provider, or managing the WebLogic Security Service—this *Introduction* is intended for all users of the WebLogic Security Service. Thus, this document is the starting point for understanding the WebLogic Security Service.

Note: The WebLogic® Security Service involves many unique terms. Before reading this manual, familiarize yourself with the terms in Chapter 5, "Terminology."

The following sections introduce the WebLogic Security Service and its features:

Audience for This Guide

This document is intended for the following audiences:

- Application Architects—Architects who, in addition to setting security goals and designing the overall security architecture for their organizations, evaluate WebLogic Server security features and determine how to best implement them. Application Architects have in-depth knowledge of Java programming, Java security, and network security, as well as knowledge of security systems and leading-edge, security technologies and tools.

- Security Developers—Developers who focus on defining the system architecture and infrastructure for security products that integrate into WebLogic Server and on developing custom security providers for use with WebLogic Server. They work with Application Architects to ensure that the security architecture is implemented according to design and that no security holes are introduced, and work with Server Administrators to ensure that security is properly configured. Security Developers have a solid understanding of security concepts, including authentication, authorization, auditing (AAA), in-depth knowledge of Java (including Java Management eXtensions (JMX), and working knowledge of WebLogic Server and security provider functionality.
- Application Developers—Developers who are Java programmers that focus on developing client applications, adding security to Web applications and Enterprise JavaBeans (EJBs), and working with other engineering, quality assurance (QA), and database teams to implement security features. Application Developers have in-depth/working knowledge of Java (including J2EE components such as servlets/JSPs and JSEE) and Java security.
- Server Administrators—Administrators work closely with Application Architects to design a security scheme for the server and the applications running on the server, to identify potential security risks, and to propose configurations that prevent security problems. Related responsibilities may include maintaining critical production systems, configuring and managing security realms, implementing authentication and authorization schemes for server and application resources, upgrading security features, and maintaining security provider databases. Server Administrators have in-depth knowledge of the Java security architecture, including Web application and EJB security, Public Key security, and SSL.
- Application Administrators—Administrators who work with Server Administrators to implement and maintain security configurations and authentication and authorization schemes, and to set up and maintain access to deployed application resources in defined security realms. Application Administrators have general knowledge of security concepts and the Java Security architecture. They understand Java, XML, deployment descriptors, and can identify security events in server and audit logs.

Introduction to the WebLogic Security Service

Deploying, managing, and maintaining security is a huge challenge for an information technology (IT) organization that is providing new and expanded services to customers using the Web. To serve a worldwide network of Web-based users, an IT organization must address the fundamental issues of maintaining the confidentiality, integrity and availability of the system and its data. Challenges to security involve every component of the system, from the network itself to the individual client machines. Security across the infrastructure is a complex business that requires vigilance as well as established and well-communicated security policies and procedures.

Beginning with release 7.0, WebLogic Server includes a completely redesigned security architecture that provides a unique and secure foundation for applications that are available via the Web. By taking advantage of the new security features in WebLogic Server, enterprises benefit from a comprehensive, flexible security infrastructure designed to address the security challenges of making applications available on the Web. WebLogic security can be used standalone to secure WebLogic Server applications or as part of an enterprise-wide, security management system that represents a best-in-breed, security management solution.

Features of the WebLogic Security Service

The open, flexible security architecture of WebLogic Server delivers advantages to all levels of users and introduces an advanced security design for application servers. Companies now have a unique application server security solution that, together with clear and well-documented security policies and procedures, can assure the confidentiality, integrity and availability of the server and its data.

The key features of the new WebLogic Security Service include:

- A comprehensive and standards-based design.
- End-to-end security for WebLogic Server-hosted applications, from the mainframe to the Web browser.
- Legacy security schemes that integrate with WebLogic Server security, allowing companies to leverage existing investments.
- Security tools that are integrated into a flexible, unified system to ease security management across the enterprise.
- Easy customization of application security to business requirements through mapping of company business rules to security policies.
- Easy updates to security policies.
- Easy adaptability for customized security solutions.
- A modularized architecture, so that security infrastructures can change over time to meet the requirements of a particular company.

- Support for configuring multiple security providers, as part of a transition scheme or upgrade path.
- A separation between security details and application infrastructure, making security easier to deploy, manage, maintain, and modify as requirements change.
- Default, WebLogic security providers that provide you with a working security scheme out of the box.
- Customization using WebLogic custom security providers
- Unified management of security rules, security policies, and security providers through the WebLogic Server Administration Console.
- Support for standard J2EE security technologies such as the Java Authentication and Authorization Service (JAAS), Java Secure Sockets Extensions (JSSE), and Java Cryptography Extensions (JCE).

Balancing Ease of Use and Customizability

The components and services of the WebLogic Security Service seek to strike a balance between ease of use, manageability (for end users and administrators), and customizability (for application developers and security developers). The following paragraphs highlight some examples:

Easy to use: For the end user, the secure WebLogic Server environment requires only a single sign-on for user authentication (ascertaining the user's identity). Users do not have to re-authenticate within the boundaries of the WebLogic Server domain that contains application resources. Single sign-on allows users to log on to the domain once per session rather than requiring them to log on to each resource or application separately.

For the developer and the administrator, WebLogic Server provides a new Domain Configuration Wizard to help with the creation of new domains with an administration server, managed servers, and optionally, a cluster, or with extending existing domains by adding individual severs. The Domain Configuration Wizard also automatically generates a `config.xml` file and start scripts for the server(s) you choose to add to the new domain.

Manageable: Administrators who configure and deploy applications in the WebLogic Server environment can use the WebLogic security providers included with the product. These default providers support all required security functions, out of the box. An administrator can store security data in the WebLogic Server-supplied, security store (an embedded, special-purpose, LDAP directory server). To simplify the configuration and management of security in WebLogic Server, a robust, default security configuration is provided.

Customizable: For application developers, WebLogic Server supports the WebLogic security API and J2EE security standards such as Java Authentication and Authorization (JAAS) and Java Secure Sockets Extensions (JSSE). Using these APIs and standards, you can create a fine-grained and customized security environment for applications that connect to WebLogic Server.

For security developers, the WebLogic Server Security Service Provider Interfaces (SSPIs) support the development of custom security providers for the WebLogic Server environment.

What Changed in WebLogic Security

Many security features have changed with respect to the security offered in WebLogic Server version 6.x.

Table 1-1 summarizes the differences.

Table 1-1 Changes in WebLogic Security Features

WebLogic Server Version 6.x	WebLogic Server 8.1
Security APIs	Many of the existing security APIs are deprecated in this release. BEA encourages you to use the corresponding J2EE standard interfaces to implement similar functionality in your application. For a complete list of deprecated APIs, see “Security APIs” in *Programming WebLogic Security*.
JAAS authentication	JAAS authentication has been enhanced to provide LoginModules for IIOP and T3 clients.
Auditing	You no longer have to create an implementation of the `weblogic.security.Audit` interface to add auditing to your WebLogic Server deployment. The WebLogic Auditing provider included with the product allows you to customize the data you want to record.
Defining security requirements in the `weblogic.xml`, `weblogic-ejb-jar.xml`, and `weblogic-ra.xml` files.	The functionality is enhanced so that security requirements can also be specified through the WebLogic Server Administration Console.
System password	There is no specific system account in this release of WebLogic Server.

Table 1-1 Changes in WebLogic Security Features

WebLogic Server Version 6.x	WebLogic Server 8.1
Access Control Lists (ACLs)	The ACLs used in releases prior to WebLogic Server 7.0 are deprecated in this release. ACLs are replaced by security policies in WebLogic 7.0.
Users and Groups	Users and groups are still used; however, instead of assigning ACLs to a resource, you now create a security policy that grants users, groups, or security roles access to a WebLogic resource.
6.x Security Realms (File realm, Caching realm, LDAP, Windows NT, UNIX, and RDBMS security realms)	The security realms used in releases prior to WebLogic Server 7.0 are deprecated in this release. The WebLogic Authentication and Authorization providers provide the same functionality offered by the File realm, the Caching realm, and the LDAP security realms. The Realm Adapter providers are available to allow you to continue to use the existing Windows NT, UNIX, and RDBMS security realms as you migrate to the new Authentication/Authorization scheme.
This feature was not available in releases prior to WebLogic Server 7.0.	Support for multiple security providers.
SSL	The SSL support in WebLogic Server has been updated to support the JSSE standard and the Transport Layer Security (TLS) v1 protocol.
This feature was not available in releases prior to WebLogic Server 7.0.	Support for J2EE Java KeyStores (JKS).

Security Fundamentals

The following sections describe security fundamentals as they relate to security in WebLogic Server:

Auditing

Auditing is the process whereby information about operating requests and the outcome of those requests are collected, stored, and distributed for the purposes of non-repudiation. In other words, auditing provides an electronic trail of computer activity. In the WebLogic Server security architecture, an Auditing provider is used to provide auditing services.

If configured, the WebLogic Security Framework will call through to an Auditing provider before and after security operations (such as authentication or authorization) have been performed, 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/or severity levels. The records containing the audit information may be written to output repositories such as an LDAP server, database, and a simple file.

Authentication

Authentication is the mechanism by which callers prove that they are acting on behalf of specific users or systems. Authentication answers the question, “Who are you?” using credentials such as username/password combinations.

In WebLogic Server, Authentication providers are used to prove the identity of users or system processes. Authentication providers also remember, transport, and make identity information available to various components of a system (via subjects) when needed. During the authentication process, a Principal Validation provider provides additional security protections for the principals (users and groups) contained within the subject by signing and verifying the authenticity of those principals.

The following sections describe authentication concepts and functionality.

- “Subjects and Principals” on page 2-2
- “Java Authentication and Authorization Service (JAAS)” on page 2-3
- “CallbackHandlers” on page 2-5
- “Mutual Authentication” on page 2-5
- “Identity Assertion Providers and LoginModules” on page 2-6
- “Identity Assertion and Tokens” on page 2-6
- “Types of Authentication” on page 2-7

Subjects and Principals

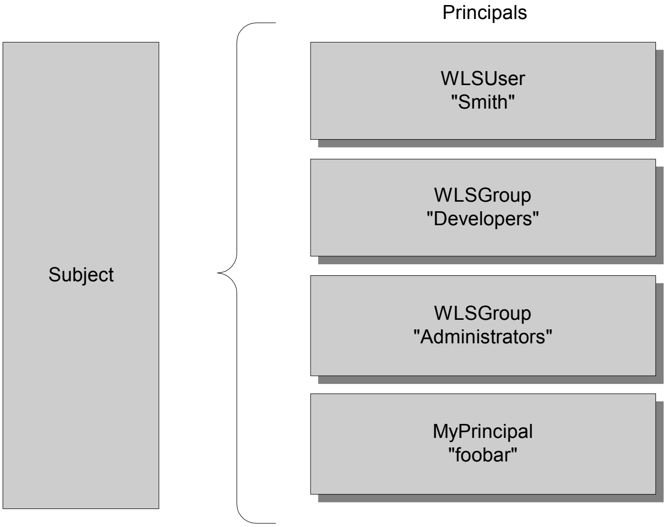
Subjects and principals are closely related.

A principal is an identity assigned to a user or group as a result of authentication. Both users and groups can be used as principals by application servers such as WebLogic Server. The Java Authentication and Authorization Service (JAAS) requires that subjects be used as containers for authentication information, including principals.

Figure 2-1 illustrates the relationships among users, groups, principals, and subjects.

Note: In this release of WebLogic Server, subjects replace WebLogic Server 6.x users.

Figure 2-1 Relationships Among Users, Groups, Principals and Subjects



As part of a successful authentication, principals are signed and stored in a subject for future use. A Principal Validation provider signs principals, and an Authentication provider's LoginModule actually stores the principals in the subject. Later, when a caller attempts to access a principal stored within a subject, a Principal Validation provider verifies that the principal has not been altered since it was signed, and the principal is returned to the caller (assuming all other security conditions are met).

Any principal that is going to represent a WebLogic Server user or group needs to implement the `WLSUser` and `WLSGroup` interfaces, which are available in the `weblogic.security.spi` package.

Java Authentication and Authorization Service (JAAS)

Whether the client is an application, applet, Enterprise JavaBean (EJB), or servlet that requires authentication, WebLogic Server uses the Java Authentication and Authorization Service (JAAS) classes to reliably and securely authenticate to the client. JAAS implements a Java version of the Pluggable Authentication Module (PAM) framework, which permits applications to remain independent from underlying authentication technologies. Therefore, the PAM framework allows the use of new or updated authentication technologies without requiring modifications to your application.

WebLogic Server uses JAAS for remote fat-client authentication, and internally for authentication. Therefore, only developers of custom Authentication providers and developers of remote fat client applications need to be involved with JAAS directly. Users of thin clients or developers of within-container fat client applications (for example, those calling an Enterprise JavaBean (EJB) from a servlet) do not require the direct use or knowledge of JAAS.

JAAS LoginModules

LoginModules are the work-horses of authentication: all LoginModules are responsible for authenticating users within the security realm and for populating a subject with the necessary principals (users/groups). LoginModules that are *not* used for perimeter authentication also verify the proof material submitted (for example, a user's password).

If there are multiple Authentication providers configured in a security realm, each of the Authentication providers' LoginModules will store principals within the same subject. Therefore, if a principal that represents a WebLogic Server user (that is, an implementation of the `WLSUser` interface) named "Joe" is added to the subject by one Authentication provider's LoginModule, any other Authentication provider in the security realm should be referring to the same person when they encounter "Joe". In other words, the other Authentication providers' LoginModules should not attempt to add another principal to the subject that represents a WebLogic Server user (for example, named "Joseph") to refer to the same person. However, it is acceptable for another Authentication provider's LoginModule to add a principal of a type other than `WLSUser` with the name "Joseph".

JAAS Control Flags

If a security realm has multiple Authentication providers configured, the Control Flag attribute on the Authenticator provider determines the ordered execution of the Authentication providers. The values for the Control Flag attribute are as follows:

- REQUIRED—This LoginModule must succeed. Even if it fails, authentication proceeds down the list of LoginModules for the configured Authentication providers. This setting is the default.
- REQUISITE—This LoginModule must succeed. If other Authentication providers are configured and this LoginModule succeeds, authentication proceeds down the list of LoginModules. Otherwise, return control to the application.
- SUFFICIENT—This LoginModule needs not succeed. If it does succeed, return control to the application. If it fails and other Authentication providers are configured, authentication proceeds down the LoginModule list.

- OPTIONAL—The user is allowed to pass or fail the authentication test of this Authentication providers. However, if all Authentication providers configured in a security realm have the JAAS Control Flag set to OPTIONAL, the user must pass the authentication test of one of the configured providers.

CallbackHandlers

A `CallbackHandler` is a highly-flexible JAAS standard that allows a variable number of arguments to be passed as complex objects to a method. There are three types of `CallbackHandlers`: `NameCallback`, `PasswordCallback`, and `TextInputCallback`, all of which are part of the `javax.security.auth.callback` package. The `NameCallback` and `PasswordCallback` return the username and password, respectively. `TextInputCallback` can be used to access the data users enter into any additional fields on a login form (that is, fields other than those for obtaining the username and password). When used, there should be one `TextInputCallback` per additional form field, and the prompt string of each `TextInputCallback` must match the field name in the form. WebLogic Server only uses the `TextInputCallback` for form-based Web application login.

An application implements a `CallbackHandler` and passes it to underlying security services so that they may interact with the application to retrieve specific authentication data, such as usernames and passwords, or to display certain information, such as error and warning messages.

`CallbackHandlers` are implemented in an application-dependent fashion. For example, implementations for an application with a graphical user interface (GUI) may pop up windows to prompt for requested information or to display error messages. An implementation may also choose to obtain requested information from an alternate source without asking the user.

Underlying security services make requests for different types of information by passing individual `Callbacks` to the `CallbackHandler`. The `CallbackHandler` implementation decides how to retrieve and display information depending on the `Callbacks` passed to it. For example, if the underlying service needs a username and password to authenticate a user, it uses a `NameCallback` and `PasswordCallback`. The `CallbackHandler` can then choose to prompt for a username and password serially, or to prompt for both in a single window.

Mutual Authentication

With mutual authentication, both the client and the server are required to authenticate themselves to each other. This can be done by means of certificates or other forms of proof material. WebLogic Server supports two-way SSL authentication, which is a form of mutual authentication. However, by strict definition, mutual authentication takes place at higher layers in the protocol stack then does SSL authentication. For more information, see "One-way/Two-way SSL Authentication" on page 2-16.

Identity Assertion Providers and LoginModules

When used with a LoginModule, Identity Assertion providers support single sign-on. For example, an 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 LoginModule that an Identity Assertion provider uses can be:

- Part of a custom Authentication provider you develop.
- Part of the WebLogic Authentication provider that BEA developed and packaged with WebLogic Server.
- Part of a third-party security vendor's Authentication provider.

Unlike in a simple authentication situation, the LoginModules that Identity Assertion providers use *do not* verify proof material such as usernames and passwords; they simply verify that the user exists.

Identity Assertion and Tokens

Identity Assertion providers support user name mappers, which map a valid token to a WebLogic Server user. You develop Identity Assertion providers to support the specific types of tokens that you will be using to assert the identities of users or system processes. You can develop an Identity Assertion provider to support multiple token types, but the WebLogic Server 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 security realm with *the ability* to validate the same token type, only one Identity Assertion provider can actually perform this validation.

Note: To use the WebLogic Identity Assertion provider for X.501 and X.509 certificates, you have the option of using the default user name mapper that is supplied with the WebLogic Server product (`weblogic.security.providers.authentication.DefaultUserNameMapperImpl`) or providing you own implementation of the `weblogic.security.providers.authentication.UserNameMapper` interface. For more mformation, see Do I Need to Develop a Custom Identity Assertion Provider? in *Developing Security Providers for WebLogic Server*.

Types of Authentication

WebLogic Server users must be authenticated whenever they request access to a protected WebLogic resource. For this reason, each user is required to provide a credential (for example, a password) to WebLogic Server. The following types of authentication are supported by the WebLogic Authentication provider that is included in the WebLogic Server distribution:

- "Username/Password Authentication" on page 2-7
- "Certificate Authentication" on page 2-7
- "Perimeter Authentication" on page 2-8

WebLogic Server can use the WebLogic Authentication provider that is provided as part of the WebLogic Server product or custom security providers to perform the different types of authentication. For information on the WebLogic Authentication provider and how to configure authentication, see "The Credential Mapping Process" on page 4-9 and the following sections in *Managing WebLogic Security*:

- *"*Configuring Security Providers*"*
- "Configuring SSL"

Username/Password Authentication

In username/password authentication, a user ID and password are requested from the user and sent to WebLogic Server. WebLogic Server checks the information and if it is trustworthy, grants access to the protected WebLogic resource.

Secure Sockets Layer (SSL), or Hyper-Text Transfer Protocol (HTTPS), can be used to provide an additional level of security to username/password authentication. Because SSL encrypts the data transferred between the client and WebLogic Server, the user ID and password of the user do not flow in the clear. Therefore, WebLogic Server can authenticate the user without compromising the confidentiality of the user's ID and password.

Certificate Authentication

When an SSL or HTTPS client request is initiated, WebLogic Server responds by presenting its digital certificate to the client. The client then verifies the digital certificate and an SSL connection is established. The digital certificate is issued by an entity (a trusted certificate authority), which validates the identity of WebLogic Server.

You can also use two-way SSL authentication, a form of mutual authentication. With two-way SSL authentication, both the client and server must present a certificate before the connection thread is enabled between the two. See "One-way/Two-way SSL Authentication" on page 2-16.

Note: Two-way SSL authentication is supported by the WebLogic Authentication provider that is provided as part of the WebLogic Server product.

Perimeter Authentication

Perimeter authentication is the process of authenticating the identity of a remote user outside of the application server domain.

The following sections describe perimeter authentication:

- "How is Perimeter Authentication Accomplished?" on page 2-8
- "How Does WebLogic Server Support Perimeter Authentication?" on page 2-8

How is Perimeter Authentication Accomplished?

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 (such as a password, a credit card number, Personal Identification Number, or some other form of personal identification information), which is used to perform the verification.

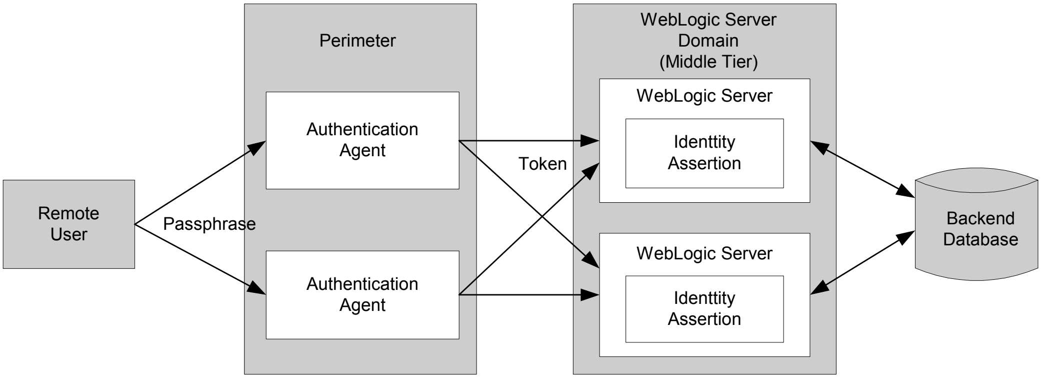
The authentication agent, the entity that actually vouches for the identity, can take many forms, such as a Virtual Private Network (VPN), firewall, an enterprise authentication service, or some other form of global identity service. Each of these forms of authentication agents has a common characteristic: they all perform an authentication process that results in an artifact or token that must be presented to determine information about the authenticated user at a later time. Currently, the format of the token varies from vendor to vendor, but there are efforts to define a standard token format using XML. In addition, there is a current standard for Attribute Certificates, which is based on the X.509 standard for digital certificates. But even after all of this, if the applications and the infrastructure on which they are built are not designed to support this concept, enterprises are still forced to require that their remote users re-authenticate to the applications within the network.

How Does WebLogic Server Support Perimeter Authentication?

WebLogic Server is designed to extend the single sign-on concept all the way to the perimeter through support for identity assertion (see Figure 2-2). Provided as a critical piece of the WebLogic Security Framework, the concept of identity assertion allows WebLogic Server to use the authentication mechanism provided by perimeter authentication schemes such as

Checkpoint's OPSEC, the emerging Security Assertion Markup Language (SAML), or enhancements to protocols such as Common Secure Interoperability (CSI) v2 to achieve this functionality.

Figure 2-2 Perimeter Authentication



Support for perimeter authentication requires the use of an Identity Assertion provider that is designed to support one or more token formats. Multiple and different Identity Assertion providers can be registered for use. The tokens are transmitted as part of any normal business request, using the mechanism provided by each of the various protocols supported by WebLogic Server. Once a request is received with WebLogic Server, 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 WebLogic Security Framework that results in the appropriate Identity Assertion provider being called 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.

Single Sign-On with Microsoft Clients

Single sign-on (SSO) is the ability for a user to sign on to an application only once and gain access to many different application components, even though these components may have their own authentication schemes. SSO enables users to login securely to all their applications, web sites and mainframe sessions with just one identity. WebLogic Server provides single sign-on (SSO) with Microsoft clients. This type of SSO uses HTTP-based authentication with Microsoft clients that have authenticated in the Windows Active Directory environment. The Windows Active

Directory environment uses Kerberos as its security protocol. Kerberos provides network authentication of heterogeneous realms. This means a user logged into a Windows domain can access a Web application running on an application server and use their Windows Active Directory credentials to authenticate to the server. The application server can run on any platform that supports Kerberos.

When a Web server receives a request from a browser it can request that the browser use the Kerberos protocol to authenticate itself. This protocol performs authentication via HTTP, and allows the browser (in most cases, Internet Explorer) to pass a delegated credential to allow a web application to log into subsequent Kerberos-based services on the user's behalf.

When an HTTP server wishes to login a Microsoft client, it returns a `401 Unauthorized` response to the HTTP request with the `WWW-Authorization:Negotiate` header. The browser then contacts the Key Distribution Center (KDC)/Ticket Granting Service (TGS) to obtain a service ticket. It chooses a special Service Principal Name for the ticket request. The returned ticket is then wrapped in a SPNEGO token which is encoded and sent back to the server using an HTTP request. The token is unwrapped and the ticket is authenticated. Once authenticated, the page corresponding to the requested URL is returned.

For information about how SSO with Microsoft clients is implemented in WebLogic Server, see "SSO with Microsoft Clients Process" on page 4-9.

Authorization

Authorization is the process whereby the interactions between users and WebLogic resources are controlled, based on user identity or other information. In other words, authorization answers the question, "What can you access?" In WebLogic Server, an Authorization provider is used to limit the interactions between users and WebLogic resources to ensure integrity, confidentiality, and availability.

The following sections describe authorization concepts and functionality:

- "WebLogic Resources" on page 2-11
- "Security Policies" on page 2-11
- "ContextHandlers" on page 2-12
- "Access Decisions" on page 2-13
- "Adjudication" on page 2-13

WebLogic Resources

A **WebLogic resource** is a structured object used to represent an underlying WebLogic Server entity, which can be protected from unauthorized access using security roles and security policies.

WebLogic resources are hierarchical. Therefore, the level at which you define these security roles and security policies is up to you. For example, you can define security roles and security policies on: entire enterprise applications (EARs); an Enterprise JavaBean (EJB) JAR containing multiple EJBs; a particular Enterprise JavaBean (EJB) within that JAR; or a single method within that EJB.

WebLogic resource implementations are available for:

- Administrative resources
- Application resources
- Common Object Model (COM) resources
- Enterprise Information System (EIS) resources
- Enterprise JavaBean (EJB) resources
- Java Database Connectivity (JDBC) resources
- Java Messaging Service (JMS) resources
- Java Naming and Directory Interface (JNDI) resources
- Server resources
- Universal Resource Locator (URL) resources
- Web Service resources

Note: Each of these WebLogic resource implementations is explained in detail in the Javadocs for WebLogic Classes. For more information, see “Types of WebLogic Resources” in *Securing WebLogic Resources*.

Security Policies

In releases prior to WebLogic Server 7.0, access control lists (ACLs) were used to protect WebLogic resources. In this release of WebLogic Server, security policies replace ACLs and answer the question “Who has access to a WebLogic resource?” A security policy is created when you define an association between a WebLogic resource and one or more users, groups, or security roles. You can optionally define a time constraint for a security policy. A WebLogic resource has no protection until you assign it a security policy.

You assign security policies to any of the defined WebLogic resources (for example, an EJB resource or a JNDI resource) or to attributes or operations of a particular instance of a WebLogic resource (an EJB method or a servlet within a Web application). If you assign a security policy to a type of WebLogic resource, all new instances of that resource inherit that security policy. Security policies assigned to individual resources or attributes override security policies assigned to a type of WebLogic resource. For a list of the defined WebLogic resources, see "WebLogic Resources" on page 2-11.

Security policies are stored in an Authorization provider's database. By default, the WebLogic Authorization provider is configured and security policies are stored in the embedded LDAP server.

To use a user or group to create a security policy, the user or group must be defined in the security provider database for the Authentication provider that is configured in the default security realm. To use a security role to create a security policy, the security role must be defined in the security provider database for the Role Mapping provider that is configured in the default security realm. By default, the WebLogic Authentication and Role Mapping providers are configured in the database in the embedded LDAP server.

By default, security policies are defined in WebLogic Server for the WebLogic resources. These security policies are based on security roles and default global groups. You also have the option of basing a security policy on a user. BEA recommends basing security policies on security roles rather than users or groups. Basing security policies on security roles allows you to manage access based on a security role that a user or group is granted, which is a more efficient method of management. For a listing of the default security policies for the WebLogic resources, see "Default Security Policies" in *Securing WebLogic Resources*.

ContextHandlers

A **ContextHandler** is a high-performing WebLogic 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 WebLogic resource container to pass additional information to a WebLogic 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 WebLogic resource container and the security provider.) Each name/value pair in a ContextHandler is known as a **context element**, and is represented by a `ContextElement` object.

Currently, three types of WebLogic resource containers pass ContextHandlers to the WebLogic 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 `AuditContext` interface (used when a security provider is implemented to post audit events) may also examine the values of context elements.

For more information about the values of particular context elements, see “ContextHandlers and WebLogic Resources” in *Developing Security Providers for WebLogic Server*.

Access Decisions

Like LoginModules for Authentication providers, an Access Decision is the component of an Authorization provider that actually answers the “is access allowed?” question. Specifically, an Access Decision is asked whether a subject has permission to perform a given operation on a WebLogic resource, with specific parameters in an application. Given this information, the Access Decision responds with a result of `PERMIT`, `DENY`, or `ABSTAIN`.

Adjudication

Adjudication involves resolving any authorization conflicts that may occur when more than one Authorization provider is configured in a security realm, by weighing the result of each Authorization provider’s Access Decision. In WebLogic Server, an Adjudication provider is used to tally the results that multiple Access Decisions return, and determines the final `PERMIT` or `DENY` decision. An Adjudication provider may also specify what should be done when an answer of `ABSTAIN` is returned from a single Authorization provider’s Access Decision.

Secure Sockets Layer (SSL)

SSL enables secure communication between applications connected through the Web. For a discussion of the components of SSL communication and why each component is necessary, see *How SSL Works*, published by the Netscape Communications Corporation.

WebLogic Server fully supports SSL communication. By default, WebLogic Server is configured for one-way SSL authentication. Using the Administration Console, you can configure WebLogic Server for two-way SSL authentication.

To use SSL with WebLogic Server, you need a private key, a digital certificate containing the matching public key, and a certificate signed by at least one trusted certificate authority (CA) that can verify the data embedded in the digital certificate. If the certificate authority that signed the digital certificate is not well-known, you may also need to install their root trusted CA's certificate in your WebLogic Server. Trusted CA ceritificates are signed by authorities that are well-know.

To acquire a digital certificate for your server, you generate a public key, private key, and a Certificate Signature Request (CSR), which contains your public key. You send the CSR request to a certificate authority and follow their procedures for obtaining a signed digital certificate.

Once you have your private keys, digital certificates, and any additional trusted CA certificates that you may need, you need to store them so that WebLogic Server can use them to verify identity. In this release of WebLogic Server, you should store your private keys and certificates in keystores. For purposes of backwards compatibility, you may also store your private keys and certificates in files. For more information about private key, public key, and certificate requirements and procedures, see "Configuring SSL*"* and *Managing WebLogic Security*.

To use SSL when connecting to a WebLogic server application with your browser, you simply specify HTTPS and the secure port (port number 7002) in the URL. For example:

```
https://localhost:7002/examplesWebApp/SnoopServlet.jsp
```

Where *localhost* is the name of the system hosting the Web application.

The following topics are discussed in this section:

- "SSL Features" on page 2-15
- "SSL Tunneling" on page 2-16
- "One-way/Two-way SSL Authentication" on page 2-16
- "Domestic SSL and Exportable SSL" on page 2-17
- "Digital Certificates" on page 2-18
- "Certificate Authorities" on page 2-19
- "Host Name Verification" on page 2-20
- "Trust Managers" on page 2-20
- "Asymmetric Key Algorithms" on page 2-20
- "Symmetric Key Algorithms" on page 2-21

- "Message Digest Algorithms" on page 2-22
- "Cipher Suites" on page 2-22

SSL Features

WebLogic Server provides a pure-Java implementation of SSL. Generally, SSL provides the following:

- A mechanism that the communicating applications can use to authenticate each other's identity.
- Encryption of the data exchanged by the applications.

When SSL is used, the target (the server) always authenticates itself to the initiator (the client). Optionally, if the target requests it, the initiator can authenticate itself to the target. Encryption makes data transmitted over the network intelligible only to the intended recipient. An SSL connection begins with a handshake during which the applications exchange digital certificates, agree on the encryption algorithms to be used, and generate the encryption keys to be used for the remainder of the session.

SSL provides the following security features:

- Server authentication—WebLogic Server uses its digital certificate, issued by a trusted certificate authority, to authenticate to clients. SSL minimally requires the server to authenticate to the client using its digital certificate. If the client is not required to present a digital certificate, the connection type is called one-way SSL authentication.
- Client Identity Verification—Optionally, clients might be required to present their digital certificates to WebLogic Server. WebLogic Server then verifies that the digital certificate was issued by a trusted certificate authority and establishes the SSL connection. An SSL connection is not established if the digital certificate is not presented and verified. This type of connection is called two-way SSL authentication, a form of mutual authentication.
- Confidentiality—All client requests and server responses are encrypted to maintain the confidentiality of data exchanged over the network.
- Data Integrity—Data that flows between a client and WebLogic Server is protected from tampering by a third-party validation of user identities.

If you are using a Web browser to communicate with WebLogic Server, you can use the Hyper-Text Transfer Protocol with SSL (HTTPS) to secure network communications.

SSL Tunneling

SSL is tunneled over an IP-based protocol. Tunneling means that each SSL record is encapsulated and packaged with the headers needed to send the record over another protocol. SSL can be used by Web browsers and Java clients as follows:

- SSL communications between Web browsers and WebLogic Server are encapsulated in HTTPS packets for transport. For example:

  ```
  https://myserver.com/mypage.html
  ```

  WebLogic Server supports HTTPS with Web browsers that support SSL version 3. The Java Virtual Machine (JVM) in WebLogic Server does not currently support the HTTPS adapter. Consequently, WebLogic Server depends on the implementation of SSL in the Web browser.

- Java clients connecting to WebLogic Server with SSL tunnelled over BEA's multiplexed T3 protocol. For example:

  ```
  t3s://myserver.com:7002/mypage.html
  ```

  Java clients running in WebLogic Server can establish either T3S connections to other WebLogic Servers, or HTTPS connections to other servers that support SSL, such as Web servers or secure proxy servers.

One-way/Two-way SSL Authentication

WebLogic Server supports one-way and two-way SSL authentication.With one-way SSL authentication, the target (the server) is required to present a digital certificate to the initiator (the client) to prove its identity. The client performs two checks to validate the digital certificate:

1. The client checks that the digital certificate is on its list of trusted certificate authorities.
2. The client checks that the host name in the certificate matches the name of the server.

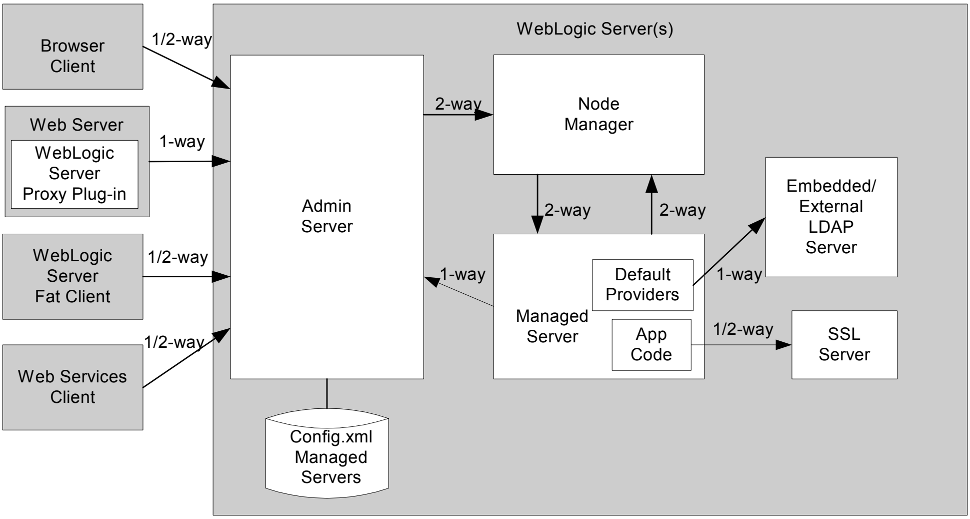
If both of the above checks return true, the SSL connection is established.

With two-way SSL authentication, both the client and server must present digital certificates before the SSL connection is enabled between the two. Thus, in this case, WebLogic Server not only authenticates itself to the client (which is the minimum requirement for certificate authentication), but it also requires authentication from the requesting client. Two-way SSL authentication is useful when you must restrict access to trusted clients only.

Figure 2-3 illustrates WebLogic Server SSL connections and shows which connections support one-way SSL, two-way SSL, or both. The Web browser client, Web Server, Fat client, Web

Services client, and SSL server connections can be configured for either one-way or two-way SSL. WebLogic Server determines whether an SSL connection is configured for one-way or two-way. Use the Administration Console to configure SSL.

Figure 2-3 How WebLogic Server Supports SSL Connections



Note: The SSL server shown in this figure can be any J2EE compliant server.

Domestic SSL and Exportable SSL

WebLogic Server is available with exportable- or domestic-strength SSL.

- Exportable SSL supports 512-bit certificates and 40- and 50-bit bulk data encryption.
- Domestic SSL also supports 768-bit, 1024-bit, and 2048-bit certificates, and 128-bit bulk data encryption.

The standard WebLogic Server distribution supports exportable-strength SSL only. The domestic version is available, by request only from your BEA sales representative.

Notes: If you request the domestic-strength version of WebLogic Server and you qualify to receive it, you will receive a domestic-strength WebLogic Server software license to use when you install the WebLogic Server distribution.

Because the United States Government relaxed restrictions on exporting encryption software early the year 2000, the domestic version of WebLogic Server can be used in most countries.

BEA recommends the domestic WebLogic Server distribution because it allows stronger encryption.

Note: If you generate a Certificate Signature Request (CSR), which is an electronically generated request for a certificate, using the exportable-strength WebLogic Server distribution, you cannot support domestic-strength SSL connections and you cannot authenticate clients that present domestic-strength certificates.

For more information, see "Configuring SSL" in *Managing WebLogic Security*.

Digital Certificates

Digital certificates are electronic documents used to verify the unique identities of principals and entities over networks such as the Internet. A digital certificate securely binds the identity of a user or entity, as verified by a trusted third party (known as a certificate authority), to a particular public key. The combination of the public key and the private key provides a unique identity to the owner of the digital certificate.

Digital certificates enable verification of the claim that a specific public key does in fact belong to a specific user or entity. A recipient of a digital certificate can use the public key in a digital certificate to verify that a digital signature was created with the corresponding private key. If such verification is successful, this chain of reasoning provides assurance that the corresponding private key is held by the subject named in the digital certificate, and that the digital signature was created by that subject.

A digital certificate typically includes a variety of information, such as the following:

- The name of the subject (holder, owner) and other information required to confirm the unique identity the subject, such as the URL of the Web server using the digital certificate, or an individual's e-mail address
- The subject's public key
- The name of the certificate authority that issued the digital certificate
- A serial number

- The validity period (or lifetime) of the digital certificate (defined by a start date and an end date)

The most widely accepted format for digital certificates is defined by the ITU-T X.509 international standard. Digital certificates can be read or written by any application complying with the X.509 standard. The public key infrastructure (PKI) in WebLogic Server recognizes digital certificates that comply with X.509 version 3, or X.509v3. BEA recommends obtaining digital certificates from a certificate authority such as Verisign or Entrust.

For more information, see "Configuring SSL" in *Managing WebLogic Security*.

Certificate Authorities

Digital certificates are issued by certificate authorities. Any trusted, third-party organization or company that is willing to vouch for the identities of those to whom it issues digital certificates and public keys can be a certificate authority. When a certificate authority creates a digital certificate, the certificate authority signs it with its private key, to ensure that any tampering will be detected. The certificate authority then returns the signed digital certificate to the requesting party.

The requesting party can verify the signature of the issuing certificate authority by using the public key of the certificate authority. The certificate authority makes its public key available by providing a certificate issued from a higher-level certificate authority attesting to the validity of the public key of the lower-level certificate authority. This scheme gives rise to hierarchies of certificate authorities. This hierarchy is terminated by a top-level, self-signed certificate known as the root certificate, because no other public key is needed to certify it. Root certificates are issued by trusted (root) Certificate Authorities.

If the recipient has a digital certificate containing the public key of the certificate authority signed by a superior certificate authority who the recipient already trusts, the recipient of an encrypted message can develop trust in the public key of a certificate authority recursively. In this sense, a digital certificate is a stepping stone in digital trust. Ultimately, it is necessary to trust only the public keys of a small number of top-level certificate authorities. Through a chain of certificates, trust in a large number of users' digital signatures can be established.

Thus, digital signatures establish the identities of communicating entities, but a digital signature can be trusted only to the extent that the public key for verifying it can be trusted.

For more information, see "Configuring SSL" in *Managing WebLogic Security*.

Host Name Verification

Host Name verification is the process of verifying that the name of the host to which an SSL connection is made is the intended or authorized party. Host name verification prevents man-in-the-middle attacks when a Web client (a Web browser, a WebLogic client, or a WebLogic Server acting as a client) requests an SSL connection to another application server.

By default, WebLogic Server, as a function of the SSL handshake, compares the common name in the SubjectDN of the SSL server's digital certificate with the host name of the SSL server used to initiate the SSL connection. If these names do not match, the SSL connection is dropped.

Trust Managers

When an SSL client connects to an SSL server, the SSL server presents its digital certificate chain to the client for authentication. That chain could contain an invalid digital certificate. The SSL specification says that the client should drop the SSL connection upon discovery of an invalid certificate. Web browsers, however, ask the user whether to ignore the invalid certificate and continue up the chain to determine if it is possible to authenticate the SSL server with any of the remaining certificates in the certificate chain. The Trust Manager eliminates this inconsistent practice by enabling you to control when to continue or discontinue an SSL connection. Using a Trust Manager you can perform custom checks before continuing an SSL connection. For example, you can use the Trust Manager to specify that only users from specific localities, such as towns, states, or countries, or users with other special attributes, can gain access via the SSL connection.

WebLogic Server provides the `weblogic.security.SSL.TrustManagerJSSE` interface, which enables you to override validation errors in a peer's digital certificate and continue the SSL handshake. You can also use this interface to discontinue an SSL handshake by performing additional validation on a server's digital certificate chain.

Note: This interface takes new style certificates and replaces the `weblogic.security.SSL.TrustManager` interface, which is being deprecated in this release of WebLogic Server.

Asymmetric Key Algorithms

Asymmetric key (also referred to as public key) algorithms are implemented through a pair of different, but mathematically related keys: a public key and a private key.

- The public key (which is distributed widely) is used for verifying a digital signature or transforming data into a seemingly unintelligible form.

- The private key (which is always kept secret) is used for creating a digital signature or returning the data to its original form.

The Public Key Infrastructure (PKI) in WebLogic Server also supports digital signature algorithms. Digital signature algorithms are simply public key algorithms used to generate digital signatures.

WebLogic Server supports the Rivest, Shamir, and Adelman (RSA) algorithm.

For information on storage of public and private keys, see "Keystore Providers" on page 3-13.

Symmetric Key Algorithms

With symmetric key algorithms, you use the same key to encrypt and decrypt a message. This common key encryption system uses a symmetric key algorithm to encrypt a message sent between two communicating entities. Symmetric key encryption operates at least 1000 times faster than public key cryptography.

A block cipher is a type of symmetric key algorithm that transforms a fixed-length block of plain text (unencrypted text) data into a block of cipher text (encrypted text) data of the same length. This transformation takes place in accordance with the value of a randomly generated session key. The fixed length is called the block size.

The PKI in WebLogic Server supports the following symmetric key algorithms:

- DES-CBC (Data Encryption Standard for Cipher Block Chaining)

  DES-CBC is a 64-bit block cipher run in Cipher Block Chaining (CBC) mode. It provides 56-bit keys. (8 parity bits are stripped from the full 64-bit key.)

- Two-key triple-DES (Data Encryption Standard)

  Two-key triple-DES is a 128-bit block cipher run in Encrypt-Decrypt-Encrypt (EDE) mode. Two-key triple-DES provides two 56-bit keys (in effect, a 112-bit key).

  For some time it has been common practice to protect and transport a key for DES encryption with triple-DES, which means that the input data (in this case the single-DES key) is encrypted, decrypted, and then encrypted again (an encrypt-decrypt-encrypt process). The same key is used for the two encryption operations.

- RC4 (Rivest's Cipher 4)

  RC4 is a variable key-size block cipher with a key size range of 40 to 128 bits. It is faster than DES and can be exported with a key size of 40 bits. A 56-bit key size is allowed for foreign subsidiaries and overseas offices of United States companies. In the United States, RC4 can be used with keys of virtually unlimited length, although the WebLogic Server PKI restricts the key length to 128 bits.

Note: WebLogic Server users cannot expand or modify this list of algorithms.

Message Digest Algorithms

WebLogic Server supports the MD5 and SHA (Secure Hash Algorithm) message digest algorithms. Both MD5 and SHA are well known, one-way hash algorithms. A one-way hash algorithm takes a message and converts it into a fixed string of digits, which is referred to as a message digest or hash value.

MD5 is a high-speed, 128-bit hash; it is intended for use with 32-bit machines. SHA offers more security by using a 160-bit hash, but is slower than MD5.

Cipher Suites

A cipher suite is an SSL encryption method that includes the key exchange algorithm, the symmetric encryption algorithm, and the secure hash algorithm. A cipher suite is used to protect the integrity of a communication. For example, the cipher suite called `RSA_WITH_RC4_128_MD5` uses RSA for key exchange, RC4 with a 128-bit key for bulk encryption, and MD5 for message digest.

WebLogic Server ships with the RSA Crypto-J 3.5 library. Check the validation lists at http://csrc.nist.gov/cryptval/ for FIPS140 for validation status.

WebLogic Server supports the RSA cipher suites described in Table 2-1.

Table 2-1 SSL Cipher Suites Supported by WebLogic Server

Cipher Suite	Symmetric Key Strength
`TLS_RSA_WITH_RC4_128_SHA`	128
`TLS_RSA_WITH_RC4_128_MD5`	128
`TLS_RSA_WITH_DES_CBC_SHA`	56
`TLS_RSA_EXPORT_WITH_RC4_40_MD5`	40
`TLS_RSA_EXPORT_WITH_DES40_CBC_SHA`	40
`TLS_RSA_WITH_3DES_EDE_CBC_SHA`	112
`TLS_RSA_WITH_NULL_SHA`	0

Table 2-1 SSL Cipher Suites Supported by WebLogic Server (Continued)

Cipher Suite	Symmetric Key Strength
TLS_RSA_WITH_NULL_MD5	0
TLS_RSA_EXPORT1024_WITH_DES_CBC_SHA	56
TLS_RSA_EXPORT124_WITH_RC4_56_SHA	56

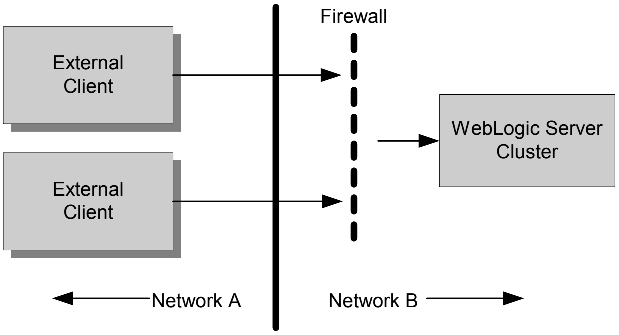
The license for WebLogic Server determines what strength (either domestic or export) of cipher suite is used to protect communications. If the cipher suite strength defined in the WebLogic Server `config.xml` file exceeds the strength specified by the license, the server uses the strength specified by the license. For example, if you have an export strength license but you define the use of an domestic strength cipher suite in the `config.xml` file, the server rejects the domestic strength cipher suite and uses the export strength cipher suite.

Firewalls

A firewall limits traffic between two networks. Firewalls can be a combination of software and hardware, including routers and dedicated gateway machines. They employ filters that allow or disallow traffic to pass based on the protocol, the service requested, routing information, and the origin and destination hosts or networks. They may also allow access for authenticated users.

Figure 2-4 illustrates a typical setup with a firewall that filters traffic destined for a WebLogic Server cluster.

Figure 2-4 Typical Firewall Setup



You can use the following features in WebLogic Server in conjunction with firewalls:

- “Connection Filters” on page 2-24
- “Perimeter Authentication” on page 2-24

Connection Filters

You can use WebLogic Server connection filters to set up firewalls that filter network traffic based on protocols, IP addresses, and DNS node names. For more information, see “Using Network Connection Filters” in *Programming WebLogic Security*.

Perimeter Authentication

You can use Identity Assertion providers to set up perimeter authentication—a special type of authentication using tokens. The WebLogic Server security architecture supports Identity Assertion providers that perform perimeter-based authentication (Web server, firewall, VPN) and handle multiple security token types/protocols (SOAP, IIOP-CSIv2).

J2EE and WebLogic Security

For implementation and use of user authentication and authorization, BEA WebLogic Server utilizes the security services of the SDK version 1.4.1 for the Java 2 Platform, Enterprise Edition (J2EE). Like the other J2EE components, the security services are based on standardized, modular components. BEA WebLogic Server implements these Java security service methods according to the standard, and adds extensions that handle many details of application behavior automatically, without requiring additional programming.

BEA WebLogic Server’s support for SDK 1.4.1 security means that application developers can take advantage of Sun Microsystems’ latest enhancements and developments in the area of security, thus leveraging a company’s investment in Java programming expertise. By following the defined and documented Java standard, WebLogic Server’s security support has a common baseline for Java developers. The innovations that WebLogic Server provides rest on the baseline support for SDK 1.4.1.

The following topics are discussed in this section:

- “SDK 1.4.1 Security Packages” on page 2-24
- “Common Secure Interoperability Version 2 (CSIv2)” on page 2-26

SDK 1.4.1 Security Packages

WebLogic Server is compliant with and supports the following SDK 1.4.1 security packages:

- "The Java Secure Socket Extension (JSSE)" on page 2-25
- "Java Authentication and Authorization Services (JAAS)" on page 2-25
- "The Java Security Manager" on page 2-25
- "Java Cryptography Architecture and Java Cryptography Extensions (JCE)" on page 2-26

The Java Secure Socket Extension (JSSE)

JSSE is a set of packages that support and implement the SSL and TLS v1 protocol, making those protocols and capabilities programmatically available. WebLogic Server provides Secure Sockets Layer (SSL) support for encrypting data transmitted across WebLogic Server clients, as well as other servers.

While JSSE provides a core set of classes for SSL functions, other companies, such as Certicom, provide extensions to those classes. WebLogic Server uses the Certicom JSSE extensions in its implementation of SSL.

Java Authentication and Authorization Services (JAAS)

JAAS is a set of packages that provide a framework for user-based authentication and access control. BEA WebLogic Server uses only the authentication classes of JAAS.

JAAS is used as follows:

- For remote Java client authentication
- For authentication internally in instances of WebLogic Server in the Web and EJB containers and in the WebLogic Authentication and Identity Assertion providers.

For more information on JAAS, see "Java Authentication and Authorization Service (JAAS)" on page 2-3.

The Java Security Manager

Developed by Sun Microsystems, Inc., the Java Security Manager is the security manager for the Java Virtual Machine (JVM). The security manager works with the Java API to define security boundaries through the `java.lang.SecurityManager` class. The `SecurityManager` class enables programmers to establish a custom security policy for their Java applications.

The Java Security Manager can be used with WebLogic Server to provide additional protection for WebLogic resources running in the JVM. Use of the Java Security Manager to protect WebLogic resources in WebLogic Server is an optional security step.

You can use the Java Security Manager to perform the following security tasks to protect WebLogic resources:

- Modify the `weblogic.policy` file for general use.
- Set application-type security policies on EJBs and Resource Adapters.

  You use the Java security policy file to perform this task.
- Set application-specific security policies on specific EJBs and Resource Adapters.

  You use the deployment descriptors (`weblogic.xml`, `weblogic-ejb-jar.xml`, and `rar.xml`) to perform this task.

For more information on how to use the Java Security Manager to perform these tasks, see "Using the Java Security Manager to Protect WebLogic Resources" in *Programming WebLogic Security*.

Java Cryptography Architecture and Java Cryptography Extensions (JCE)

Developed by Sun Microsystems, Inc., these security APIs provide a framework for accessing and developing cryptographic functionality for the Java platform and developing implementations for encryption, key generation and key agreement, and Message Authentication Code (MAC) algorithms.

WebLogic Server fully supports these security APIs.

Common Secure Interoperability Version 2 (CSIv2)

WebLogic Server provides support for the Enterprise JavaBean (EJB) interoperability protocol that is based on Internet Inter-ORB (IIOP) (GIOP version 1.2) and the CORBA Common Secure Interoperability version 2 (CSIv2) specification. CSIv2 support in WebLogic Server:

- Interoperates with the Java 2 Enterprise Edition (J2EE) version 1.4.1 reference implementation.
- Allows WebLogic Server IIOP clients to specify a username and password in the same manner as T3 clients.
- Supports Generic Security Services Application Programming Interface (GSSAPI) initial context tokens. For this release, only usernames and passwords and GSSUP (Generic Security Services Username Password) tokens are supported.

Note: The CSIv2 implementation in WebLogic Server passed Java 2 Enterprise Edition (J2EE) Compatibility Test Suite (CTS) conformance testing.

The external interface to the CSIv2 implementation is a JAAS LoginModule that retrieves the username and password of the CORBA object. The JAAS LoginModule can be used in a WebLogic Java client or in a WebLogic Server instance that acts as a client to another J2EE application server. The JAAS LoginModule for the CSIv2 support is called `UsernamePasswordLoginModule`, and is located in the `weblogic.security.auth.login` package.

Note: For information related to load balancing support for CSIv2 in a WebLogic Server cluster, see "Server Affinity and IIOP Client Authentication Using CSIv2" in *Using WebLogic Server Clusters*.

CHAPTER 3

Security Realms

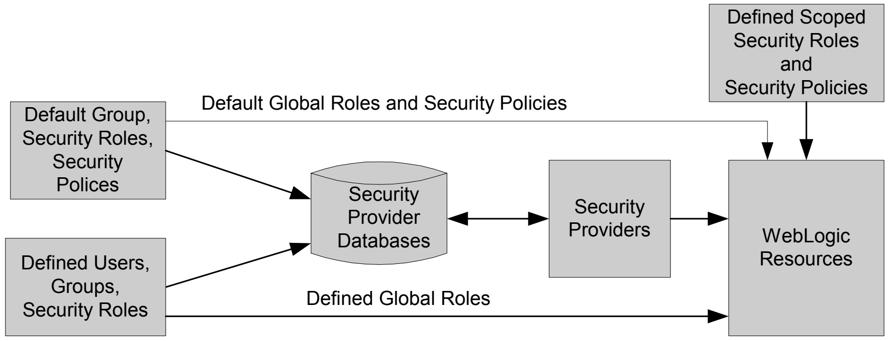
This section covers the following topics:

- "Introduction to Security Realms" on page 3-1
- "Users" on page 3-2
- "Groups" on page 3-3
- "Security Roles" on page 3-3
- "Security Policies" on page 3-4
- "Security Providers" on page 3-4

Introduction to Security Realms

In this release of WebLogic Server, a security realm comprises mechanisms for protecting WebLogic resources. Each security realm consists of a set of configured security providers, users, groups, security roles, and security policies (see Figure 3-1). A user must be defined in a security realm in order to access any WebLogic resources belonging to that realm. When a user attempts to access a particular WebLogic resource, WebLogic Server tries to authenticate and authorize the user by checking the security role assigned to the user in the relevant security realm and the security policy of the particular WebLogic resource.

Figure 3-1 WebLogic Server Security Realm


Defined Scoped Security Roles and Security Policies
Default Global Roles and Security Policies
Default Group, Security Roles, Security Polices
Security Provider Databases
Security Providers
WebLogic Resources
Defined Users, Groups, Security Roles
Defined Global Roles

Users

Users are entities that can be authenticated in a security realm, such as `myrealm` (see Figure 3-1). A user can be a person, such as application end user, or a software entity, such as a client application, or other instances of WebLogic Server. As a result of authentication, a user is assigned an identity, or principal. Each user is given a unique identity within the security realm. Users may be placed into groups that are associated with security roles, or be directly associated with security roles.

When users want to access WebLogic Server, they present proof material (for example, a password or a digital certificate) typically through a JAAS LoginModule to the Authentication provider configured in the security realm. If WebLogic Server can verify the identity of the user based on that username and credential, WebLogic Server associates the principal assigned to the user with a thread that executes code on behalf of the user. Before the thread begins executing code, however, WebLogic Server checks the security policy of the WebLogic resource and the principal (that the user has been assigned) to make sure that the user has the required permissions to continue.

When you use the WebLogic Authentication provider and you define a user, you also define a password for that user. WebLogic Server hashes all passwords. Subsequently, when WebLogic Server receives a client request, the password presented by the client is hashed and WebLogic Server compares it to the already hashed password to see if it matches.

Note: All user names and groups must be unique within a security realm.

For more information, see "Users and Groups" in *Securing WebLogic Resources.*

Groups

Groups are logically ordered sets of users (see Figure 3-1). Usually, group members have something in common. For example, a company may separate its sales staff into two groups, Sales Representatives and Sales Managers. Companies may do this because they want their sales personnel to have different levels of access to WebLogic resources, depending on their job functions.

Managing groups is more efficient than managing large numbers of users individually. For example, an administrator can specify permissions for 50 users at one time by placing the users in a group, assigning the group to a security role, and then associating the security role with a WebLogic resource via a security policy.

All user names and groups must be unique within a security realm.

For more information, see "Users and Groups" in *Securing WebLogic Resources.*

Security Roles

A security role is a privilege granted to users or groups based on specific conditions (see Figure 3-1). Like groups, security roles allow you to restrict access to WebLogic resources for several users at once. However, unlike groups, security roles:

- Are computed and granted to users or groups dynamically, based on conditions such as user name, group membership, or the time of day.
- Can be scoped to specific WebLogic resources within a single application in a WebLogic Server domain (unlike groups, which are always scoped to an entire WebLogic Server domain).

Granting a security 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 "in" the security role. Multiple users or groups can be granted a single security role.

Note: In WebLogic Server 6.x, security roles applied to Web applications and Enterprise JavaBeans (EJBs) only. In this release of WebLogic Server, the use of security roles is expanded to include all of the defined WebLogic resources.

For more information on security roles, see "Security Roles" in *Securing WebLogic Resources*.

Security Policies

A security policy is an association between a WebLogic resource and one or more users, groups, or security roles. Security policies protect the WebLogic resource against unauthorized access. A WebLogic resource has no protection until you create a security policy for it.

Note: Security policies replace the access control lists (ACLs) that were used to protect WebLogic resources in WebLogic Server 6.x.

For more information on security policies, see "Security Policies" on page 2-11.

Security Providers

Security providers are modules that provide security services to applications to protect WebLogic resources (see Figure 3-1). You can use the security providers that are provided as part of the WebLogic Server product, purchase custom security providers from third-party security vendors, or develop your own custom security providers. For information on how to develop custom security providers, see *Developing Security Providers for WebLogic Server*.

The following topics are discussed in this section.

- "Security Provider Databases" on page 3-4
- "Types of Security Providers" on page 3-7
- "Security Providers and Security Realms" on page 3-15

Security Provider Databases

The following sections explain what a security provider database is and describe how security realms affect the use of security provider databases:

- "What Is a Security Provider Database?" on page 3-4
- "Security Realms and Security Provider Databases" on page 3-5
- "Embedded LDAP Server" on page 3-6

What Is a Security Provider Database?

A security provider database contains the users, groups, security roles, security policies, and credentials used by some types of security providers to provide security services (see Figure 3-1). For example: an Authentication provider requires information about users and groups; an Authorization provider requires information about security policies; a Role Mapping provider requires information about security roles, and a Credential Mapping provider requires information about credentials to be used with a Resource Adapter to access an Enterprise Information System (EIS). These security providers need this information to be available in a database in order to function properly.

The security provider database can be the embedded LDAP server (as used by the WebLogic security providers), a properties file (as used by the sample custom security providers, available on the Web), or a production-quality, customer-supplied database that you may already be using.

Note: The sample custom security providers are available at `http://dev2dev.bea.com/code/wls.jsp` on the BEA dev2dev Web site.

The security provider database should be initialized the first time security providers are used. (That is, before the security realm containing the security providers is set as the default (active) security realm.) This initialization can be done:

- When a WebLogic Server instance boots.
- When a call is made to one of the security provider's MBeans.

At minimum, the security provider database is initialized with the default groups, security roles, security policies provided by WebLogic Server. For more information, see Security Providers and WebLogic Resources in *Developing Security Providers for WebLogic Server*.

Security Realms and Security Provider Databases

If you have multiple security providers of the *same type* configured in the *same security realm*, these security providers may use the same security provider database. This behavior holds true for all of the WebLogic security providers and the sample security providers that are available `http://dev2dev.bea.com/code/wls.jsp` on the BEA dev2dev Web site.

For example, if you configure two WebLogic Authentication providers in the default security realm (called `myrealm`), both WebLogic Authentication providers will use the same location in the embedded LDAP server as their security provider database, and thus, will use the same users and groups. Furthermore, if you or an administrator add a user or group to one of the WebLogic Authentication providers, you will see that user or group appear for the other WebLogic Authentication provider as well.

Note: If you have two WebLogic security providers (or two sample security providers) of the *same type* configured in *two different security realms*, each will use its own security provider database. Only one security realm can be active at a time.

Custom security providers that you develop (or the custom security providers that you obtain from third-party security vendors) can be designed so that each instance of the security provider uses its own database *or* so that all instances of the security provider in a security realm share the same database. This is a design decision that you need to make based on your existing systems and security requirements. For more information about design decisions that affect security providers, see Design Considerations in *Developing Security Providers for WebLogic Server*.

Embedded LDAP Server

The embedded LDAP server is used as the database that stores user, group, security roles, and security policies for the WebLogic security providers. The embedded LDAP server is a complete LDAP server. It supports the following access and storage functions:

- Access and modification of entries in the LDAP server
- Use of an LDAP browser to import and export security data into and from the LDAP server.
- Read and write access by the WebLogic security providers.

Note: WebLogic Server does not support adding attributes to the embedded LDAP server.

Table 3-1 shows how each of the WebLogic security providers uses the embedded LDAP server.

Table 3-1 Usage of the Embedded LDAP Server

WebLogic Security Provider	Embedded LDAP Server Usage
Authentication	Stores user and group information.
Identity Assertion	None.
Authorization	Stores security roles and security policies.
Adjudication	None.
Role Mapping	Supports dynamic role associations by obtaining a computed set of roles granted to a requestor for a given WebLogic resource.
Auditing	None.
Credential Mapping	Stores Username-Password credential mapping information.

Types of Security Providers

The following sections describe the types of security providers that you can use with WebLogic Server:

- "Authentication Providers" on page 3-7
- "Identity Assertion Providers" on page 3-8
- "Principal Validation Providers" on page 3-9
- "Authorization Providers" on page 3-10
- "Adjudication Providers" on page 3-10
- "Role Mapping Providers" on page 3-11
- "Auditing Providers" on page 3-12
- "Credential Mapping Providers" on page 3-12
- "Keystore Providers" on page 3-13
- "Realm Adapter Providers" on page 3-13

Note: You cannot develop a single security provider that merges several provider types (for example, you cannot have one security provider that does authorization *and* role mapping).

Authentication Providers

Authentication providers allow WebLogic Server to establish trust by validating a user. The WebLogic Server security architecture supports Authentication providers that perform: username/password authentication, certificate-based authentication directly with WebLogic Server, and HTTP certificate-based authentication proxied through an external Web server.

Note: An Identity Assertion provider is a special type of Authentication provider that handles perimeter-based authentication and multiple security token types/protocols. For more information, see "Identity Assertion Providers" on page 3-8.

A LoginModule is the part of an Authentication provider that actually performs the authentication of a user or system. Authentication providers also use Principal Validation providers which provide additional security by signing and verifying the authenticity of principals (users/groups). For more information about Principal Validation providers, see Principal Validation Providers in *Developing Security Providers for WebLogic Server*.

You must have at least one Authentication provider in a security realm, and you can configure multiple Authentication providers in a security realm. Having multiple Authentication providers allows you to have multiple LoginModules, each of which may perform a different kind of authentication. An administrator configures each Authentication provider to determine how multiple LoginModules are called when users attempt to login to the system. Because they add security to the principals used in authentication, a Principal Validation provider must be accessible to your Authentication providers.

Authentication providers and LoginModules are discussed in more detail in Authentication Providers in *Developing Security Providers for WebLogic Server*.

Identity Assertion Providers

Identity assertion involves establishing a client's identity using client-supplied tokens that may exist *outside* of the request. Thus, the function of an Identity Assertion provider is to validate and map a token to a username. Once this mapping is complete, an Authentication provider's LoginModule can be used to convert the username to principals. Identity Assertion providers allow WebLogic Server to establish trust by validating a user.

An Identity Assertion provider is a specific form of Authentication provider that allows users or system processes to assert their identity using tokens (in other words, perimeter authentication). You can use an Identity Assertion provider in place of an Authentication provider if you create a LoginModule for the Identity Assertion provider, or in addition to an Authentication provider if you want to use the Authentication provider's LoginModule. Identity Assertion providers enable perimeter authentication and support single sign-on.

The WebLogic Server security architecture supports Identity Assertion providers that perform perimeter-based authentication (Web server, firewall, VPN). You can write Identity Assertion providers that support different token types, such as Kerberos, SAML (Security Assertion Markup Language) and Microsoft Passport, and can handle multiple security protocols (SOAP, IIOP-CSIv2). When used with an Authentication provider's LoginModule, 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.

Note: To use the WebLogic Identity Assertion provider for X.501 and X.509 certificates, you have the option of using the default user name mapper that is supplied with the WebLogic Server product (`weblogic.security.providers.authentication.DefaultUserNameMapperImpl`) or providing you own implementation of the `weblogic.security.providers.authentication.UserNameMapper` interface. For more information, see Do I Need to Develop a Custom Identity Assertion Provider? in *Developing Security Providers for WebLogic Server*.

Multiple Identity Assertion providers can be configured in a security realm, but none are required. An Identity Assertion provider can support more than one token type, but only one token type at a time can be active in a particular Identity Assertion provider. For example, a particular Identity Assertion provider can support both Kerberos and SAML, but an administrator configuring the system must select which token type (Kerberos or SAML) is to be active in that Identity Assertion provider. For example, if there only one Identity Assertion provider configured and it is set to handle Kerberos tokens, but SAML token types must be supported as well, then another Identity Assertion provider must be configured that can handle SAML tokens and SAML must be set as its active token type.

Identity Assertion providers are discussed in more detail in Identity Assertion Providers in *Developing Security Providers for WebLogic Server*.

Principal Validation Providers

A Principal Validation provider is a special type of security provider that primarily acts as a "helper" to an Authentication provider. Because some LoginModules can be remotely executed on behalf of RMI clients, and because the client application code can retain the authenticated subject between programmatic server invocations, Authentication providers rely on Principal Validation providers to provide additional security protections for the principals contained within the subject.

Principal Validation providers provide these additional security protections by signing and verifying the authenticity of the principals. This principal validation provides an additional level of trust and may reduce the likelihood of malicious principal tampering. Verification of the subject’s principals takes place during the WebLogic Server’s demarshalling of RMI client requests for each invocation. The authenticity of the subject’s principals is also verified when making authorization decisions.

Because you must have at least one Authentication provider in a security realm, you must also have one Principal Validation provider in a security realm. If you have multiple Authentication providers, each of those Authentication providers must have a corresponding Principal Validation provider.

Note: You cannot use the Administration Console to configure Principal Validation providers directly. WebLogic Server configures the required Principal Validation providers for you when you configure your Authentication providers.

Principal Validation providers are discussed in more detail in Principal Validation Providers in *Developing Security Providers for WebLogic Server*.

Authorization Providers

Authorization providers control access to WebLogic resources based on the security role a user or group is granted, and the security policy assigned to the requested WebLogic resource. For more information about WebLogic resources, security roles, and security policies, see *Securing WebLogic Resources*."

An Access Decision is the part of the Authorization provider that actually determines whether a subject has permission to perform a given operation on a WebLogic resource. For more information about, see Principal Validation Providers in *Developing Security Providers for WebLogic Server*.

You must have at least one Authorization provider in a security realm, and you can configure multiple Authorization providers in a security realm. Having multiple Authorization providers allows you to follow a more modular design. For example, you may want to have one Authorization provider that handles Web application and Enterprise JavaBean (EJB) permissions and another that handles permissions for other types of WebLogic resources. Another example might be to have one Authorization provider that handles domestic employees, and another that handles permissions for overseas employees.

Authorization providers and Access Decisions are discussed in more detail in Authorization Providers in *Developing Security Providers for WebLogic Server*.

Adjudication Providers

As part of an Authorization provider, an Access Decision determines whether a subject has permission to access a given WebLogic resource. Therefore, if multiple Authorization providers are configured, each may return a different answer to the “is access allowed?” question. These answers may be `PERMIT`, `DENY`, or `ABSTAIN`. Determining what to do if multiple Authorization providers’ Access Decisions do not agree on an answer is the function of an Adjudication provider. The Adjudication provider resolves authorization conflicts by weighing each Access Decision’s answer and returning a final result. If you only have one Authorization provider and no Adjudication provider, then an `ABSTAIN` returned from the single Authorization provider’s Access Decision is treated like a `DENY`.

Note: The WebLogic Adjudication provider supports the use of the Administration Console to control whether an abstain is treated as a permit or a deny.

You must configure an Adjudication provider in a security realm *only* if you have multiple Authorization providers configured. You can have only one Adjudication provider in a security realm.

Note: Since the default security realm has only one Authorization provider, it does not require an Adjudication provider, even though an Adjudication provider is provided. However, the Compatibility realm has two Authorization providers, so that realm does require an Adjudication provider.

Adjudication providers are discussed in more detail in Adjudication Providers in *Developing Security Providers for WebLogic Server*.

Role Mapping Providers

A Role Mapping provider supports dynamic role associations by obtaining a computed set of security roles granted to a requestor for a given WebLogic resource. The WebLogic Security Framework determines which security roles (if any) apply to a particular subject at the moment that access is required for a given WebLogic resource by:

- Obtaining security roles from the J2EE and WebLogic deployment descriptor files.
- Using business logic and the current operation parameters to determine security roles.

A Role Mapping provider supplies Authorization providers with this security role information so that the Authorization provider can answer the “is access allowed?” question for WebLogic resources that use role-based security (that is, Web application and Enterprise JavaBean container resources).

You set security roles in J2EE deployment descriptors, or create them using the WebLogic Server Administration Console. Security roles set in deployment descriptors are applied at deployment time (unless you specifically choose to ignore deployment descriptors).

You must have at least one Role Mapping provider in a security realm, and you can configure multiple Role Mapping providers in a security realm. Having multiple Role Mapping providers allows you to work within existing infrastructure requirements (for example, configuring one Role Mapping provider for each LDAP server that contains user and security role information), or follow a more modular design (for example, configuring one Role Mapping provider that handles mappings for Web applications and Enterprise JavaBeans (EJBs) and another that handles mappings for other types of WebLogic resources).

Note: If multiple Role Mapping providers are configured, the set of security roles returned by all Role Mapping providers will be *intersected* by the WebLogic Security Framework. That is, security role names from all the Role Mapping providers will be merged into single list, with duplicates removed.

Role Mapping providers are discussed in more detail in Role Mapping Providers in *Developing Security Providers for WebLogic Server*.

Auditing Providers

An Auditing provider collects, stores, and distributes information about operating requests and the outcome of those requests for the purposes of non-repudiation. An Auditing provider makes the decision about whether to audit a particular event based on specific audit criteria, including audit severity levels. Auditing providers can write the audit information to output repositories such as an LDAP back-end, database, or simple file. Specific actions, such as paging security personnel, can also be configured as part of an Auditing provider.

Other types of security providers (such as Authentication or Authorization providers) can request audit services before and after security operations have been performed by calling through the WebLogic Security Framework. For more information, see Auditing Events From Custom Security Providers in *Developing Security Providers for WebLogic Server*.

You can configure multiple Auditing providers in a security realm, but none are required.

Auditing providers are discussed in more detail in Auditing Providers in *Developing Security Providers for WebLogic Server*.

Credential Mapping Providers

A credential map is a mapping of credentials used by WebLogic Server to credentials used in a legacy (or any remote) system, which tell WebLogic Server how to connect to a given resource in that system. In other words, credential maps allow WebLogic Server to log into a remote system on behalf of a subject that has already been authenticated.

A Credential Mapping provider can handle several different types of credentials (for example, username/password combinations, Kerberos tickets, and public key certificates). You can set credential mappings in deployment descriptors or by using the WebLogic Server Administration Console. These credential mappings are applied at deploy time (unless you specifically choose to ignore the credential mappings).

You must have at least one Credential Mapping provider in a security realm, and you can configure multiple Credential Mapping providers in a security realm. If multiple Credential Mapping providers are configured, then the WebLogic Security Framework calls into each Credential Mapping provider to find out if they contain the type of credentials requested by the container. The WebLogic Security Framework then accumulates and returns all the credentials as a list.

Credential Mapping providers are discussed in more detail in Credential Mapping Providers in *Developing Security Providers for WebLogic Server*.

Keystore Providers

With WebLogic Server, a keystore creates and manages password-protected stores of private keys (and their associated public key certificates) and trusted certificate authorities.

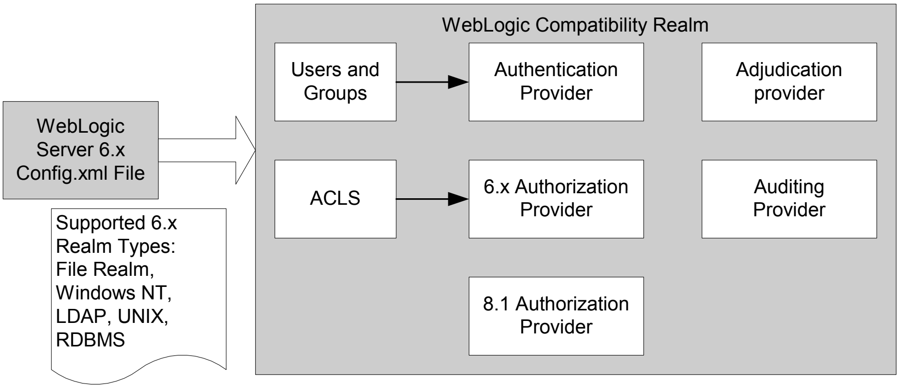
The WebLogic Keystore provider that is included as part of the WebLogic Server product is used to obtain secured private keys from keystores.

Note: The WebLogic Keystore provider is deprecated in this release of WebLogic Server but is still supported. The development of custom Keystore providers is not supported. Use Java KeyStores (JKS) instead. All of the functionality that was supported by the WebLogic Keystore provider is available through use of Java KeyStores. The WebLogic Keystore provider is only supported for backward compatibility. BEA recommends using the WebLogic Keystore provider only when it is needed to support backward compatibility with a WebLogic Server 7.0 configuration. For information on how to use Java KeyStores, see Configuring Keystores in *Managing WebLogic Server*.

Realm Adapter Providers

Realm Adapter providers provide backward-compatibility with 6.x WebLogic security realms by allowing the use of existing, 6.x security realms with the security features in this release of WebLogic Server. The Realm Adapter providers map the realm API (`weblogic.security.acl`) used in WebLogic Server 6.x to the APIs used in this release of WebLogic Server. Figure 3-2 shows a Compatibility realm and the types of security providers supported.

Figure 3-2 Compatibility Realm


WebLogic Compatibility Realm
Users and Groups
Authentication Provider
Adjudication provider
WebLogic Server 6.x Config.xml File
ACLS
6.x Authorization Provider
Auditing Provider
Supported 6.x Realm Types: File Realm, Windows NT, LDAP, UNIX, RDBMS
8.1 Authorization Provider

Security Provider Summary

Table 3-2 indicates whether you can configure multiple security providers of the same type in a security realm.

Table 3-2 Multiple Providers of Same Type in Same Security Realm

Type	Multiple Providers Supported?
Authentication provider	Yes
Identity Assertion provider	Yes
Principal Validation provider	Yes
Authorization provider	Yes
Adjudication provider	No
Role Mapping provider	Yes
Auditing provider	Yes
Credential Mapping provider	Yes
Keystore provider	Yes

Table 3-2 Multiple Providers of Same Type in Same Security Realm

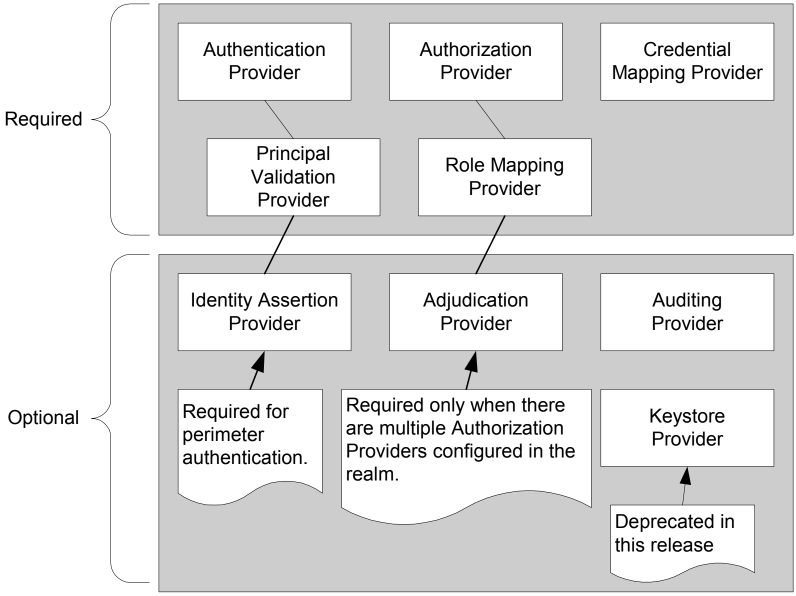
Type	Multiple Providers Supported?
Realm Adapter provider	Yes for all types of Realm Adapter providers supported except the Adjudication provider. See Figure 3-2 for the supported types.

Security Providers and Security Realms

All security providers exist within the context of a security realm. If you are *not* running a prior, 6.x release of WebLogic Server, the WebLogic Server security realm defined out-of-the-box as the default realm (that is, the active security realm called `myrealm`) contains the WebLogic security providers displayed in Figure 3-3.

Note: If you are upgrading from a 6.x release to the 8.1 release, your out-of-the-box experience begins with a Compatibility realm—which is initially defined as the default realm—to allow you to work with your existing configuration. Because the 6.x model is deprecated, you need to upgrade your security realm to the 8.1 model. see "Security" under "Upgrading WebLogic Server 6.x to Version 8.1" in the *Upgrade Guide for BEA WebLogic Server 8.1*.

Figure 3-3 WebLogic Security Providers in a Security Realm


Required
Optional
Authentication Provider
Authorization Provider
Credential Mapping Provider
Principal Validation Provider
Role Mapping Provider
Identity Assertion Provider
Adjudication Provider
Auditing Provider
Required for perimeter authentication.
Required only when there are multiple Authorization Providers configured in the realm.
Keystore Provider
Deprecated in this release

Because security providers are individual modules or components that are "plugged into" a WebLogic Server security realm, you can add, replace, or remove a security provider with minimal effort. You can use the WebLogic security providers, custom security providers you develop, security providers obtained from third-party security vendors, or a combination of all three to create a fully-functioning security realm. However, as Figure 3-3 also shows, some types of security providers are required for a 8.1 security realm to operate properly. Table 3-3 summarizes which security providers must be configured for a fully-operational 8.1 security realm.

Table 3-3 Security Providers in a Security Realm

Type	Required?
Authentication provider	Yes
Identity Assertion provider	Yes, if using perimeter authentication.
Principal Validation provider	Yes
Authorization provider	Yes

Table 3-3 Security Providers in a Security Realm

Type	Required?
Adjudication provider	Yes, if there are multiple Authorization providers configured.
Role Mapping provider	Yes
Auditing provider	No
Credential Mapping provider	Yes
Keystore provider	No **Note:** The WebLogic Keystore provider is deprecated in this release of WebLogic Server but is still supported. The development of custom Keystore providers is not supported. Use Java KeyStores (JKS) instead. BEA recommends using the WebLogic Keystore provider only when it is needed to support backward compatibility with a WebLogic Server 7.0 configuration. For information on how to use Java KeyStores, see Configuring Keystores in *Managing WebLogic Server*.

For more information about security realms, see the following topics in *Managing WebLogic Security*:

- Configuration Steps for Security
- Setting a New Security Realm as the Default (Active) Security Realm
- Deleting a Security Realm

WebLogic Security Service Architecture

This section covers the following topics:

- "Architectural Overview" on page 4-1
- "How the Architecture Benefits Users" on page 4-18

Architectural Overview

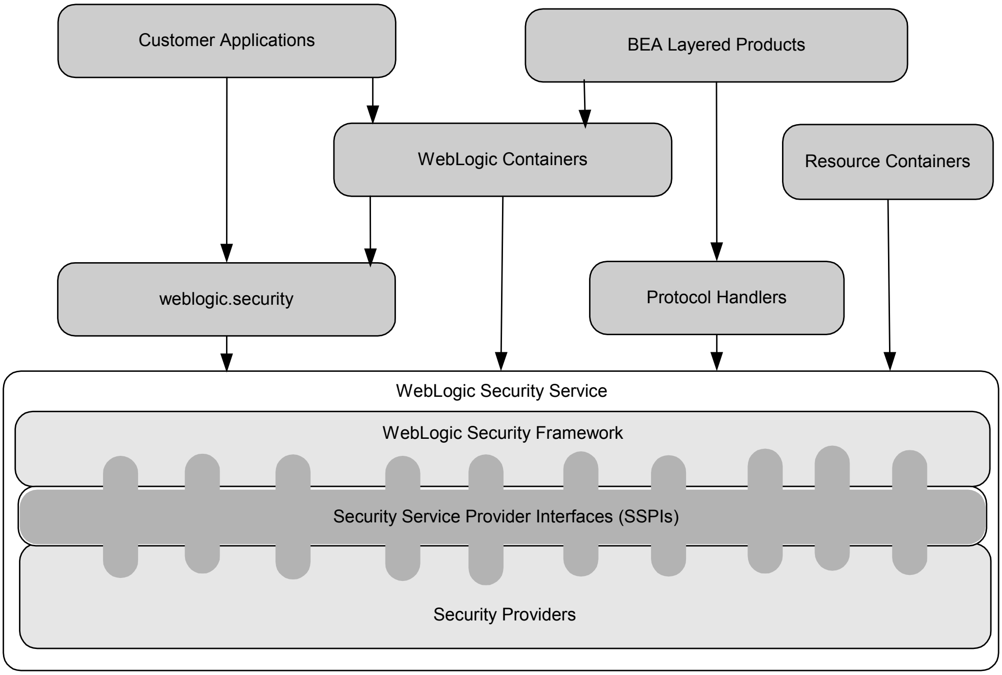
This section provides a description of the architecture of the WebLogic Security Service. The architecture comprises three major components, which are discussed in the following sections:

- "WebLogic Security Framework" on page 4-1.
- "The Security Service Provider Interfaces (SSPIs)" on page 4-11.
- "The WebLogic Security Providers" on page 4-11.

WebLogic Security Framework

Figure 4-1 shows a high-level view of the WebLogic Security Framework. The framework comprises interfaces, classes, and exceptions in the `weblogic.security.service` package.

Figure 4-1 WebLogic Security Service Architecture



The primary function of the WebLogic Security Framework is to provide a simplified application programming interface (API) that can be used by security and application developers to define security services. Within that context, the WebLogic Security Framework also acts as an intermediary between the WebLogic containers (Web and EJB), the Resource containers, and the security providers.

The following sections describe the interactions between the WebLogic containers and Resource containers and each of the security providers via the WebLogic Security Framework:

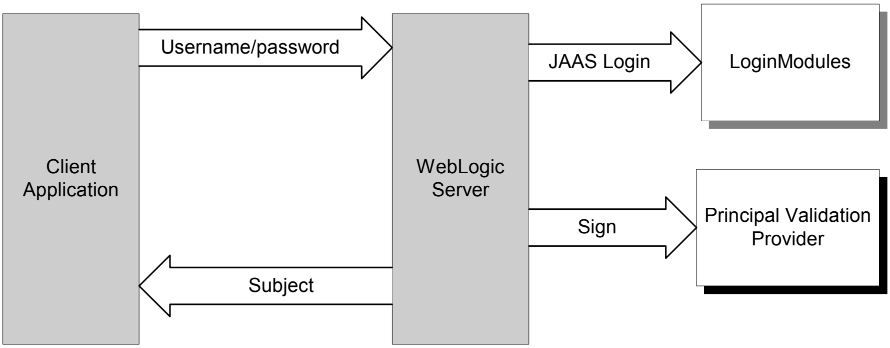
- "The Authentication Process" on page 4-3
- "The Identity Assertion Process" on page 4-4
- "SSO with Microsoft Clients Process" on page 4-9
- "The Principal Validation Process" on page 4-5
- "The Authorization Process" on page 4-5

- "The Adjudication Process" on page 4-6
- "The Role Mapping Process" on page 4-7
- "The Auditing Process" on page 4-8
- "The Credential Mapping Process" on page 4-9
- "SSO with Microsoft Clients Process" on page 4-9

The Authentication Process

Figure 4-2 shows the authentication process for a fat-client login. JAAS runs on the server to perform the login. Even in the case of a thin-client login (that is, a Web browser client) JAAS is still run on the server.

Figure 4-2 The Authentication Process



Notes: Only developers of custom Authentication providers will be involved with this JAAS process directly. The client application could either use a JNDI Initial Context or JAAS to initiate the passing of the username and password.

When a user attempts to log into a system using a username/password combination, WebLogic Server establishes trust by validating that user's username and password, and returns a subject that is populated with principals per JAAS requirements. As Figure 4-2 also shows, this process requires the use of a LoginModule and a Principal Validation provider. For more information on Principal Validation providers, see "WebLogic Principal Validation Provider" on page 4-14.

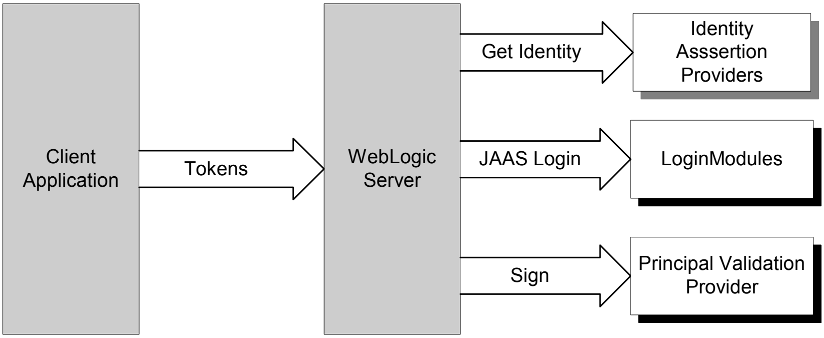
After successfully proving a caller's identity, an authentication context is established, which allows an identified user or system to be authenticated to other entities. Authentication contexts may also be delegated to an application component, allowing that component to call another application component while impersonating the original caller.

The Identity Assertion Process

Identity Assertion providers are used as part of perimeter authentication process. When perimeter authentication is used (see Figure 4-3), a token from outside of the WebLogic Server domain is passed to an Identity Assertion provider in a security realm that is responsible for validating tokens of that type and that is configured as "active." If the token is successfully validated, the Identity Assertion provider maps the token to a WebLogic Server username, and sends that username back to WebLogic Server, which then continues the authentication process. Specifically, the username is sent via a JAAS `CallbackHandler` and passed to each configured Authentication provider's LoginModule, so that the LoginModule can populate the subject with the appropriate principals.

Note: To use the WebLogic Identity Assertion provider for X.501 and X.509 certificates, you have the option of using the default user name mapper that is supplied with the WebLogic Server product (`weblogic.security.providers.authentication.DefaultUserNameMapperImpl`) or providing you own implementation of the `weblogic.security.providers.authentication.UserNameMapper` interface. For more information, see Do I Need to Develop a Custom Identity Assertion Provider? in *Developing Security Providers for WebLogic Server*.

Figure 4-3 Perimeter Authentication

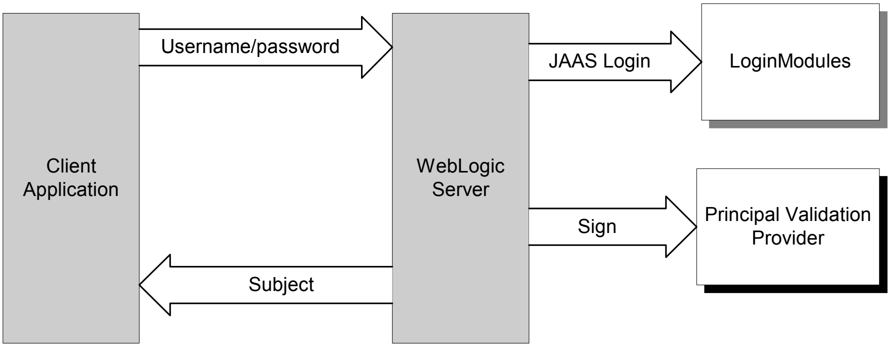


As Figure 4-3 also shows, perimeter authentication requires the same components as the authentication process, but also adds an Identity Assertion provider.

The Principal Validation Process

As shown in Figure 4-4, a user attempts to log into a system using a username/password combination. WebLogic Server establishes trust by calling the configured Authentication provider's LoginModule, which validates the user's username and password and returns a subject that is populated with principals per JAAS requirements.

Figure 4-4 The Principal Validation Process

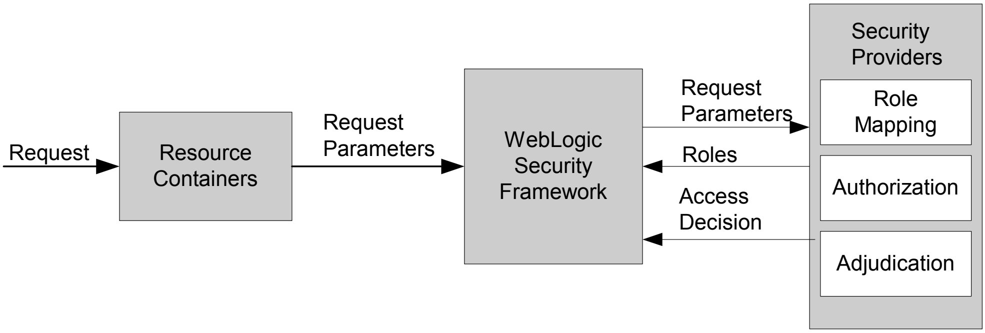


WebLogic Server passes the subject to the specified Principal Validation provider, which signs the principals and then returns them to the client application via WebLogic Server. Whenever the principals stored within the subject are required for other security operations, the same Principal Validation provider will verify that the principals stored within the subject have not been modified since they were signed.

The Authorization Process

Figure 4-5 illustrates how Authorization providers (and the associated Adjudication and Role Mapping providers) interact with the WebLogic Security Framework during the authorization process.

Figure 4-5 Authorization Process



The authorization process is initiated when a user or system process requests a WebLogic resource on which it will attempt to perform a given operation. The resource container that handles the type of WebLogic resource being requested receives the request (for example, the EJB container receives the request for an EJB resource). The resource container calls the WebLogic Security Framework and passes in the request parameters, including information such as the subject of the request and the WebLogic resource being requested. The WebLogic Security Framework calls the configured Role Mapping providers and passes in the request parameters in a format that the Role Mapping providers can use. The Role Mapping providers use the request parameters to compute a list of roles to which the subject making the request is entitled and passes the list of applicable roles back to the WebLogic Security Framework. The Authorization provider determines whether the subject is entitled to perform the requested action on the WebLogic resource, that is, the Authorization provider makes the Access Decision. If there are multiple Authorization providers configured, the WebLogic Security Framework delegates the job of reconciling any conflicts in the Access Decisions rendered by the Authorization providers to the Adjudication provider and the Adjudication provider determines the ultimate outcome of the authorization decision.

The Adjudication Process

If there are multiple Authorization providers configured (see Figure 4-5), an Adjudication provider is required to tally the multiple Access Decisions and render a verdict. The Adjudication provider returns either a `TRUE` or `FALSE` verdict to the Authorization providers, which forward it to the resource container through the WebLogic Security Framework.

– If the decision is `TRUE`, the resource container dispatches the request to the protected WebLogic resource.

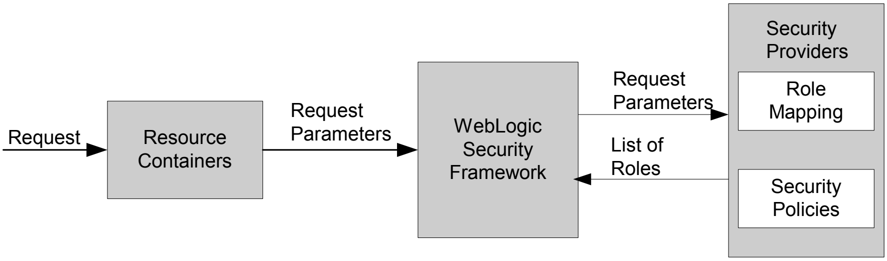
– If the decision is FALSE, the resource container throws a security exception that indicates that the requestor was not authorized to perform the requested access on the protected WebLogic resource.

The Role Mapping Process

The WebLogic Security Framework calls each Role Mapping provider that is configured for a security realm as part of an authorization decision. For related information, see "The Authorization Process" on page 4-5.

Figure 4-6 shows how the Role Mapping providers interact with the WebLogic Security Framework to create dynamic role associations.

Figure 4-6 Role Mapping Process



The role mapping process is initiated when a user or system process requests a WebLogic resource on which it will attempt to perform a given operation. The resource container that handles the type of WebLogic resource being requested receives the request (for example, the EJB container receives the request for an EJB resource). The resource container calls the WebLogic Security Framework and passes in the request parameters, including information such as the subject of the request and the WebLogic resource being requested. The WebLogic Security Framework calls each configured Role Mapping provider to obtain a list of the roles that apply. If a security policy specifies that the requestor is entitled to a particular role, 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 WebLogic resource or the resource container have been evaluated. The list of roles is returned to the WebLogic Security Framework, where it can be used as part of other operations, such as access decisions.

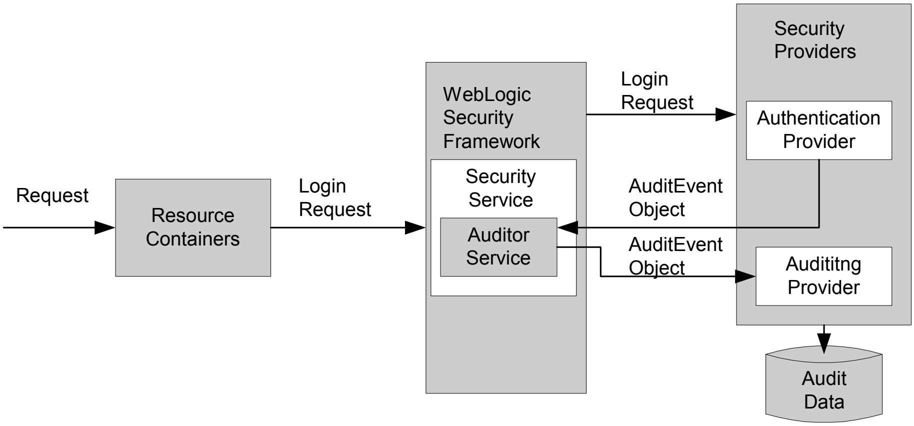
The result of the dynamic role association (performed by the Role Mapping providers) is a set of roles that apply to the principals stored in a subject at a given moment. These roles can then be used to make authorization decisions for protected WebLogic resources, as well as for resource container and application code. For example, an Enterprise JavaBean (EJB) could use the Java 2

Enterprise Edition (J2EE) `isCallerInRole` method to retrieve fields from a record in a database, without having knowledge of the business policies that determine whether access is allowed.

The Auditing Process

Figure 4-7 shows how Auditing providers interact with the WebLogic Security Framework and other types of security providers (using an Authentication provider as an example).

Figure 4-7 Auditing Process

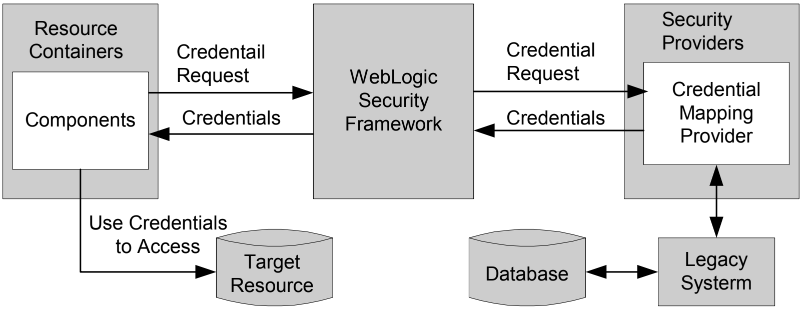


The auditing process is initiated when a resource container passes a user's authentication information (for example, a username/password combination) to the WebLogic Security Framework as part of a login request. The WebLogic Security Framework passes the information associated with the login request to the configured Authentication provider. If, in addition to providing authentication services, the Authentication provider is designed to post audit events, the Authentication provider instantiates an `AuditEvent` object. The `AuditEvent` object includes information such as the event type to be audited and an audit severity level. The Authentication provider then calls the Auditor Service in the WebLogic Security Framework, passing in the `AuditEvent` object. The Auditor Service passes the `AuditEvent` object to the configured Auditing providers' runtime classes, enabling audit event recording. The Auditing providers' runtime classes use the information obtained from the `AuditEvent` object to control audit record content. When the criteria for auditing specified by the Authentication providers in the `AuditEvent` object is met, the appropriate Auditing provider's runtime class writes out audit records. Depending on the Auditing provider implementation, audit records may be written to a file, a database, or some other persistent storage medium.

The Credential Mapping Process

Figure 4-8 illustrates how Credential Mapping providers interact with the WebLogic Security Framework during the credential mapping process.

Figure 4-8 Credential Mapping Process



The credential mapping process is initiated when application components, such as JavaServer Pages (JSPs), servlets, Enterprise JavaBeans (EJBs), or Resource Adapters call into the WebLogic Security Framework (through the appropriate resource container) to access an Enterprise Information System (EIS), for example, some relational database like Oracle, SQL Server, and so on. As part of the call, the application component passes in the subject (that is, the "who" making the request), the WebLogic resource (that is, the "what" that is being requested) and information about the type of credentials needed to access the WebLogic resource. The WebLogic Security Framework sends the application component's request for credentials to a configured Credential Mapping provider that handles the type of credentials needed by the application component. The Credential Mapping provider consults its database to obtain a set of credentials that match those requested by the application component and returns the credentials to the WebLogic Security Framework. The WebLogic Security Framework passes the credentials back to the requesting application component through the resource container. The application component uses the credentials to access the external system.

SSO with Microsoft Clients Process

SSO with Microsoft clients involves the Single Pass Negotiate Identity Assertion provider interacting with the Servlet container and the WebLogic Security Framework to achieve authentication.

The interaction works as follows:

1. A user logs into the Windows 2000 or 2003 domain. The user acquires Kerberos credentials from the domain.
2. Using a browser that supports the SPNEGO protocol (for example, Internet Explorer or Mozilla), the user tries to access a web application running on WebLogic Server. WebLogic Server can be running on a UNIX or Windows 2000/2003 platform.
3. The browser sends a `GET` request to WebLogic Server.
4. WebLogic Server sends back an unauthorized response.
5. The browser receives the `WWW-Authenticate` header and determines whether or not it can support the negotiate authentication scheme.

   If the browser can support the negotiate authentication scheme, it contacts the Kerberos key distribution center (KDC) to obtain a ticket.

   The browser uses the information in the ticket to create a SPNEGO token containing the supported GSS mechanism token types.

   The browser Base64 encodes the token and sends it back to the application server via an Authorization header on the original GET message as follows:

   ```
   GET...
   Authorization: Negotiate <Base64 encoded SPNEGO token>
   ```
6. Since the request is still unauthorized, the Servlet container in WebLogic Server handles the Authorization request header and calls the WebLogic Security Framework. The framework passes the token to the Single Pass Negotiate Identity Assertion provider.
7. The Single Pass Negotiate Identity Assertion provider decodes the SPNEGO tokens and uses GSS methods to accept the security context.
   - If the GSS accept is complete, the security context is established and the Single Pass Negotiate Identity Assertion provider uses the GSS to get the name of the initiating Principal.
   - If the GSS accept is not complete, the Servlet container returns:

     ```
     401 Unauthorized
     WWW-Authenticate: Negotiate
     ```
8. The name of the initiating Principal is mapped to a username and passed back to the WebLogic Security framework via a Callback handler.

   The WebLogic Security framework also determines to which groups the user belongs.
9. The authentication is complete and the GET request is processed.

The Security Service Provider Interfaces (SSPIs)

Security in this release of WebLogic Server is based on a set of Security Service Provider Interfaces (SSPIs). The SSPIs can be used by developers and third-party vendors to develop security providers for the WebLogic Server environment. SSPIs are available for Adjudication, Auditing, Authentication, Authorization, Credential Mapping, Identity Assertion, and Role Mapping.

Note: The SSPI for Keystore providers is deprecated in this release of WebLogic Server. Use Java KeyStores (JKS) instead. For information on how to use Java KeyStores, see Configuring Keystores in *Managing WebLogic Server*.

The SSPIs allow customers to use custom security providers for securing WebLogic Server resources. Customers can use the SSPIs to develop custom security providers or they can purchase customer security providers from third-party vendors.

Note: To assist customers in developing custom security providers, sample custom security providers are also available from the BEA online dev2dev Web site at `http://dev2dev.bea.com/code/wls.jsp`. For more information on developing custom security providers, see *Developing Security Providers for WebLogic Server*.

The WebLogic Security Providers

This section provides descriptions of the WebLogic security providers that are included in the WebLogic Server product for your use. Security providers are modules that “plug into” a WebLogic Server security realm to provide security services to applications. They call into the WebLogic Security Framework on behalf of applications.

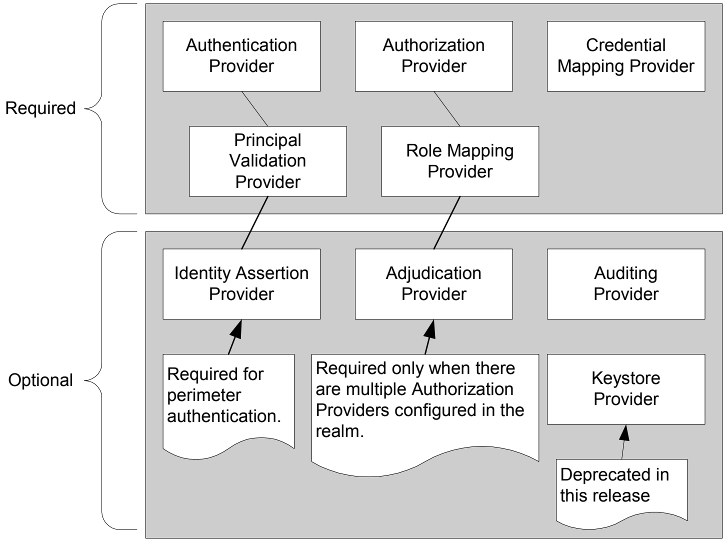
If the WebLogic security providers supplied with the WebLogic Server product do not fully meet your security requirements, you can supplement or replace them with custom security providers. You develop a custom security provider by:

- Implementing the appropriate security service provider interfaces (SSPIs) from the `weblogic.security.spi` package to create runtime classes for the security provider.
- Creating an MBean Definition File (MDF) and using the WebLogic MBeanMaker utility to generate an MBean type, which is used to configure and manage the security provider.

For more information, see *Developing Security Providers for WebLogic Server*.

Figure 4-9 shows the security providers that are required and those that are optional in a WebLogic security realm.

Figure 4-9 WebLogic Security Providers


Required
Authentication Provider
Authorization Provider
Credential Mapping Provider
Principal Validation Provider
Role Mapping Provider
Optional
Identity Assertion Provider
Adjudication Provider
Auditing Provider
Required for perimeter authentication.
Required only when there are multiple Authorization Providers configured in the realm.
Keystore Provider
Deprecated in this release

The WebLogic security providers are described in the following sections:

- "WebLogic Authentication Provider" on page 4-13
- "Alternative Authentication Providers" on page 4-13
- "WebLogic Identity Assertion Provider" on page 4-13
- "Single Pass Negotiate Identity Assertion Provider" on page 4-14
- "WebLogic Principal Validation Provider" on page 4-14
- "WebLogic Authorization Provider" on page 4-15
- "WebLogic Adjudication Provider" on page 4-15
- "WebLogic Role Mapping Provider" on page 4-16
- "WebLogic Auditing Provider" on page 4-16
- "WebLogic Credential Mapping Provider" on page 4-16
- "WebLogic Keystore Provider" on page 4-17
- "WebLogic Realm Adapter Providers" on page 4-17

WebLogic Authentication Provider

The default (active) security realm for WebLogic Server includes a WebLogic Authentication provider. The WebLogic Authentication provider supports delegated username/password and digest authentication. It utilizes an embedded LDAP server to store user and group information. This provider allows you to edit, list, and manage users and group membership.

Note: In conjunction with the WebLogic Authorization provider, the WebLogic Authentication provider replaces the functionality of the File realm that was available in 6.x releases of WebLogic Server.

Alternative Authentication Providers

WebLogic Server provides the following additional Authentication providers which can be used instead of or in conjunction with the WebLogic Authentication provider in the default security realm:

- A set of LDAP Authentication providers that access external LDAP stores (Open LDAP, Netscape iPlanet, Microsoft Active Directory, and Novell NDS).
- An LDAP X509 Identity Assertion provider that looks up the LDAP object for the user associated with an X509 certificate, ensures that the certificate in the LDAP object matches the presented certificate, and then retrieves the name of the user from the LDAP object for the purpose of authentication.

Note: By default, these additional Authentication providers are available but not configured in the WebLogic default security realm.

WebLogic Identity Assertion Provider

The WebLogic Identity Assertion provider supports certificate authentication using X.509 certificates and CORBA Common Secure Interoperability version 2 (CSIv2) identity assertion.

The WebLogic Identity Assertion provider validates the token type, then maps X.509 digital certificates and X.501 distinguished names to WebLogic usernames. It also specifies a list of trusted client principals to use for CSIv2 identity assertion. The wildcard character (*) can be used to specify that all principals are trusted. If a client is not listed as a trusted client principal, the CSIv2 identity assertion fails and the invoke is rejected.

The WebLogic Identity Assertion provider supports the following token types:

- `AU_TYPE`—for a WebLogic `AuthenticatedUser` used as a token.
- `X509_TYPE`—for an X.509 client certificate used as a token.

- CSI_PRINCIPAL_TYPE—for a CSIv2 principal name identity used as a token.
- CSI_ANONYMOUS_TYPE—for a CSIv2 anonymous identity used as a token.
- CSI_X509_CERTCHAIN_TYPE—for a CSIv2 X.509 certificate chain identity used as a token.
- CSI_DISTINGUISHED_NAME_TYPE—for a CSIv2 distinguished name identity used as a token.

Single Pass Negotiate Identity Assertion Provider

The Single Pass Negotiate Identity Assertion provider is used for SSO with Microsoft clients that support the SPNEGO protocol. Specifically, it decodes SPNEGO tokens to obtain Kerberos tokens, validates the Kerberos tokens, and maps Kerberos tokens to WebLogic users. The Single Pass Negotiate Identity Assertion provider utilizes the Java Generic Security Service (GSS) Application Programming Interface (API) to accept the GSS security context via Kerberos. For more information about the Java GSS API, see http://java.sun.com/j2se/1.4.2/docs/guide/security/jgss/jgss-features.html.

The Single Pass Negotiate Identity Assertion provider interacts with the WebLogic Servlet container which handles `WWW-Authenticate` and `WWW-Authorization` headers, adding the appropriate Negotiate header.

By default, the Single Pass Negotiate Identity Assertion provider is available but not configured in the WebLogic default security realm. The Single Pass Negotiate Identity Assertion provider can be used instead of or in addition to the WebLogic Identity Assertion provider.

WebLogic Principal Validation Provider

The default (active) security realm for WebLogic Server includes a WebLogic Principal Validation provider. This provider signs and verifies WebLogic Server principals. In other words, it signs and verifies principals that represent WebLogic Server users or WebLogic Server groups.

Note: You can use the `WLSPrincipals` class (located in the `weblogic.security` package) to determine whether a principal (user or group) has special meaning to WebLogic Server (that is, whether it is a predefined WebLogic Server user or WebLogic Server group). Furthermore, any principal that is going to represent a WebLogic Server user or group needs to implement the `WLSUser` and `WLSGroup` interfaces (available in the `weblogic.security.spi` package).

The WebLogic Principal Validation provider includes implementations of the `WLSUser` and `WLSGroup` interfaces, named `WLSUserImpl` and `WLSGroupImpl`. These are located in the weblogic.security.principal package. It also includes an implementation of the PrincipalValidator SSPI called PrincipalValidatorImpl. For more information about the PrincipalValidator SSPI, see "Implement the PrincipalValidator SSPI" in *Developing Security Providers for WebLogic Server.*

Much as an Identity Assertion provider supports a specific type of token, a Principal Validation provider signs and verifies the authenticity of a specific type of principal. Therefore, you can use the WebLogic Principal Validation provider to sign and verify principals that represent WebLogic Server users or WebLogic Server groups.

WebLogic Authorization Provider

The default (active) security realm for WebLogic Server includes a WebLogic Authorization provider. This provider supplies the default enforcement of authorization for this version of WebLogic Server. Using a policy-based authorization engine, the WebLogic Authorization provider returns an access decision to determine if a particular user is allowed access to a protected WebLogic resource. The WebLogic Authorization provider also supports the deployment and undeployment of security policies within the system.

WebLogic Adjudication Provider

The default (active) security realm for WebLogic Server includes a WebLogic Adjudication provider. This provider would normally be responsible for tallying the potentially differing results rendered by multiple Authorization providers' Access Decisions and rendering a final verdict on whether or not access will be granted to a WebLogic resource. However, because the default security realm only has one Authorization provider, only one Access Decision is produced so the WebLogic Adjudication provider is not used.

Note: The WebLogic Adjudication provider is used in the Compatibility realm, which has two Authorization providers.

The WebLogic Adjudication provider has an attribute called Require Unanimous Permit that governs its behavior. By default, the Require Unanimous Permit attribute is set to TRUE, which causes the WebLogic Adjudication provider to act as follows:

- If all the Authorization providers' Access Decisions return PERMIT, then return a final verdict of TRUE (that is, permit access to the WebLogic resource).
- If some Authorization providers' Access Decisions return PERMIT and others return ABSTAIN, then return a final verdict of FALSE (that is, deny access to the WebLogic resource).

- If any of the Authorization providers' Access Decisions return `ABSTAIN` or `DENY`, then return a final verdict of `FALSE` (that is, deny access to the WebLogic resource).

If you change the Require Unanimous Permit attribute to `FALSE`, the WebLogic Adjudication provider acts as follows:

- If all the Authorization providers' Access Decisions return `PERMIT`, then return a final verdict of `TRUE` (that is, permit access to the WebLogic resource).
- If some Authorization providers' Access Decisions return `PERMIT` and others return `ABSTAIN`, then return a final verdict of `TRUE` (that is, permit access to the WebLogic resource).
- If any of the Authorization providers' Access Decisions return `DENY`, then return a final verdict of `FALSE` (that is, deny access to the WebLogic resource).

Note: You set the Require Unanimous Permit attributes when you configure the WebLogic Adjudication provider. For more information about configuring an Adjudication provider, see "Configuring a WebLogic Adjudication Provider" in *Managing WebLogic Security*.

WebLogic Role Mapping Provider

The default (active) security realm for WebLogic Server includes a WebLogic Role Mapping provider. This provider determines dynamic roles for a specific user (subject) with respect to a specific protected WebLogic resource for each of the default users and WebLogic resources. The WebLogic Role Mapping provider supports the deployment and undeployment of roles within the system. The WebLogic Role Mapping provider uses the same security policy engine as the WebLogic Authorization provider.

WebLogic Auditing Provider

The default (active) security realm for WebLogic Server includes a WebLogic Auditing provider. This provider records information from a number of security requests, which are determined internally by the WebLogic Security Framework. The WebLogic Auditing provider also records the event data associated with these security requests, and the outcome of the requests.

WebLogic Credential Mapping Provider

The default (active) security realm for WebLogic Server includes a WebLogic Credential Mapping provider. You use the WebLogic Credential Mapping provider to associate, or map, a WebLogic Server user to the appropriate credentials to be used with a Resource Adapter to access an Enterprise Information System (EIS), for example, some relational database like Oracle, SQL

Server, and so on. The provider maps a user's authentication credentials (username and password) to those required for legacy applications, so that the legacy application gets the necessary credential information. For example, the EIS may be a mainframe transaction processing, database systems, or legacy applications not written in the Java programming language.

If you only want to map WebLogic Server users and groups to username/password credentials in another system, then the WebLogic Credential Mapping provider is sufficient.

WebLogic Keystore Provider

The WebLogic Keystore provider uses the reference keystore implementation supplied by Sun Microsystems in the Java Software Development Kit (SDK). It utilizes the standard Java KeyStore (JKS) keystore type, which implements the keystore as a file (one per machine). It protects each private key with its individual password. There are two keystore files associated with the WebLogic Keystore provider:

- One keystore file holds the trusted certificate authority (CA) certificates. WebLogic Server also ships a trusted certificate authority Keystore file that it uses by default to locate the trusted CA certificates, which are then used in SSL to verify client certificates.
- The other keystore file holds the server's private keys. WebLogic Server retrieves a private key from this file to initialize SSL. You can use the Sun Microsystems SDK `keytool` utility or the WebLogic Server `ImportPrivateKey` utility to add private keys to this file. Note that WebLogic Server can retrieve private keys and certificates from this keystore file.

Note: The WebLogic Keystore provider is deprecated in this release of WebLogic Server but is still supported. The development of custom Keystore providers is not supported. Use Java KeyStores (JKS) instead. All of the functionality that was supported by the WebLogic Keystore provider is available through use of Java KeyStores. The WebLogic Keystore provider is only supported for backward compatibility. BEA recommends using the WebLogic Keystore provider only when it is needed to support backward compatibility with a WebLogic Server 7.0 configuration. For information on how to use Java KeyStores, see Configuring Keystores in *Managing WebLogic Server*.

WebLogic Realm Adapter Providers

The WebLogic Realm Adapter providers provide backward-compatibility with 6.x WebLogic security realms by allowing the use of existing, 6.x security realms with the security features in this release of WebLogic Server. The WebLogic Realm Adapter providers map the realm API (`weblogic.security.acl`) used in WebLogic Server 6.x to the APIs used in this release of WebLogic Server. The following WebLogic Realm Adapter providers are provided:

- Authentication (includes an Identity Assertion provider)

  The Realm Adapter Authentication provider allows you to use version 6.x security realms and their data stores with the WebLogic security providers in WebLogic Server. The Realm Adapter Authentication provider also allows you to use implementations of the `weblogic.security.acl.CertAuthenticator` class with WebLogic Server. The Realm Adapter Authentication provider includes a component that provides identity assertion based on X.509 tokens.

- Authorization

  In Compatibility security, two types of Authorization providers are used: the WebLogic Authorization provider and the Realm Adapter Authorization provider. The WebLogic Authorization provider is used for new security policies. The Realm Adapter Authorization provider is used for mapping to WebLogic Server 6.1 access control lists (ACLs).

- Auditing

  The Realm Adapter Auditing provider allows you to use implementations of the `weblogic.security.audit` interface with WebLogic Server deployments using Compatibility security.

- Adjudication

  The Realm Adapter Adjudication provider enables both the WebLogic Authorization provider and the Realm Adapter Authorization provider to be used together for a security realm in Compatibility security.

Although these security providers are configured using the WebLogic Server Administration Console, your existing 6.x security realms will continue to use the same MBeans and user interface present in WebLogic Server 6.1.

Note: The WebLogic Realm Adapter providers are deprecated and should only be used while upgrading to the WebLogic Server 8.1 security model.

How the Architecture Benefits Users

The WebLogic security service architecture provides specific benefits for the following categories of users:

- "Application Developers" on page 4-19

- “Server/Application Administrators” on page 4-19
- “Third-Party Security Service Providers” on page 4-19

Application Developers

Since most security for Web applications and EJBs can be implemented by a system administrator, application developers need not pay attention to the details of securing the application, unless there are special considerations that must be addressed in the code. For programming custom security into an application, WebLogic Server application developers can take advantage of BEA-supplied Application Programming Interfaces (APIs) for obtaining information about subjects and principals (identifying information for users) that are used by WebLogic Server. These APIs are found in the `weblogic.security` package.

With WebLogic Server’s comprehensive support for the Java standards, developers of applications for WebLogic Server can also use the APIs in the Java platform security packages such as JAAS and JSSE, as well as the security-specific methods defined by J2EE.

Server/Application Administrators

Out of the box, administrators can use WebLogic Sever security providers to implement a complete security solution. Administrators can use the Administration Console to define security roles and assign security policies to WebLogic resources so as to create an authorization scheme that implements your company’s business rules.

Third-Party Security Service Providers

Most industry leading, security service providers have announced plans to support BEA WebLogic Server 8.1. These third-party providers are integrating their products with the WebLogic Server environment using the Security Service Provider Interfaces (SSPIs). As the underlying integration mechanism for WebLogic security providers, the SSPIs allow development of customized security providers for the WebLogic Server environment. SSPIs are available for Adjudication, Auditing, Authentication, Authorization, Credential Mapping, Identity Assertion, and Role Mapping.

Note: The SSPI for Keystore providers is deprecated in this release of WebLogic Server. Use Java KeyStores (JKS) instead. For information on how to use Java KeyStores, see Configuring Keystores in *Managing WebLogic Server*.

This architecture allows security developers to provide tightly integrated solutions that are easy to implement. The result is a reduction in development requirements, which means an increased return on investment when implementing an enterprise security management solution.

For additional information, see the BEA Security Center at http:// www.bea.com/security.

Terminology

Key terms that you will encounter throughout the WebLogic Server security documentation include the following:

access control list (ACL)

In WebLogic 6.x, a data structure used to control access to computer resources. Each entry on the access control list (ACL) contains a set of permissions associated with a particular principal that represents an individual user or a group of users. Entries can be positive or negative. An entry is positive if it grants permission and negative if it denies permission. In WebLogic Server 7.0 and later, ACLs are deprecated and are replaced by security policies. To continue to protect WebLogic resources with ACLs, use Compatibility security. See also Compatibility security, group, principal, security policy, user, WebLogic resource.

Access Decision

Code that determines whether a subject has permission to perform a given operation on a WebLogic resource. The result of an Access Decision is to permit, deny, or abstain from making a decision. An Access Decision is a component of an Authorization provider. See also Authorization provider, subject, WebLogic resource.

ACL

See access control list (ACL).

Adjudication provider and Adjudicator

A WebLogic security provider that tallies the results that multiple Access Decisions return, resolves conflicts between the Access Decisions, and determines the final `PERMIT` or `DENY` decision. The Adjudicator is a component of the Adjudication provider. See also Access Decision, security provider.

asymmetric key cryptography

A key-based cryptography that uses an encryption algorithm in which different keys, private and public, are used to encrypt and decrypt the data. Data that is encrypted with the public key can be decrypted only with the private key. This asymmetry is the property that makes public key cryptography so useful. Asymmetric key cryptography is also called public key cryptography. See also private key, public key, symmetric key cryptography.

auditing

Process whereby information about operating requests and the outcome of those requests is collected, stored, and distributed for the purposes of non-repudiation. Auditing provides an electronic trail of computer activity. See also Auditing provider.

Auditing provider

A security provider that provides auditing services. See also auditing, security provider.

authentication

Process whereby the identity of users or system processes are proved or verified. Authentication also involves remembering, transporting, and making identity information available to various components of a system when that information is needed. Authentication typically involves username/password combinations, but can also be done using tokens. See also Authentication provider, Identity Assertion, LoginModule, perimeter authentication, token, user.

Authentication provider

A security provider that enables WebLogic Server to establish trust by validating a user. The WebLogic Security Service architecture supports Authentication providers that perform username/password authentication; certificate-based authentication directly with WebLogic Server; and HTTP certificate-based authentication proxied through an external Web server. See also authentication, digital certificate, security provider, user.

authorization

Process whereby a user’s access to a WebLogic resource is permitted or denied based on the user’s security role and the security policy assigned to the requested WebLogic resource. See also Authorization provider, security policy, user, WebLogic resource.

Authorization provider

A security provider that controls access to WebLogic resources based on the user’s security role and the security policy assigned to the requested WebLogic resource. See also security provider, user, WebLogic resource.

Caching realm

A WebLogic Server 6.x feature that applies to WebLogic Server 7.0 and later only if you use Compatibility security. A Caching realm is a temporary location in memory that contains frequently called ACLs, users, groups, and so on, from the primary realm. In WebLogic Server 6.x, users, groups, and ACL objects are stored in the `filerealm.properties` file, and reading from a file can be very slow. The Caching realm is a communication layer on top of the primary realm and is used for lookups, by default. If the Caching realm lookup fails, a lookup is performed on the primary realm. See also access control list (ACL), Compatibility security, group, user.

certificate

See digital certificate.

certificate authentication

Method of providing a confident identification of a client by a server through the use of digital certificates. Certificate authentication is generally preferred over password authentication because it is based on what the user has (a private key), as well as what the user knows (a password that protects the private key). See also authentication, certificate authority, certificate.

certificate authority

A trusted entity that issues public key certificates. A certificate authority attests to a user's real-world identity, much as a notary public does. See also certificate chain, digital certificate, entity, private key, public key, trusted (root) certificate authority.

certificate chain

An array that contains a private key, the matching public key, and a chain of digital certificates for trusted certificate authorities, each of which is the issuer of the previous digital certificate. The certificate for the server, authority, authority2, and authority3, constitute a chain, where the server certificate is signed by the authority, the authority's certificate is signed by authority2, and authority2's certificate is signed by authority3. If the certificate authority for any of these authorities is recognized by the client, the client authenticates the server. See also trusted (root) certificate authority.

Compatibility realm

Security realm that is the default (active) security realm if you are using Compatibility security. The Compatibility realm adapts your existing WebLogic Server 6.x Authentication and Authorization providers so that you can use them in WebLogic Server 7.x or later. The only security realm available in Compatibility security is the Compatibility realm. See also Compatibility security, default realm, security provider, security realm, WebLogic security provider.

Compatibility security

The capability to run security configurations from WebLogic Server 6.x in later releases of WebLogic Server. Using Compatibility security in WebLogic Server 7.x or later, you configure 6.x security realms; define users, groups, and ACLs; manage protection of user accounts; and install custom auditing providers. The only security realm available in Compatibility security is the Compatibility realm. The Realm Adapter providers in the Compatibility realm allow backward compatibility to the authentication and authorization services in 6.x security realms. See also access control list (ACL), Auditing provider, Compatibility realm, group, Realm Adapter Authentication provider, Realm Adapter Authorization provider, security realm, user.

connection filter

A programmable filter that WebLogic Server uses to determine whether the server should allow incoming connections from a network client. In addition to security policies that protect WebLogic resources based on user characteristics, you can add another layer of security by filtering based on network connections. See also security policy, user, WebLogic resource.

connector

See resource adapter

context handler

A ContextHandler is a high-performing WebLogic 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 WebLogic resource container to pass additional information to a WebLogic 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 WebLogic resource container and the security provider.) See also security provider, WebLogic container, WebLogic Security Framework.

credential

Security-related attribute of a subject, which may contain information used to authenticate the subject to new services. Types of credentials include username/password combinations, Kerberos tickets, and public key certificates. See also credential mapping, Credential Mapping provider, digital certificate, Kerberos ticket, public key, subject.

credential mapping

The process whereby a legacy system's database is used to obtain an appropriate set of credentials to authenticate users to a target resource. WebLogic Server uses credential mapping to map credentials used by WebLogic Server users to credentials used in a legacy (or any remote) system. WebLogic Server then uses the credential maps to log in to a remote system on behalf of a subject that has already been authenticated. See also credential, Credential Mapping provider, resource.

Credential Mapping provider

A security provider that is used to provide credential mapping services and bring new types of credentials into the WebLogic Server environment. See also credential, credential mapping, security provider.

Cross-Domain Single Sign-on

WebLogic Server security feature that allows users to authenticate once but access multiple applications, even if these applications reside in different DNS domains. You can use this feature to construct a network of affiliates or partners that participate in a Single Sign-On domain. See also single sign-on (SSO).

Note: Cross-domain single sign-on is only supported for Java clients, that is, clients that are running a Java Virtual Machine (JVM); Cross-domain single sign-on is not supported with Web browser clients.

CSIv2 protocol

A protocol that is based on IIOP (GIOP 1.2) and the CORBA Common Secure Interoperability version 2 (CSIv2) CORBA specification. The secure interoperability requirements for EJB2.0 and other J2EE1.4.1 containers correspond to Conformance Level 0 of the CSIv2 specification. The CORBA Security Attribute Service (SAS) is the protocol that is used in CSIv2. For more information, see `http://www.omg.org/technology/documents/formal/omg_security.htm`.

custom security provider

Security provider written by third-party security vendors or security developers that can be integrated into the WebLogic Security Service. Custom security providers are implementations of the Security Service Provider Interfaces (SSPIs) and are *not* supplied with the WebLogic Server product. See also security provider, security realm, WebLogic security provider, WebLogic Security Service.

Custom security realm

In WebLogic Server 7.0 and later, supported only in Compatibility security. In WebLogic Sever 6.x, you customize authentication by creating your own security realm and integrating it into the WebLogic Server environment. See also Compatibility security.

database delegator

Intermediary class that mediates initialization calls between a security provider and the security provider's database. See also security provider database.

declarative security

Security that is defined, or declared, using the application deployment descriptors. For Web applications, you define the deployment descriptors in the `web.xml` and `weblogic.xml` files. For EJBs, you define the deployment descriptors in the `ejb-jar.xml` and `weblogic-ejb-jar.xml` files.

default realm

The active security realm. In WebLogic Server 7.0 and later, you can configure multiple security realms in a WebLogic Server domain; however, only one can be the default (active) security realm. See also Custom security realm, security realm,WebLogic Server domain.

digital certificate

Digital statement that associates a particular public key with a name or other attributes. The statement is digitally signed by a certificate authority. By trusting that authority to sign only true statements, you can trust that the public key belongs to the person named in the certificate. See also certificate authority, digital signature, public key, trusted (root) certificate authority.

digital signature

String of bits used to protect the security of data being exchanged between two entities by verifying the identities of those entities. Specifically, this string is used to verify that the data came from the sending entity of record and was not modified in transit. A digital signature is computed from an entity's signed data and private key. It can be trusted only to the extent that the public key used to verify it can be trusted. See also entity, private key, public key.

Domain Configuration Wizard

An interactive, graphical user interface (GUI) that facilitates the creation of a new WebLogic Server domain. The wizard can create WebLogic Server domain configurations for stand-alone servers, Administration Servers with Node Managers and Managed Servers, and clustered servers. You can use it to create the appropriate directory structure for your WebLogic Server domain, a basic `config.xml` file, and scripts that you can use to start the servers in your domain.

embedded LDAP server

A server that contains user, group, security role, security policy and credential information. The WebLogic Authentication, Authorization, Role Mapping, and

Credential Mapping providers use the embedded LDAP server as their security provider databases. See also credential, group, security policy, security role.

entity

Something that exists independently as a particular and discrete unit. Persons, corporations, and objects are examples of entities.

File realm

In WebLogic Server 6.x, a realm that stores users, groups, encrypted passwords, and ACLs in a file. In WebLogic Server 7.0 and later you use a File realm only with Compatibility security. See also Compatibility security.

firewall

Software that monitors traffic between an internal network and the Internet, and that regulates the type of network traffic that can enter and leave the internal network. A firewall can be connected to the Internet or set up within a company's network to prevent unauthorized access to the network. Firewalls protect information on computers and information that is being carried over the network. Firewalls use various types of filters to prevent access, including limiting the types of protocols allowed and restricting access from network nodes by IP addresses and DNS node names.

global role

A security role that applies to all WebLogic resources within a security realm. For example, if the WebLogic Role Mapping provider is being used in the default security realm, global roles can be defined in terms of user, group, and hours of access. See also Role Mapping provider, scoped role, security realm, security role, WebLogic resource.

group

Collection of users that share some characteristic, such as a department, a job function, or a job title. Groups are a static identity that a server administrator assigns. Groups are associated with security roles. Giving permission to a group is the same as giving the permission to each user who is a member of the group. See also user.

host name verification

The process of verifying that the name of the host to which an SSL connection is made is the intended or authorized party. See also Host Name Verifier, Secure Sockets Layer (SSL).

Host Name Verifier

Code that validates that the host to which an SSL connection is made is the intended or authorized party. A Host Name Verifier is useful when a WebLogic Server client or a WebLogic Server instance acts as an SSL client to another application server. It helps prevent man-in-the-middle attacks. By default, WebLogic Server, as a function of the SSL handshake, compares the common name in the subject distinguished name (DN) of the SSL server's digital certificate with the host name of the SSL server used to initiate the SSL connection. If the subject DN and the host name do not match, the SSL connection is dropped. See also digital certificate, host name verification, Secure Sockets Layer (SSL), subject.

Identity Assertion

Special type of authentication whereby a client's identity is established through the use of client-supplied tokens that are generated from an outside source. Identity is asserted when these tokens are mapped to usernames. For example, the client's identity can be established by using a digital certificate, and that certificate can be passed around the system so that users are not asked to sign on more than once. Thus, identity assertion can be used to enable single sign-on. See also authentication, digital certificate, Identity Assertion provider, single sign-on (SSO), SSL tunneling, token.

Identity Assertion provider

A security provider that performs perimeter authentication—a special type of authentication using tokens. Identity Assertion providers also allow WebLogic Server to establish trust by validating a user. Thus, the function of an Identity Assertion provider is to validate and map a token to a username. See also perimeter authentication, security provider, token, user.

JAAS control flag

If a security realm has multiple Authentication providers configured, the JAAS control flag determines how the login sequence uses the Authentication providers. See also Authentication provider.

JAAS LoginModule

Responsible for authenticating users within the security realm and for populating a subject with the necessary principals (users/groups). A LoginModule is a required component of an Authentication provider, and can be a component of an Identity Assertion provider if you want to develop a separate LoginModule for perimeter authentication. LoginModules that are not used for perimeter authentication also verify the proof material submitted (for example, a user's password). See also authentication, group, Identity Assertion provider, perimeter authentication, principal, security realm, subject.

Java Authentication and Authorization Service (JAAS)

Set of Java packages that enable services to authenticate and enforce access controls upon users. JAAS implements a Java version of the standard Pluggable Authentication Module (PAM) framework, and supports user-based authorization. WebLogic Server only implements the authentication portion of JAAS. See also authentication, authorization, user.

Java Cryptography Architecture

A framework for accessing and developing cryptographic functionality for the Java platform. For a description of the Java Cryptography Architecture provided by Sun Microsystems, Inc., see `http://java.sun.com/j2se/1.4/docs/guide/security/CryptoSpec.html#Introduction`. See also Java Cryptography Extensions (JCE)

Java Cryptography Extensions (JCE)

Set of Java packages that extends the Java Cryptography Architecture API to include APIs for encryption, key exchange, and Message Authentication Code (MAC) algorithms. See `http://java.sun.com/j2se/1.4/docs/guide/security/jce/JCERefGuide.html` for a description of JCE provided by Sun Microsystems, Inc. See also Java Cryptography Architecture.

Java Naming and Directory Interface (JNDI)

The Java Naming and Directory Interface (JNDI) is an application programming interface (API) that provides naming services to Java applications. JNDI is an integral component of the Sun Microsystems J2EE technology and is defined to be independent of any specific naming or directory service implementation. It supports the use of a single method for accessing various new and existing services. This support allows any service-provider implementation to be plugged into the JNDI framework using the standard service provider interface (SPI) conventions. In addition, JNDI allows Java applications in WebLogic Server to access external directory services such as LDAP in a standardized fashion, by plugging in the appropriate service provider.

Java Security Manager

Security manager for the Java virtual machine (JVM). The Java Security Manager works with the Java API to define security boundaries through the `java.lang.SecurityManager` class, thus, enabling developers to establish a custom security policy for their Java applications.

WebLogic Server supports the use of the Java Security Manager to prevent untrusted code from performing actions that are restricted by the Java security policy file. The Java Security Manager uses the Java security policy file to enforce a set of permissions granted to classes. The permissions allow specified classes running in that instance of the JVM to permit or deny certain runtime operations. See also Java security policy file, policy condition.

Java security policy file

File used by the Java Security Manager to enforce a set of permissions granted to specified classes running in an instance of the WebLogic Server-supported Java Virtual Machine (JVM). Classes running in that instance of the JVM use the permissions to permit or deny certain runtime operations. See also Java Security Manager, policy condition.

JNDI

See JNDI.

Kerberos ticket

A sequence of a few hundred bytes in length that is used to control access to physically insecure networks. Kerberos tickets are based on the Kerberos protocol. Kerberos is a network authentication protocol that allows entities (users and services) communicating over networks to prove their identity to each other, while preventing eavesdropping or replay attacks. The protocol was designed to provide strong authentication for client/server applications by using secret-key cryptography. For more information, see `http://web.mit.edu/kerberos/www/`. See also private key.

keystore

An in-memory collection of private key and trusted certificate pairs. The information is protected by a passphrase, such as a password, a credit card number, Personal Identification Number, or some other form of personal identification information. In the Administration Console, the keystore is referred to as the Trusted Keystore. For more information, see SDK 1.4.1 Javadoc produced by Sun Microsystems, Inc., which is available at `http://java.sun.com/j2se/1.4/docs/api/index.html`. See also private key, trusted (root) certificate authority.

LDAP Authentication provider

Authentication provider that uses a Lightweight Data Access Protocol (LDAP) server to access user and group information, for example, iPlanet's Active Directory and Novell's OpenLDAP. See also group, user.

LDAP security realm

A WebLogic Server 6.x security realm. In WebLogic Server 6.x, security realms provide authentication and authorization services. The LDAP security realm provides authentication through an LDAP server. This server allows you to manage all the users for your organization in one place: the LDAP directory. The LDAP security realm supports Open LDAP, Netscape iPlanet, Microsoft Site Server, and Novell NDS. In WebLogic Server 7.x or later, you can only use the LDAP security realm when using Compatibility security. See also authentication, authorization, Compatibility security, File realm, security realm, user.

LoginModule

See JAAS LoginModule.

MBean

Short for "managed bean," a Java object that represents a Java Management eXtensions (JMX) manageable resource. MBeans are instances of MBean types. MBeans are used to configure and manage security providers. See also MBean type, security provider.

MBean Definition File (MDF)

An XML file used by the WebLogic MBeanMaker to generate files for an MBean type. See also MBean type, WebLogic MBeanMaker.

MBean implementation file

One of several intermediate Java files generated by the WebLogic MBeanMaker utility to create an MBean type for a custom security provider. You edit this file to supply your specific method implementations. See also MBean information file, MBean interface file, MBean type, WebLogic MBeanMaker.

MBean information file

One of several intermediate Java files generated by the WebLogic MBeanMaker utility to create an MBean type for a custom security provider. This file contains mostly metadata and therefore requires no editing. See also MBean implementation file, MBean interface file, MBean type, WebLogic MBeanMaker.

MBean interface file

One of several intermediate Java files generated by the WebLogic MBeanMaker utility to create an MBean type for a custom security provider. This file is the client-side API to the MBean that your runtime class or your MBean implementation will use to obtain configuration data, and requires no editing. See also MBean implementation file, MBean information file, MBean type, runtime class, WebLogic MBeanMaker.

MBean JAR File (MJF)

JAR file that contains the runtime classes and MBean types for a security provider. MJFs are created by the WebLogic MBeanMaker. See also MBean type, runtime class, security provider, WebLogic MBeanMaker.

MBean type

Factory for creating the MBeans used to configure and manage security providers. MBean types are created by the WebLogic MBeanMaker. See also MBean, security provider, WebLogic MBeanMaker.

message digest

A digitally created hash, or fingerprint, created from a block of plain text. Even though the complete message is used to create the hash, the message cannot be recreated from the hash. Message digests help prevent man-in-the-middle attacks. Because there is only one digest for any given block of plain text, the digest can be used to verify the authenticity of the message. Thus, this process results in a digital signature of the message, which can be used to provide non-repudiation and integrity services. See also message digest algorithm.

message digest algorithm

A computational procedure that is used to produce a message digest from a block of plain text. Once a message digest is produced, other security mechanisms are used to encrypt and convey the digest. See also message digest.

mutual authentication

Authentication that requires both client and server to present proof of identity. Two-way SSL authentication is a form of mutual authentication in that both client and server present digital certificates to prove their identity. However, with two-way SSL, the authentication happens at the SSL level, whereas other forms of mutual authentication are executed at higher levels in the protocol stack. See also authentication, digital certificate, Secure Sockets Layer (SSL), two-way SSL authentication, trusted (root) certificate authority.

non-repudiation

Irrefutable evidence that a security event occurred.

Negotiate Identity Assertion provider

Identity Assertion provider that decodes Simple and Protected Negotiate (SPNEGO) tokens to obtain Kerberos tokens, validates the Kerberos tokens, and maps Kerberos tokens to WebLogic uses for the purpose of single sign-on. See also Simple and Protected Negotiation Mechanism (SPNEGO) and single sign-on (SSO) with Desktop clients.

one-way SSL authentication

Type of SSL authentication which requires the server to present a certificate to the client, but the client is not required to present a certificate to the server. The client must authenticate the server, but the server will accept any client into the connection. Enabled by default in WebLogic Server. See also mutual authentication, two-way SSL authentication.

perimeter authentication

Authentication that occurs outside the application server domain. Perimeter authentication is typically accomplished when a remote user specifies an asserted identity and some form of corresponding proof material, normally in the form of a passphrase (such as a password, a credit card number, Personal Identification Number, or some other form of personal identification information.), to an authentication server (typically a Web server) that performs the verification and then passes an artifact, or token, to the application server domain (for example, a WebLogic Server domain). The application server can then pass the token around to systems in the domain so that users are not asked to sign on more than once.

The authentication agent, the entity that actually vouches for the identity, can take many forms, such as a Virtual Private Network (VPN), a firewall, an enterprise authentication service (Web server), or some other form of global identity service.

The WebLogic Server security architecture supports Identity Assertion providers that perform perimeter authentication (Web server, firewall, VPN) and handle multiple security token types and protocols (SOAP, IIOP-CSIv2). See also authentication, Identity Assertion.

policy condition
A condition under which a security policy will be created. Policy conditions, along with the specific information you supply for the condition (such as an actual user name, group, security role, or start/stop times), are called expressions. See also policy statement.

policy expression
See policy statement.

policy statement
A policy statement is the collection of expressions that define who is granted access to a WebLogic resource, and is therefore the main part of any security policy you create. Policy statements are also referred to as policy expressions. See also policy condition.

principal
The identity assigned to a user, group, or system process as a result of authentication. A principal can consist of any number of users and groups. Principals are typically stored within subjects. See also authentication, group, subject, user.

principal validation
The act of signing and later verifying that a principal has not been altered since it was signed. Principal validation establishes trust of principals. See also principal.

private key
An encryption/decryption key known only to the party or parties that exchange secret messages. It is called private because it must be kept secret from everyone but the owner. See also public key.

private key algorithm

The computational procedure used to encode, or encrypt, ciphertext. Data encrypted with the private key can only be decrypted by the public key. See also private key, public key, RDBMS security realm.

programmatic security

Application security that is defined in servlets and EJBs using Java methods.

public key

Value provided by a certificate authority as an encryption/decryption key that, combined with a private key, can be used to effectively encrypt and decrypt messages and digital signatures. The key is called public because it can be made available to anyone. Public key cryptography is also called asymmetric cryptography because different keys are used to encrypt and decrypt the data. See also asymmetric key cryptography, private key.

public key algorithm

The computational procedure used to encode, or encrypt, plain text. Data encrypted with the public key can only be decrypted by the private key. See also private key, private key algorithm, public key.

public key cryptography

See asymmetric key cryptography.

RDBMS security realm

A WebLogic Server 6.x security realm. In WebLogic Server 6.x, security realms provided authentication and authorization services. The RDBMS security realm stores Users, Groups, and ACLs in a relational database. In WebLogic Server 7.0 and later, you can only use the RDMS security realm when using Compatibility security. See also access control list (ACL), authentication, authorization, Compatibility security, group, security realm, user.

Realm Adapter Adjudication provider

The Realm Adapter Adjudication provider enables both the WebLogic Authorization provider and the Realm Adapter Authorization provider to be used together for a security realm in Compatibility security. See also Compatibility security, Compatibility realm.

Realm Adapter Auditing provider

Auditing provider in the CompatibilityRealm that allows you to use implementations of the `weblogic.security.audit` interface with WebLogic Server deployments using Compatibility security. You must run Compatibility security in order to access the Compatibility realm and the Realm Adapter providers through the WebLogic Server Administration Console. See also Compatibility security, Compatibility realm.

Realm Adapter Authentication provider

Authentication provider in the Compatibility realm that allows backward compatibility to the authentication services in 6.x security realms. You must run Compatibility security in order to access the Compatibility realm and the Realm Adapter providers through the WebLogic Server Administration Console. See also Compatibility security, Compatibility realm.

Realm Adapter Authorization provider

Authorization provider in the Compatibility realm that allows backward compatibility to the authorization services in 6.x security realms. You must run Compatibility security in order to access the Compatibility realm and the Realm Adapter providers through the WebLogic Server Administration Console. See also Compatibility security, Compatibility realm.

Realm Adapter provider

Type of security provider used to access WebLogic Server 6.x security services when using Compatibility security in WebLogic Server 7.0 or later. These providers allow you to adapt 6.x security providers so that they can be used with WebLogic Server 7.0 and later. You must run Compatibility security in order to access the Compatibility realm and the Realm Adapter providers through the WebLogic Server Administration Console. See also Compatibility security, Compatibility realm.

resource

See WebLogic resource.

resource adapter

System-level software driver (also called a connector) used by an application server (such as WebLogic Server) or an application client to connect to an enterprise information system (EIS). Resource adapters contain the Java components and, if necessary, the native components required to interact with the EIS.

The WebLogic J2EE Connector Architecture supports resource adapters developed by EIS vendors and third-party application developers that can be deployed in any application server supporting the Sun Microsystems J2EE Platform Specification, version 1.3.

role condition

A condition under which a security role (global or scoped) will be granted to a user or group. Role conditions, along with the specific information you supply when creating the condition (such as an actual user name, group, or start/stop times), are called expressions. See security policy, role mapping.

role expression

Specific information that you supply when creating role conditions. See role condition.

role mapping

Process by which the WebLogic Security Service compares users or groups against a security role condition to determine whether they should be dynamically granted a security role. Role mapping occurs at runtime, just prior to when an Access Decision is rendered for a protected WebLogic resource. See also Access Decision, group, principal, role condition, security role, user, WebLogic resource, WebLogic Security Service.

Role Mapping provider

A security provider that determines what security roles apply to the principals stored in a subject when the subject is attempting to perform an operation on a WebLogic resource. Because this operation usually involves gaining access to the WebLogic resource, Role Mapping providers are typically used with Authorization providers. See also Authorization provider, principal, security role, subject, WebLogic resource.

role statement

A collection of expressions that define how a security role is granted, and is therefore the main part of any security role you create. See role expression.

runtime class

Java class that implements a Security Service Provider Interface (SSPI) and contains the actual security-related behavior for a security provider. See also security provider, Security Service Provider Interfaces (SSPIs).

scoped role

A security role that applies to a specific WebLogic resource in a security realm. See also global role, Role Mapping provider, security role, security realm.

secret key cryptography

See symmetric key cryptography.

Secure Sockets Layer (SSL)

An Internet transport-level technology developed by Netscape to provide data privacy between applications. Generally, Secure Sockets Layer (SSL) provides (1) a mechanism that the applications can use to authenticate each other's identity and (2) encryption of the data exchanged by the applications. SSL supports the use of public key cryptography for authentication, and secret key cryptography and digital signatures to provide privacy and data integrity. See also authentication, digital signature, public key cryptography, symmetric key cryptography.

Security Assertion Markup Language (SAML)

An XML-based framework for exchanging security information. SAML implementations provide an interoperable, XML-based, security solution that allows authentication and authorization information to be exchanged securely. SAML is the key to enabling single sign-on capabilities for Web services. For more information, see `http://xml.coverpages.org/saml.html`.

You can develop custom Identity Assertion providers for WebLogic Server that support different token types, including SAML. See also authentication, authorization, Identity Assertion, perimeter authentication, Cross-Domain Single Sign-on, user.

security policy

An association between a WebLogic resource and a user, group, or security role that protects the WebLogic resource against unauthorized access. A WebLogic resource has no protection until you assign it a security policy. You can assign security policies to an individual WebLogic resource or to components of the WebLogic resource.

In WebLogic Server 7.0 and later, security policies replace access control lists (ACLs), except when Compatibility security is used. See also access control list (ACL), group, security role, user, WebLogic resource.

security provider

In WebLogic Server 7.0 and later, software modules that can be "plugged into" a WebLogic Server security realm to provide security services (such as authentication, authorization, auditing, and credential mapping) to applications. A security provider consists of runtime classes and MBeans, which are created from SSPIs and MBean types, respectively. Security providers are WebLogic security providers (provided with WebLogic Server) or custom security providers. See also custom security provider, MBean, MBean type, runtime class, Security Service Provider Interfaces (SSPIs), WebLogic security provider.

security provider database

Database that contains the users, groups, security policies, roles, and credentials used by some types of security providers to provide security services. The security provider database can be the embedded LDAP server (as used by the WebLogic security providers), a properties file (as used by the sample security providers), or a production-quality database that you may already be using. See also credential, embedded LDAP server, group, security role, security policy, WebLogic security provider.

security realm

In WebLogic Server 6.x, security realms provide authentication and authorization services. You use the File realm or a set of alternative security realms, including the

Lightweight Data Access Protocol (LDAP), Windows NT, Unix, or RDBMS realms. If you want to customize authentication, you write your own security realm and integrate it into the WebLogic Server environment. In WebLogic Server 6.x you cannot have multiple security realms in a domain. See also File realm.

In WebLogic Server 7.0 and later, security realms act as a scoping mechanism. Each security realm consists of a set of configured security providers, users, groups, roles, and security policies. You can configure multiple security realms in a domain; however, only one can be the default (active) security realm. WebLogic Server provides two default security realms: myrealm and Compatibility realm. You can access an existing 6.x security configuration through the Compatibility realm. You can no longer write a custom security realm using the application programming interfaces as you could in WebLogic Server 6.x; rather, you configure a new security realm (called myrealm by default) to provide the security services you want and then set the new security realm as the default security realm. See also Compatibility realm, Custom security realm, default realm, Domain Configuration Wizard, security provider, WebLogic resource.

security role

A dynamically computed privilege that is granted to users or groups based on specific conditions. The difference between groups and roles is that a group is a static identity that a server administrator assigns, while membership in a role is dynamically calculated based on data such as user name, group membership, or the time of day. Security roles are granted to individual users or to groups, and multiple roles can be used to create security policies for a WebLogic resource. Once you create a security role, you define an association between the role and a WebLogic resource. This association (called a security policy) specifies who has what access to the WebLogic resource. See also global role, group, role mapping, scoped role, security policy, user, WebLogic resource.

Security Service Provider Interfaces (SSPIs)

Set of WebLogic packages that enables custom security providers to be developed and integrated with the WebLogic Server Security Service. These interfaces are implemented by the WebLogic security providers and custom security providers. The WebLogic Security Framework calls methods in these interfaces to perform security operations. See also security provider, WebLogic Security Framework.

Simple and Protected Negotiation Mechanism (SPNEGO)

Allows participation in a Kerberos-based Single Sign-On (SSO) environment. The SPNEGO protocol performs a Kerberos authentication via HTTP, and allows Internet Explorer to pass a delegated credential to allow a web application to log in to subsequent Kerberized services on the user's behalf. See also Negotiate Identity Assertion provider, single sign-on (SSO) with Desktop clients.

single sign-on (SSO)

Ability to require a user to sign on to an application only once and gain access to many different application components, even though these components may have their own authentication schemes. Single sign-on is achieved using identity assertion, LoginModules, and tokens. See also authentication, Cross-Domain Single Sign-on, Identity Assertion, JAAS LoginModule, token, user.

single sign-on (SSO) with Desktop clients

Ability of desktop clients that have authenticated in the Windows Active Directory enviroment to access a Web application running on WebLogic Server and use their Windows Active Directory credentials to authenticate to the server.

SSO with desktop clients is achieved by usig the Negotiate Identity Assertion provider. The provider decodes SPNEGO tokens to obtain Kerberos tokens, validates the Kerberos tokens, and maps Kerberos tokens to WebLogic users. See also Negotiate Identity Assertion provider and Simple and Protected Negotiation Mechanism (SPNEGO).

SSL hardware accelerator

A peripheral Secure Sockets Layer (SSL) platform that attaches to a Web switch with the express purpose of improving SSL performance for a client. For example, the Alteon SSL Accelerator can be used with WebLogic Server. This accelerator performs a TCP handshake with the client (in this case, WebLogic Server) through a Web switch and performs all the SSL encryption and decryption for the session.

SSL tunneling

Tunneling Secure Socket Layer (SSL) over an IP-based protocol. Tunneling means that each SSL record is encapsulated and packaged with the headers needed to send the record over another protocol.

SSPI MBean

Interfaces used by BEA to generate MBean types for the WebLogic security providers, and from which you generate MBean types for custom security providers. SSPI MBeans may be required (for configuration) or optional (for management). See also custom security provider, MBean type, WebLogic security provider.

subject

A grouping of related information for a single entity, such as a person, as specified by the Java Authentication and Authorization Service (JAAS). The related information includes the Subject's identities, or Principals, as well as its security-related attributes (for example, passwords and cryptographic keys). A subject can contain any number of Principals. Both users and groups can be used as Principals by application servers such as WebLogic Server. In WebLogic security providers (security providers supplied with the WebLogic Server product), the Subject contains a Principal for the user (`WLSUser Principal`) and a Principal for each group of which the user is a member (`WLSGroups Principals`). Custom security providers may store identities differently. See also authentication, custom security provider, group, JAAS control flag, principal, user.

symmetric key cryptography

A key-based cryptography that uses an encryption algorithm in which the same key is used both to encrypt and decrypt the data. Symmetric key cryptography is also called secret key cryptography. See also asymmetric key cryptography.

token

Artifact generated as part of the authentication process of users or system processes. When using Identify Assertion, a token is presented to show that the user has been authenticated. Tokens come in many different types, including Kerberos and Security Assertion Markup Language (SAML). See also authentication, Security Assertion Markup Language (SAML), Secure Sockets Layer (SSL), Identity Assertion, SSL tunneling, Security Assertion Markup Language (SAML), user.

Trust Manager

An interface that enables you to override validation errors in a peer's digital certificate and continue the SSL handshake. You can also use the interface to discontinue an SSL handshake by performing additional validation on a server's digital certificate chain.

trusted (root) certificate authority

A well-known and trusted third-party organization or company that issues digital certificates used to create digital signatures and public-private key pairs. The function of the trusted certificate authority is similar to that of a notary public: to guarantee the identify of the individual or organization presenting the certificate. Trusted certificate authorities issue certificates that are used to sign other certificates. Certificate authorities are referred to as root certificate authorities because their authority is recognized and thus they do not need anyone to validate their identity. Trusted (root) certificate authority (CA) certificates are installed into applications that authenticate certificates. For example, Web browsers are usually distributed with several trusted (root) CA certificates pre-installed. If the server certificate is not signed by a well-known certificate authority and you want to ensure that the server's certificate will be authenticated by the client, it is good practice for the server to issue a certificate chain that terminates with a certificate that is signed by a well-known certificate authority. See also certificate chain, private key, public key.

two-way SSL authentication

Authentication that requires both the client and server to present a certificate before the connection thread is enabled between the two. With two-way SSL authentication, WebLogic Server not only authenticates itself to the client (which is the minimum requirement for certificate authentication), it also requires authentication from the requesting client. Clients are required to submit digital certificates issued by a trusted certificate authority. This type of authentication is useful when you must restrict access to trusted clients only. Two-way SSL authentication is a form of mutual authentication. See also authentication, digital certificate, mutual authentication, Secure Sockets Layer (SSL), trusted (root) certificate authority.

UNIX security realm

A WebLogic Server 6.x security realm. The UNIX security realm executes a small native program, `wlauth`, to look up Users and Groups and to authenticate users on the basis of their UNIX login names and passwords. The `wlauth` program uses PAM (Pluggable Authentication Modules), which allows you to configure authentication services in the operating system without altering applications that use the service. In WebLogic Server 7.0 and later, you can only use the UNIX security realm when using Compatibility security. See also authentication, authorization, Compatibility security, group, security realm.

user

An entity that can be authenticated. A user can be a person or a software entity, such as a Java client. Each user is given a unique identity within a security realm. For more efficient security management, BEA recommends adding users to groups. A group is a collection of users who usually have something in common, such as working in the same department in a company. Users can be placed into groups that are associated with security roles, or be directly associated with security roles. See also entity, group, security role, WebLogic resource.

WebLogic component

WebLogic Server implements J2EE component technologies, which include servlets, JSP Pages, and Enterprise JavaBeans. To build a WebLogic Server application, you must create and assemble components, using the service APIs when necessary. Components are executed in the WebLogic Server Web container or EJB container. Web components provide the presentation logic for browser-based J2EE applications. EJB components encapsulate business objects and processes. See also WebLogic container, Windows NT security realm.

WebLogic container

To promote fast development and portability, J2EE identifies common services needed by components and implements them in the container that hosts the component. Containers provide the life cycle support and services defined by the J2EE specifications so that the components you build do not have to handle underlying details. A component has only the code necessary to describe the object or process that it models. It has no code to access its execution environment or services such as transaction management, access control, network communications, or persistence mechanisms. These services are provided by the container, which is implemented in WebLogic Server. Additionally, WebLogic containers give applications access to the J2EE application programming interfaces (APIs). WebLogic containers are available for use once the server is started. This component/container abstraction allows developers to work within their fields of expertise. WebLogic Server provides two types of containers: the Web container and the EJB container. See also WebLogic component, Windows NT security realm.

WebLogic J2EE service

WebLogic Server implements J2EE services, which include access to standard network protocols, database systems, and messaging systems. To build a WebLogic Server application, you must create and assemble components, using the service APIs when necessary. Web applications and EJBs are built on J2EE application services, such as JDBC, Java Messaging Service (JMS), and Java Transaction API (JTA). See also WebLogic component.

WebLogic MBeanMaker

Command-line utility that takes an MBean Definition File (MDF) as input and output files for an MBean type. See also MBean Definition File (MDF), MBean type.

WebLogic resource

Entities that are accessible from WebLogic Server, such as events, servlets, JDBC connection pools, JMS destinations, JNDI contexts, connections, sockets, files, and enterprise applications and resources, such as databases. See also entity.

WebLogic Security Framework

Interfaces in the `weblogic.security.service` package that unify security enforcement and present security as a service to other WebLogic Server components. Security providers call into the WebLogic Security Framework on behalf of applications requiring security services. See also security provider.

WebLogic security provider

Any of the security providers that are supplied by BEA as part of the WebLogic Server product. These providers were developed using the Security Service Provider Interfaces (SSPIs) for WebLogic Server. See also custom security provider, security provider, Security Service Provider Interfaces (SSPIs).

WebLogic Security Service

The WebLogic Server subsystem that implements the security architecture. This subsystem comprises there major components: the WebLogic Security Framework, the Security Service Provider Interfaces (SSPIs), and the WebLogic security providers.

WebLogic Server domain

A collection of servers, services, interfaces, machines, and associated WebLogic resource managers defined by a single configuration file. See also WebLogic resource.

Windows NT security realm

A WebLogic Server 6.x security realm. The Windows NT Security realm uses account information defined for a Windows NT domain to authenticate Users and Groups. In WebLogic Server 7.0 and later, you can only use the Windows NT security realm when using Compatibility security. See also authentication, authorization, Compatibility security, group, security realm, user.