BEA WebLogic Server™

Introduction to WebLogic Security

Release 8.1
Document Date: December 9, 2002
Revised: NA

Copyright

Restricted Rights Legend

Trademarks or Service Marks

Introduction to WebLogic Security

Part Number	Date	Software Version
N/A	December 9, 2002	BEA WebLogic Server Version 8.1

Contents

About This Document

1. Overview of the WebLogic Security Service

2. Security Fundamentals

3. The WebLogic Security Service Architecture

4. Terminology

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 and its features.
- Chapter 2, "Security Fundamentals" describes security concepts as they relate to WebLogic Server security.
- Chapter 3, "The WebLogic Security Service Architecture," describes the WebLogic Server Security architecture.
- Chapter 4, "Terminology," defines Key terms that you will encounter throughout the WebLogic Server security documentation.

Audience

This document is intended for the following audiences:

- Application developers who use the WebLogic Security service to secure interactions between Enterprise JavaBeans (EJBs), Web applications, or Java applications, and WebLogic Server.
- Security vendors or sophisticated application developers who use the Security Service Provider Interfaces (SSPIs) provided by the WebLogic Security service to write security services for the WebLogic Server environment.

- Administrators of WebLogic Server who are responsible for the installation, configuration, and deployment of WebLogic Server, including security.

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 use with WebLogic Server.
- Securing a WebLogic Server Deployment —This document explains how to use the security features of WebLogic Server to protect a WebLogic Server deployment.
- Upgrading Security in WebLogic Server Version 6.x to Version 8.1—This document provides procedures and other information you need to upgrade earlier versions of WebLogic Server to WebLogic Server 8.1. It also provides information about moving applications from an earlier version of WebLogic Server to 8.1.
- Security FAQ—This document introduces WebLogic Server Frequently Asked Questions.
- Security Javadocs—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`*

Convention	Item
UPPERCASE TEXT	Indicates device names, environment variables, and logical operators. *Examples*: 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*: `buildobjclient [-v] [-o name ] [-f file-list]... [-l file-list]...`
`|`	Separates mutually exclusive choices in a syntax line.
`...`	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*: `buildobjclient [-v] [-o name ] [-f file-list]... [-l file-list]...`
. . .	Indicates the omission of items from a code example or from a syntax line. The vertical ellipsis itself should never be typed.

CHAPTER

1 Overview of the WebLogic Security Service

While other security documents in the WebLogic Server documentation set guide users through a specific job task—such as developing a custom security provider or managing the WebLogic Security Service—this *Introduction* is intended for all users of the WebLogic Security Service. This document summarizes the features of the WebLogic Security Service, presents an overview of the architecture and capabilities of the WebLogic Security Service, and defines WebLogic Server security terms. It is the starting point for understanding the WebLogic Security Service.

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

Audience

This document is intended for the following audiences:

- Application developers who use the WebLogic Security Service to secure interactions between Enterprise JavaBeans (EJBs), Web applications, or Java applications, and WebLogic Server.
- Security vendors or sophisticated application developers who use the Security Service Provider Interfaces (SSPIs) provided by the WebLogic Security Service to write their own security services for use in the WebLogic Server environment.
- Administrators of WebLogic Server who are responsible for the installation, configuration, and deployment of WebLogic Server, including security.

The Security Challenge of the Web

Installing and maintaining security is a huge challenge for an IT organization that is providing new and expanded service to customers using the Web. To serve a worldwide network of Web-based users, the 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.

This document focuses on securing the Web application component of an enterprise system—the Java-based application and the BEA WebLogic Server on which it is deployed. This release of WebLogic Server incorporates a completely redesigned security architecture that provides a unique and secure foundation for Web applications. By taking advantage of these new features in WebLogic Server, applications benefit from a comprehensive, flexible security infrastructure designed to address the security challenges of Web applications. BEA WebLogic Server security services 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 third-party security management solution.

Practical Benefits

BEA WebLogic Server security provides the following benefits to help solve the practical dilemmas that information technology (IT) departments face when installing and maintaining Web applications:

- Provides end-to-end security for WebLogic Server-hosted applications, from mainframe to the Web browser
- Integrates legacy security schemes with WebLogic Server security, thus leveraging all existing investments
- Integrates best-of-breed security tools into a flexible, unified system to ease security management across the enterprise
- Enables adding custom security extensions to simplify creation of tighter security policies
- Allows mapping company business rules to security policies in distributed deployments, providing easy customization of application security to business requirements
- Enables easy updates of security policies, with no downtime

- Runs more than one security plug-in at a time, as part of a transition scheme or migration path
- Supports standard J2EE security technologies such as JAAS

With an open architecture, standard interfaces, and unified administration, BEA WebLogic Server security gives the IT department the tools it needs to address real-world issues in security. BEA WebLogic Server's new security architecture provides essential integration or "plug-in" points that allow best-in-breed solutions to be provided by commercial security vendors or customers themselves. In addition, BEA WebLogic Server provides pre-built implementations (WebLogic security provider plug-ins) for most of these plug-in points.

The security services design for BEA WebLogic Server sets a new standard for security architecture in application servers. The new security services offer a carefully crafted, unified approach for security that is:

- Comprehensive and standards-based, out of the box
- Easily adaptable for customized security solutions
- Modularized so that security infrastructures can change over time to meet the requirements of a particular company

Balancing Ease of Use and Customizability

The components and services of the WebLogic Server security system seek to strike a balance between ease of use (for end users and for administrators) and customizability (for application developers and for security service providers).

***Easy to use**:* For the end user, a secure BEA 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. Single sign-on is automatically available to BEA WebLogic Server applications, without additional programming.

Manageable: Administrators who configure, install, and deploy applications to the BEA WebLogic Server environment can use the BEA-supplied security plug-ins that support all required security functions, out of the box. The administrator can store authorization and security policy information in the WebLogic Server-supplied security store (an embedded, special-purpose LDAP directory server). Existing external security stores such as Lightweight Directory Access Protocol (LDAP) servers can also be used with the WebLogic environment. Native, third party, and customized security services can all be integrated into the WebLogic Server management environment, so that configuration and management of all security solutions can be done through the WebLogic Server Administration Console.

Customizable: For application developers, BEA WebLogic Server supports the WebLogic Security API and J2EE security standards such as JAAS and JSSE, for creating a fine-grained and customized security environment for applications that connect to WebLogic Server.

For security-specific application developers and third-party providers of security service technology, WebLogic Server Security Service Provider Interfaces (SSPIs) support both the development of new security plug-ins for the WebLogic Server environment, and interconnection of third-party services to the BEA-supplied security plug-ins.

Securing BEA Web Services Clients

In BEA WebLogic Server, the Web Services run-time is tightly integrated with the WebLogic Server Security framework. As with securing Web clients, specifying SSL security for BEA Web Services requires only a simple modification to the Web Services deployment descriptor, *web-services.xml*. In addition, the Administration Console has a special Web Services Components deployment node so that an Administrator can perform standard administration tasks, including security tasks, on Web Services components. Administrators can configure user authentication and service authorization using the same facilities they use to configure security for other server components.

BEA provides confidentiality for Web Services via the SSL (Secure Sockets Layer) protocol for assuring security between two hosts connected on an insecure network. In addition to server-side support, BEA provides a Web Services client with an integrated SSL implementation. Alternatively, developers can plug-in their own SSL implementation via a public API.

Within the Web Services framework, BEA provides the architecture for pluggable handlers on both the client and the server implementations. These handlers can be used to moderate Web Service transactions and provide additional functionality such as message level encryption, digital signatures, and token-based authentication.

A Key Advantage: Security via Users, Roles, and Security Policies

The key to the BEA WebLogic Server security architecture is the organization of application users into *users* and *groups* that take on *roles* according to defined *security policie*s. Users can be organized into groups. Groups can be used to represent organizational boundaries as well as to simplify administration. Each application user and group is mapped to a role dynamically during application execution, when authorization is needed.

Roles and policies determine access to system resources, and permitted behaviors. User roles are registered by an administrator using the built-in menu-driven security policy tool embedded in the BEA-supplied Authorization plug-in. The security policy tool's interface reflects business concepts, not programming concepts, and allows an administrator to create simple prose-based rules for dynamically assigning roles and calculating access privileges. Application developers are freed from having to write application code to implement complex business policies, because the policy tool separates the tasks of business policy creation and application creation.

The roles that a user can be assigned to are determined by policies defined by the administrator, on behalf of the company. Since policies reflect business security rules, a company's management sets security policies rather than the software development staff. Security policies can easily be changed with changes in business conditions.

The role-and-policy-based security scheme replaces the previous scheme of users, groups, and access control lists (ACLs), and offers clear advantages for ease of administration and ease of adaptability as security requirements change. Using roles and policies for authorization permits dynamic computation of access status for each resource, for each user or group.

The WebLogic Server dynamic, role-based authorization scheme can be applied to all WebLogic Server resources. The administrator and applications developer are no longer constrained by the limitations of the declarative security model in J2EE, which embeds security constraints in the code and makes it difficult to modify a security scheme when business requirements change.

Authentication and Authorization: Some Details

Here are some details on how BEA WebLogic Server implements the primary task areas of a security system, which are authentication (determining a user's identity as a valid user) and authorization (determining a user's role or roles and computing the appropriate access privileges based on the policies in place).

Authentication

Authentication is the process of ascertaining that a user requesting system services is a valid user. BEA WebLogic Server builds upon the authentication classes of the Java Authentication and Authorization Service (JAAS), which is a standard extension to the security in the Java Software Development Kit, version 1.3. The JAAS standard provides the support needed to submit a username and credential (password or digital certificate) when authenticating a user to WebLogic Server. BEA WebLogic Server goes beyond this by providing a link to legacy systems so a particular system's authentication credentials can be transparently used by WebLogic Server.

In addition, the new architecture supports perimeter authentication via Web server, firewall, or Virtual Private Network (VPN). Multiple "security token types" are possible (e.g. Liberty token types, or Microsoft Passport™, or SAML Assertions). Perimeter authentication is supported by multiple protocols, such as HTTP, and IIOP-CSIv2.

BEA WebLogic Server architecture supports:

- Certificate-based authentication, directly with WebLogic Server
- HTTP certificate-based, proxied through an external Web server
- Delegated username/password authentication (via Unix, for example)

The WebLogic Server authentication scheme is flexible enough to benefit developers who have special security situations. By using and extending classes and configurations provided, developers can link to multiple authentication and authorization stores. Examples of connectivity include NT domains, LDAP directories, and Unix security services. For example, an organization might prefer to use the execute permissions associated with a Unix file system for a set of servlets stored on the Unix machine. To enable this, WebLogic Server can defer authentication of users to Unix, and reflect access control to Unix files in Java.

Levels of Authentication

Since security requirements vary greatly, the server is easily configurable to allow different levels of authentication depending upon the needs of the deployment. When a client, administrator, or peer wants to access a WebLogic Server resource, it may be necessary to authenticate identities. BEA WebLogic Server supports two default mechanisms for authenticating users: passwords and digital certificates.

For minimal security, password-based authentication allows users to enter a username and password to authenticate themselves when requesting access to a given resource on the server. This is the basic authentication scheme defined in the HTTP protocol. This basic authentication is very easy to implement and manage, but provides limited security because passwords provide a weak form of authentication.

For stronger authentication, cryptographic authentication is supported in the form of digital certificates. These certificates are issued from a trusted third-party called a Certificate Authority (CA). A CA issues certificates that can be used to prove the

identity of servers or users. BEA WebLogic Server does not restrict the number of CAs or the number of levels in a certificate chain. This provides more flexibility in business-to-business applications that require longer CA chains.

Authentication via digital certificates is the key to authentication using the Secure Sockets Layer (SSL) security protocol, which is supported by WebLogic Server. BEA WebLogic Server is compatible with the X.509v3 international digital certificate standard. This compatibility allows use of certificates from any CA that supports this standard. For mutual authentication, both the client and server must present a certificate before the connection is enabled between the two. When mutual authentication is used, the identity from the certificate can be mapped to a WebLogic Server user identity through a customization point called certificate authenticator that is registered with an Identity Asserter.

BEA WebLogic Server provides full-strength, 128-bit encryption for customers in the United States and Canada, and international customers that are allowed full-strength encryption by U.S. trade laws. A 56-bit encryption scheme is compatible with "low-strength" encryption, allowing universal compatibility. For this compatibility, WebLogic Server generates a temporary low-strength key for that connection.

Authorization

Authorization determines what a user is allowed to do, using security roles and policies. Each user is assigned one or more roles, dynamically during application execution. Roles can be implemented as an identity, or as a named collection of permissions. Role assignments are dynamically computed according to security policies, set by the administrator using the BEA WebLogic Server Policy tool, as the user seeks to access various resources. With this release of WebLogic Server, Access Control Lists (ACLs) have been deprecated, and have been replaced by security policies.

The security roles and policies determine the level of access a particular user has to resources hosted on BEA WebLogic Server, for example:

- Enterprise JavaBeans
- Web Applications (Servlets, JSP)
- Web Services
- Miscellaneous J2EE resources (JMS destinations, JNDI)

Authorization and role assignments are delegated to the security service from the WebLogic Server container. This allows integration of external plug-ins and provides a unified authorization approach to all resources. The tight coupling between implementations of roles and access decisions allows for authorization decisions and role assignments to be based on context, which supports more “real life” conditions of doing business.

It is often necessary for security policies to be changed dynamically as new users are added and situations change. To accommodate this, BEA WebLogic Server can dynamically update users and roles by leveraging an external, centralized security database or service, such as LDAP (or by using the embedded LDAP directory server that has been customized for use by WebLogic Server).

In previous versions of WebLogic Server, user roles were defined exclusively in J2EE Deployment Descriptors. Changing these static role assignments required editing and updating Deployment Descriptors. The new architecture supports a comprehensive scheme that consumes all previous definitions and assignments, and lets administrators use the Administration Console to manage role definitions and assignments without editing deployment descriptors. This separation of the security of the application from the business logic simplifies security management.

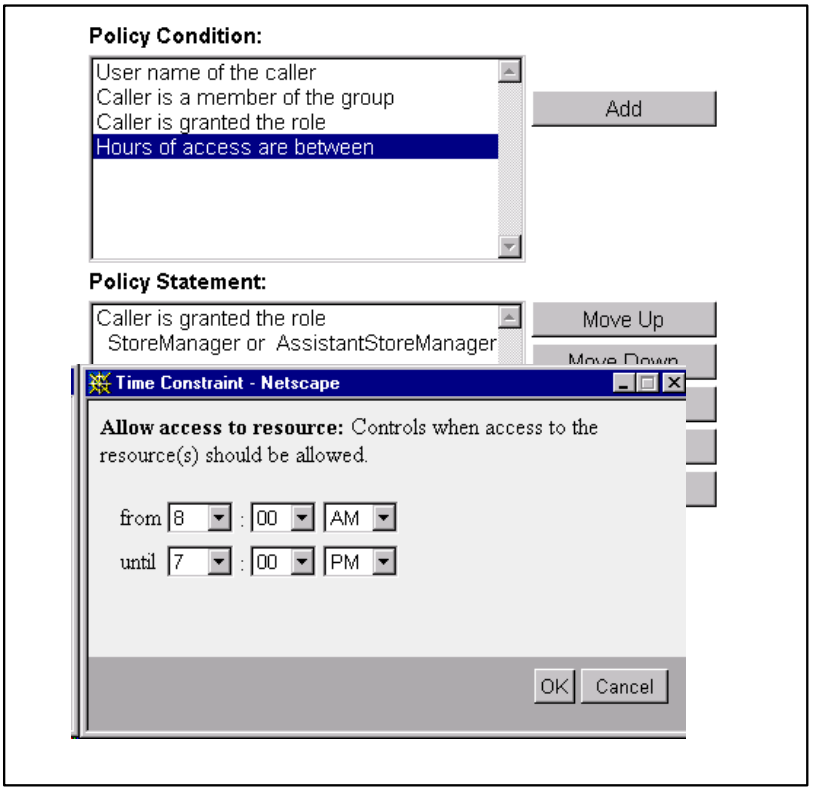
Role definitions are specific to a unified security realm that represents the broadest policy scope to which security policies apply (e.g., the entire company, the company’s Western Division, the company’s Pacific Rim branch). Roles can be defined as *scoped*—associated with given resources; or *global*—associated with all resources. Roles can be assigned statically for specified users or groups at administration time, or dynamically, based on the context of the request at run-time.

Setting Policies: No Programming Required

The built-in Policy engine provides a GUI interface that lets Administrators set policies in the Administration Console—without writing application code. By right clicking on the system resource displayed in the Administration Console, users can select among the constraints displayed on the drop-down menus.

Figure 3 illustrates this simple menu-based approach to adding or changing security in applications.

Figure 1-1 Authorization Policy Sample



Security Auditing

BEA WebLogic Server can be configured to allow administrators to construct security audits for their systems. Administrators can configure the audit to detect and log authentication attempts, access attempts, and other significant security events. Administrators can use the BEA-supplied Auditing provider plug-in, which creates a log entry for access failures. More sophisticated auditing schemes can be implemented using custom Auditing providers or third-party auditing tools. As with the other

security tools, administrators can have multiple Auditing provider plug-ins in place, and set various conditions for their use. This offers administrators fine-grained control over when and whether to take action on an event.

Connection Filtering

BEA WebLogic Server also provides administrator control over permissions to establish a connection to the server. The Connection Filter feature allows enforcement of rules that govern whether a connection should be established based on client IP access or protocol. This capability can be used to restrict the location from which connections to the application server are made, enhancing security of the application. Connection Filter features are configurable from the Administration Console.

Counter Measures for Denial-of-Service and other Attacks

In enterprise applications, denial-of-service and other attacks can render significant damage. The BEA WebLogic Server is designed to prevent certain forms of denial-of-service attacks. From the Administration Console, the administrator can set timeouts and other parameters that will counteract attempts to attack the integrity and availability of the application running on WebLogic Server. In addition, BEA WebLogic Server provides measures to counter both online and offline guessing of users and passwords. Offline protection for passwords is protected through use of encryption and other security techniques specifically targeted to counter brute force and dictionary attacks. Counter measures for online password guessing include the ability for administrators to control the number of times a login attempt can fail within a given period of time before the account is disabled or locked-out.

Unified Administration for Security Services

BEA WebLogic Server supports unified administration of security services through the Administration Console, for third-party and customized security services, as well as for BEA implementations. The Administration Console is a security command center, providing tools for functions such as the Role Mapper, the Auditor, the Authenticator, the Authorizer, the Credential Mapper, and the Keystore. For compatibility with earlier WebLogic Server 6.x environments, the Administration Console also provides a Realm function and a Realm Adapter.

With BEA WebLogic Server, the Administration Console becomes the central tool (and the best tool) for creating and modifying the security sections of configuration files and deployment descriptors. Configuration files no longer have clear text passwords. BEA WebLogic Server automatically creates protected passwords when it writes this information to the configuration file. Passwords are created on a per-realm basis.

Security Management and Storage

Security management in the Administration Console is implemented using the classes of the Java Management Extensions (JMX) package. JMX supports both console and programmatic access to management functions, via Management Beans (MBeans). Plug-ins can extend base MBeans to meet their needs. This means that any security function in the Administration Console can be extended. Third-party security vendors or security-specific application programmers can add or replace pages in the Administration Console, as desired.

Security storage is provided by the Keystore, and by the BEA WebLogic Server system data store. The Java standard keystore stores public and private keys with encrypted storage. The WebLogic Server system data store is an embedded LDAP directory server optimized for use within WebLogic Server. The system data store provides unified storage (per security realm) of security information and proof material for users, groups, roles, and security policies. Customers can also use an existing LDAP directory service for retrieval of storage of user and group information.

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	Most 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 *Programming WebLogic Security*.
JNDI authentication	Java Authentication and Authorization Service (JAAS) authentication classes.
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 allows you to customize the data you want to record. You can further customize your auditing needs by implementing the Auditing SSPI to create a custom Auditing provider.
Defining security constraints in the weblogic.xml, weblogic-ejb-jar.xml, and weblogic-ra.xml files.	The functionality is enhanced so that security constraints can also be specified through the WebLogic Server Administration Console.
System password	There is no specific system account in WebLogic Server 8.1.

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 previous releases of WebLogic Server are deprecated in this release. ACLs are replaced by security policies.
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 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 previous releases of WebLogic Server 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 realm. 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 previous releases of WebLogic Server.	Support for multiple security providers.
The SSL Protocol	The SSL support in WebLogic Server has been updated to provide the JSSE standard and the TLS v1 protocol.
This feature was not available in previous releases of WebLogic Server.	Support for J2EE JKS Keystores

Migrating from BEA WebLogic Server 6.x Security to 8.1 Security

Customers who installed security under BEA WebLogic Server 6.x have three choices when they transition to BEA WebLogic Server 8.1:

- Running in "native" mode (converting completely to WebLogic Server 8.1 security)
- Running in "compatibility security" mode using the Realm Adapter (supports only existing 6.x security features)
- Running in "mixed" mode (using the Authorization service of WebLogic Server 8.1, while retaining the 6.x authentication service)

An Export utility (not part of BEA WebLogic Server, but provided on BEA dev2dev Online at http://dev2dev.bea.com/) enables customers to extract users (but not passwords), groups, and ACLs from WebLogic Server 6.x and import the information into the embedded LDAP Directory Server in WebLogic Server.

Summary

BEA WebLogic Server's open, flexible security architecture delivers advantages to all levels of users and introduces an advanced security design for application servers. BEA WebLogic Server provides out-of-the-box implementations for each of the Security Service Provider Interfaces (SSPIs). Companies now have a unique application server security solution that, together with clear and well-document security policies and procedures, can assure the confidentiality, integrity and availability of the server and its data. The key benefits of the new security architecture of BEA WebLogic Server include:

- Separate security details from the application infrastructure, which makes security easier to install, maintain, and modify as requirements change.

- Out-of-the-box security for WebLogic Server applications with BEA-supplied security plug-ins.
- More than one security plug-in for a given function. Any installation can accommodate legacy security measures while migrating to a new scheme, without undue disruption to an application and its users.
- Customized choice of BEA, custom-developed, or third party security plug-ins because security plug-ins is written to a defined (security SPI) interface.
- Ease of administration, as all security plug-ins in an installation can be managed through the BEA Administration Console.
- Support of the J2EE standards and the SDK 1.3 so that development teams can take advantage of the latest Java standard security technologies, as well as BEA enhancements.

CHAPTER

2 Security Fundamentals

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

- J2EE and WebLogic Security
- Security Realms
- Authentication
- Embedded LDAP Server
- Authorization
- Auditing
- Secure Sockets Layer (SSL)
- Firewalls
- Security Providers
- PKI
- Single Sign-on

J2EE and WebLogic Security

For implementation and use of user authentication and authorization, BEA WebLogic Server utilizes the security services of the SDK version 1.3 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 1.3 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 1.3.

The following topics are discussed in this section:

SDK 1.3 Security Packages

WebLogic Server is compliant with and supports the following SDK 1.3 security packages:

- The Java Secure Socket Extension (JSSE)
- Java Authentication and Authorization Services (JAAS)
- The Java Security Manager
- "Java Cryptography Architecture and Java Cryptography Extensions (JCE)" on page 2-3

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. BEA WebLogic Server provides Secure Sockets Layer (SSL) support for encrypting data transmitted across WebLogic Server clients, and other servers.

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.

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 resources running in the Java Virtual Machine. Use of the Java Security Manager to protect resources in WebLogic Server is optional.

You may use the Java Security Manager is the following ways:

- To set default security policies for EJBs, and J2EE Connector Resource Adapters in the Java security policy. You cannot modify security policies for Web applications.
- To set security policies for a specific EJB or a Resource adapter by adding policies to their deployment descriptors.

  Note: You cannot use the Java Security Manager to modify security policies for Web applications.

For more information on using the Java Security Manager to protect WebLogic Server resources, see "*Using the Java Security Manager*" in *Managing 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.

The Java Cryptography Architecture and JCE have been integrated into the Java 2 SDK, Standard Edition, v 1.4.

WebLogic Server fully supports these security APIs.

WebLogic APIs versus J2EE APIs

To be supplied.

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, roles, and security policies. 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 role assigned to the user in the relevant realm and the security policy of the particular WebLogic resource.

Figure 2-1 Security Realms (To be supplied)

For more information on security realms in WebLogic Server, see *Managing WebLogic Security*.

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 that 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 provider concepts and functionality.

Users

Users are entities that can be authenticated in a security realm (such as myrealm). A user can be a person, such as application end user, or a software entity, such as client application and other instances of WebLogic Server. As a result of authentication, a user 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 roles, or be directly associated with 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 providers 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 security policy of the resource and the user's principal that the user has been assigned to make sure that the user has the required permissions to continue.

When defining a user in WebLogic Server, you also define a password for that user. WebLogic Server hashes all passwords. 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 for matching.

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

For more information, see "Defining Users" in *Managing WebLogic Security*:

Groups

Groups are logically ordered sets of users. 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 defining a security policy for a resource and assigning a role, or access privilege, to a group. Usually, group members have something in common. For example, a company may separate its sales staff into two groups, Sales Representatives and Sales Managers, because staff members would logically have different levels of access to WebLogic Server resources depending on their job functions.

For more information, see the "Defining Groups" topic in *Managing WebLogic Security*.

Principals and Subjects

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 like WebLogic Server. The Java Authentication and Authorization Service (JAAS) requires that **subjects** be used as containers for authentication information, including principals.

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

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

Figure 2-2 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.

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 reside in 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 end 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.

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 a 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-->General tab of the Administration Console 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.

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

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 you or an administrator 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.

Mutual authentication

With mutual authentication, both the client and the server are required to authenticate themselves to each other. This can be done using by means of certificates or other forms of proof material. WebLogic Server supports two-way SSL authentication, which is a form of mutual authentication. For more information, see "One-way/Two-way SSL Authentication" on page 2-22.

Types of Authentication

WebLogic Server users must be authenticated whenever they request access to a protected WebLogic Server resource. For this reason, each user is required to provide a credential (a username/password pair or a digital certificate) to WebLogic Server. The following types of authentication are supported by default by the WebLogic security providers that are included in the WebLogic Server distribution:

- Username/Password authentication
- Certificate authentication
- Perimeter authentication

WebLogic Server can use the WebLogic security providers that are provided as part of the WebLogic Server product or Custom security providers to perform the different types of authentication. For information on WebLogic security providers and how to configure authentication, see "The WebLogic Security Providers" on page 3-7 and the following sections in *Managing WebLogic Security*:

- "Introduction of Security Providers"
- "Configuring the SSL Protocol"

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 resource.

The SSL (or HTTPS) protocol can be used to provide an additional level of security to password authentication. Because the SSL protocol 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.

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. In this case, 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. Two-way SSL authentication is useful when you must restrict access to trusted clients only. For example, you might restrict access by accepting only clients with digital certificates provided by you.

Note: Two-way SSL authentication is supported by the WebLogic security providers that are 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.

Figure 2-3 illustrates perimeter authentication.

Figure 2-3 Perimeter Authentication (To be supplied)

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," 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. 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.

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.

WebLogic Server provides a rich single sign-on environment on which enterprise applications can be built. The single sign-on capabilities reach far beyond those of any other J2EE application server to address the issues found in today's enterprise environments. The WebLogic Security Framework allows the single sign-on capabilities to be integrated with the best-of-breed security vendors and allows for customization to meet an enterprise's ever-changing needs.

Embedded LDAP Server

The embedded LDAP server is used to store user, group, security roles, and security policies for the WebLogic security providers. It is a complete LDAP server. You can use LDAP to access and modify entries in the LDAP server. The embedded LDAP connection allows the WebLogic security providers to read and write LDAP information from and to the server. You can use an LDAP Browser to import and export various information into and from the server. WebLogic Server does not support adding attributes to the embedded LDAP server.

Note: Question to Reviewers: Will we add LDAP import and export to the console in the Olympic release?

Note: Reminder: Refer to the TOIs for the various default providers. They describe how the security information such as users, groups, roles, and policies are stored in the LDAP server.

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 provider concepts and functionality:

WebLogic Resources

A **WebLogic resource** is a structured object used to represent an underlying WebLogic Server entity that can be protected from unauthorized access. The level of granularity for WebLogic resources is up to you. For example, you can consider an entire Web application, a particular Enterprise JavaBean (EJB) within that Web application, or a single method within that EJB to be a WebLogic resource.

Note: WebLogic resources replace WebLogic Server 6.x access control lists (ACLs).

WebLogic resource implementations are available for:

- Administrative resources
- Application resources
- COM resources
- EIS resources
- EJB resources
- JDBC resources
- JMS resources
- JNDI resources
- Server resources
- URL resources
- Web Service resources

Note: Each of these WebLogic resource implementations are explained in detail in the *WebLogic Server 7.0 API Reference Javadoc*.

Roles

Roles are an abstract, logical collections of users similar to a group. The difference between groups and roles is a group is a static identity that a system administrator assigns, while membership in a role is dynamically calculated based on data such as username, group membership, or the time of day. Roles are granted to individual users or to groups. For more efficient management, BEA recommends granting roles to groups rather than to individual users. Once you create a 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.

WebLogic Server supports two types of roles:

- Global roles—Apply to all WebLogic resources in a security realm. WebLogic Server defines a set of default global roles for protecting WebLogic resources.

You do not have to use the default global roles. You can create global roles that more closely reflect your own business structure and practices.

- Scoped roles—Apply to a specific instance of a WebLogic resource such as a method of an EJB or a branch of a JNDI tree. Most roles are scoped.

In WebLogic Server 6.x, roles were used to protect EJBs and Web applications only. In this release of WebLogic Server, roles are expanded to protect all of the defined WebLogic Server resources. For a list of the defined WebLogic resources, see "WebLogic Resources" on page 2-15.

Multiple roles (either scoped or global) can be applied to a WebLogic resource. Both scoped and global roles are stored in the default Role Mapping provider. To use a role to create a security policy for a WebLogic resource, the role must be in the Role Mapping provider configured in the default security realm. By default, the WebLogic Role Mapping provider is the configured Role Mapping provider.

Security Policies

In the previous release of WebLogic Server, ACLs were used to protect WebLogic resources. In 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 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 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 resource. For a list of the defined WebLogic resources, see "WebLogic Resources" on page 2-15.

Security policies are stored in an Authorization provider. 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 Authentication provider configured in the default security realm. To use a role to create a security policy, the role must be defined in the Role Mapping provider configured in the default security realm. By default, the WebLogic Authentication and Role Mapping providers are configured.

By default, security policies are defined in WebLogic Server for the WebLogic resources. These security policies are based on roles and groups. You also have the option of basing a security policy on a user. BEA recommends basing security policies on roles rather than users or groups. Basing security policies on roles allows you to manage access based on a role 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 *Managing WebLogic Security*.

ContextHandlers

A ContextHandler is a high-performing WebLogic class that allows a variable number of arguments to be passed as strings to a method.

The ContextHandler interface provides a way for an internal WebLogic container to pass additional information to a WebLogic Security Framework call, so that a security provider can obtain additional context 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 a security provider to know what names to look for (that is, use of a ContextHandler requires close cooperation between the WebLogic container and a security provider). The name/value list is also called a context element, and is represented by a ContextElement object.

For example, a ContextHandler argument is passed into the isAccessAllowed method of an Access Decision. As another example, if one was attempting to access a file in a directory, the name of that file could be passed through a ContextHandler object.

The Role Mapping providers use the `ContextHandler` to request various pieces of information about the request. They construct a set of `Callback` objects that represent the type of information being requested. This set of `Callback` objects is then passed as an array to the `ContextHandler` using the `handle` method.

The Role Mapping providers use the values contained in the `Callback` objects, the subject, and the resource to compute a list of roles to which the subject making the request is entitled, and pass the list of applicable roles back to the WebLogic Security Framework.

The Authorization providers' Access Decisions also use the `ContextHandler` to request various pieces of information about the request. They too construct a set of `Callback` objects that represent the type of information being requested. This set of `Callback` objects is then passed as an array to the `ContextHandler` using the `handle` method.

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, 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.

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 Server 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 back-end, database, and a simple file.

Secure Sockets Layer (SSL)

SSL offers security to applications connected through a network. 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 when connecting to 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.

SSL Features

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 the 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.

The SSL protocol 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.
- 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. If the client is not required to present a digital certificate, the connection type is called one-way SSL 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.

If you are using a Web browser to communicate with WebLogic Server, you can use the Hypertext 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. The use of SSL is signified in the protocol scheme of the URL used to specify the location of WebLogic Server.

- 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 protocol adapter. Consequently, WebLogic Server depends on the implementation of the SSL protocol in the Web browser.

- Java clients connecting to WebLogic Server with the SSL protocol 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 the SSL protocol, 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 then verifies the digital certificate and an SSL authentication connection is established.

With two-way SSL authentication, both the client and server must present digital certificates before the connection thread is enabled between the two. 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.

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 and 1024-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. Because the United States Government relaxed restrictions on exporting encryption software in early 2000, the domestic version of WebLogic Server can be used in most countries.

During installation, you are prompted to choose which strength of the SSL protocol you want to use.We recommend the domestic WebLogic Server distribution because it allows stronger encryption. If you generate a Certificate Signature Request (CSR) using the exportable WebLogic Server distribution, you cannot support high-strength connections and you cannot authenticate clients that present domestic-strength certificates.

The implementation of the SSL protocol provided by WebLogic Server is a pure-Java implementation.

For more information, see the "Configuring the SSL Protocol" section 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 allow 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. We recommend obtaining digital certificates from a certificate authority such as Verisign or Entrust.

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

Certificate Authorities

Digital certificates are issued by a certificate authority. 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 subject.

The subject 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 digital 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 digital 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 (CAs).

The recipient of an encrypted message can develop trust in the private key of a certificate authority recursively, 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. 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 digital 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 the "Configuring the SSL Protocol" section in *Managing WebLogic Security*.

Mutual Authentication

Two-way SSL authentication is a form of mutual.

host name verification (client-side SSL)

trust managers

Asymmetric Key Algorithms

Asymmetric key (also so referred to a 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 Infrastructures (PKI) in WebLogic Server also support 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 "Principal Validation Providers" on page 2-36.

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.

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

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 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 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
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 `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.

Java Cryptography Extensions (for SP2)

hardware/software accelerators

Firewalls

Credential Mapping

Credential Map

Connection Filters

Security policies allow you to secure WebLogic Server resources using some characteristic of a user. However, you can add an additional layer of security by using connection filters to filter network connections. For example, you can deny any non-SSL connections originating outside of your corporate network.

WebLogic Server network connection filters are a type of firewall in that they can be configured to filter on protocols, IP addresses, and DNS node names. For more information, see "Using Network Connection Filters to Protect Application Server Resources" in *Programming WebLogic Security*.

Security Providers

Security providers are modules that "plug into" a WebLogic Server security realm to provide security services to applications. 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 Security Service Provider Interfaces (SSPIs)

The security plug-in scheme in BEA WebLogic Server is based on a set of Security Service Provider Interfaces (SSPIs) for the plug-in points. The SSPIs can be used by BEA administrators, customers, or third-party vendors to develop security plug-ins for the WebLogic Server environment. SSPIs are available for Authentication, Authorization, Auditing, Credential mapping, Role mapping, and Keystore.

The SSPI scheme means that customers have *four* choices for securing WebLogic Server resources:

- Using BEA-supplied, default, security provider plug-ins, which are referred to in this document as WebLogic security providers. These providers are included in the WebLogic Server product.
- Using third-party, security provider plug-ins that are developed based on the WebLogic Server SSPIs, which are referred to in this document as custom security providers. These providers are available for purchase from several security vendors.
- Using the WebLogic Server SSPIs to create customized security provider plug-ins for WebLogic Server systems. Sample custom security providers are also available from the BEA online dev2dev Web site at `http://dev2dev.bea.com/index.jsp`. For more information on developing custom security providers, see *Developing Security Services for WebLogic Server*.

Security Service Provider Interface (SSPI) MBeans

A required task for developing a custom security provider is to generate an MBean type for the custom security provider to be developed. The term **MBean** is short for managed bean, a Java object that represents a Java Management eXtensions (JMX) manageable resource.

Note: JMX is a specification created by Sun Microsystems that defines a standard management architecture, APIs, and management services. For more information, see the *Java Management Extensions White Paper*.

An **MBean type** is a factory for instances of MBeans. You use MBean interfaces called **SSPI MBeans** to create MBean types. There are two types of SSPI MBeans you can use to create an MBean type for a custom security provider:

- **Required SSPI MBeans**, which you must extend because they define the basic methods that allow a security provider to be configured and managed within the WebLogic Server environment.
- **Optional SSPI MBeans**, which you can implement to define additional methods for managing security providers. Different types of security providers are able to use different optional SSPI MBeans.

Once you have created the MBean types, you can use the Administration Console to create instances of MBeans. After you have created the MBean instances, you can use them, via the Administration Console, to configure and manage the custom security provider.

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?

A **security provider database** contains the users, groups, policies, roles, and credentials used by some types of security providers to provide security services. 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 roles, and a Credential Mapping provider requires information about credentials. 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 security providers), or a production-quality, customer-supplied database that you may already be using.

Note: The sample security providers are available under "Code Direct" on the *dev2dev Web site*.

The security provider database should be initialized the first time security providers are used. This initialization can be done:

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

At minimum, you must initialize a security provider database with the default users, groups, policies, roles, or credentials that your Authentication, Authorization, Role Mapping, and Credential Mapping providers expect. For more information, see the following sections in *Managing WebLogic Security*:

- Default Group Associations
- Default Security Policies
- Default Global Roles and Permissions
- Using WebLogic Server to Authenticate to Remote Systems

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 under "Code Direct" on the *dev2dev Web site.*

For example, if you or an administrator 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.

The 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.

Types of Security Providers

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

- "Authentication Providers" on page 2-34
- "Identity Assertion Providers" on page 2-35
- "Principal Validation Providers" on page 2-36
- "Authorization Providers" on page 2-37
- "Adjudication Providers" on page 2-38
- "Role Mapping Providers" on page 2-38
- "Auditing Providers" on page 2-39
- "Credential Mapping Providers" on page 2-40
- "Keystore Providers" on page 2-40
- "WebLogic Realm Adapter Providers" on page 2-41

"Security Provider Summary" on page 2-41 specifies whether you can configure multiple security providers of the same type in a security realm.

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 2-35.

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 to verify the authenticity of **principals** (users/groups) associated with the Java Authentication and Authorization Service (JAAS). For more information about Principal Validation providers, see "Principal Validation Providers" on page 2-36.

You must have 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 Chapter 3, "Authentication Providers."

LDAP Authentication Provider

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.

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.

You can develop Identity Assertion providers that support different token types, including Kerberos, SAML (Security Assertion Markup Language) and Microsoft Passport. 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.

You can configure multiple Identity Assertion providers in a security realm, but none are required. Identity Assertion providers can support more than one token type, but only one token type per Identity Assertion provider can be active at a given time. For example, an Identity Assertion provider can support both Kerberos and SAML, but an administrator configuring the system must select which token type (Kerberos or SAML) is the active token type for the Identity Assertion provider. If this Identity Assertion provider is set to Kerberos, but SAML token types must be supported, then another Identity Assertion provider that can handle SAML must have SAML set as its active token type.

Common Secure Interoperability Version 2 (CSIv2)

WebLogic Server provides support for an Enterprise JavaBean (EJB) interoperability protocol 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 1.3 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.

Principal Validation Providers

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 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.

Authorization Providers

Authorization providers control access to WebLogic resources based on user identity or other information. The WebLogic security architecture supports several types of authorization:

- *Parametric authorization*: The parameters to an operation, which are obtained from the application running inside the resource container, are used in making the authorization decision. Thus, parametric authorization allows an authorization decision about a protected WebLogic resource to be determined based on the context of the request.
- *Permissions-based authorization*: Explicit permissions are given to a user via security policies. Java 2 Enterprise Edition (J2EE) security uses this type of authorization by obtaining data from the `weblogic.policy` file. Although deprecated in WebLogic Server 7.0, access control lists (ACLs) are an example of permissions-based authorization.

  Note: For more information about security policies, see "Understanding WebLogic Security Policies" in *Managing WebLogic Security*.
- *Capabilities-based authorization*: Explicit permissions are given to a user (as in permissions-based authorization), but the objects and operations on those objects are also specified. For example, a user can be given permission to access a particular method of an Enterprise JavaBean (EJB). Thus, permissions may be specified at a high or low level. An example of capabilities-based authorization is the use of roles.

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. Authorization providers can also use Principal Validation providers to verify the authenticity of principals (users and groups) associated with the Java Authentication and Authorization Service (JAAS). For more information about Principal Validation providers, see "Principal Validation Providers" on page 2-36.

You must have 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 an Authorization provider that handles JNDI permissions and another that handles Web application permissions).

Authorization providers and Access Decisions are discussed in more detail in Chapter 6, "Authorization Providers."

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 do to 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`.

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

Adjudication providers are discussed in more detail in Chapter 7, "Adjudication Providers."

Role Mapping Providers

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

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

A Role Mapping provider supplies Authorization providers with this 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 roles in J2EE deployment descriptors, or create them using the WebLogic Server Administration Console. These roles are applied at deployment time (unless you specifically choose to ignore the roles).

You must have 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 role information), or follow a more modular design (for example, configuring one Role Mapping provider that handles mappings for JNDI resources and another that handles mappings for Web applications).

Note: If multiple Role Mapping providers are configured, the set of roles returned by all Role Mapping providers will be *intersected* by the WebLogic Security Framework. That is, 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 Chapter 8, "Role Mapping Providers."

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 Auditor. (The Auditor is the portion of the WebLogic Server Framework that calls into each Auditing provider, enabling audit event recording.)

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

Auditing providers are discussed in more detail in Chapter 9, "Auditing Providers."

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. You can develop a Credential Mapping provider to map credentials in this way.

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 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 Credential Manager—the portion of the WebLogic Security Framework that calls into each Credential Mapping provider to find out if they contain the type of credentials requested by the container—accumulates and returns all the credentials as a list.

Credential Mapping providers are discussed in more detail in Chapter 10, “Credential Mapping Providers.”

Keystore Providers

With WebLogic Server, you can use a Keystore to create and manage password-protected stores of private keys (and their associated public key certificates) and trusted certificate authorities. The Keystore is available to applications that may need it for authentication or signing purposes.

The Keystore provider that is included as part of the WebLogic Server product is used to obtain secured private keys from Keystores. For more information on Keystore providers, see “The WebLogic Security Providers” on page 3-7.

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. There are four WebLogic Realm Adapter providers:

- Authentication (includes an Identity Assertion provider)
- Authorization
- Auditing
- Adjudication!

Security Provider Summary

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

Table 2-2 Security Provider Summary

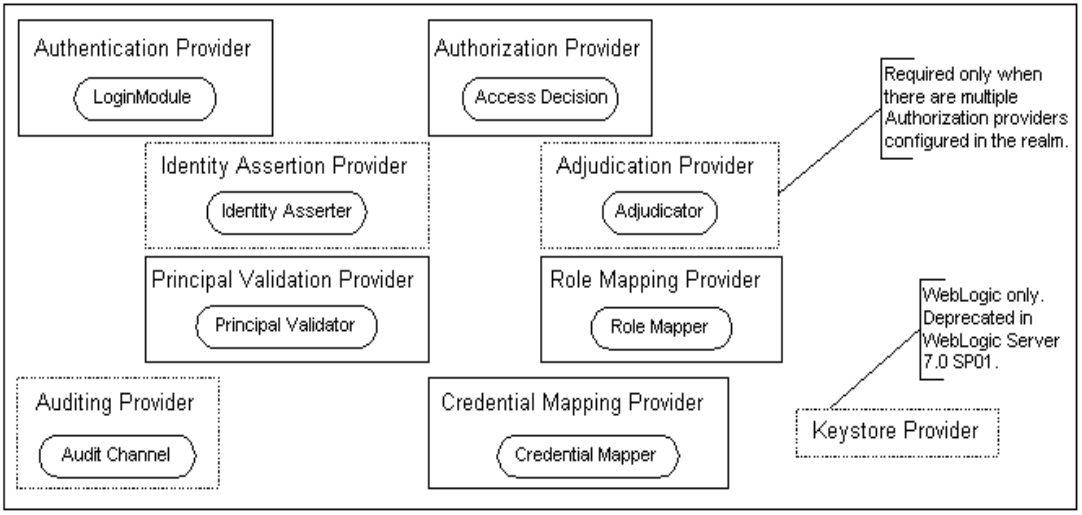
Type	Multiple?
Authentication provider	Yes
Identity Assertion provider	Yes
Principal Validation provider	Yes
Authorization provider	Yes
Adjudication provider	No
Role Mapping provider	Yes
Auditing provider	Yes
Credential Mapping provider	Yes

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 7.0 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 2-4.

Note: If you are upgrading from a 6.x release to the 7.0 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 7.0 model. For information about upgrading, see "Upgrading Security" under "Upgrading WebLogic Server 6.x to Version 7.0" in the *Upgrade Guide for BEA WebLogic Server 7.0*.

Figure 2-4 A WebLogic Server 7.0 Security Realm



Note: The types of security providers that are required for a 7.0 security realm are shown in solid boxes; the security providers that are optional for a 7.0 security realm are shown in dashed boxes.

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 2-4 also shows, some types of security providers are required for a 7.0 security realm to operate properly. Table 2-3 summarizes which security providers must be configured for a fully-operational 7.0 security realm.

Table 2-3 Security Providers in a Security Realm

Type	Required?
Authentication provider	Yes
Identity Assertion provider	No
Principal Validation provider	Yes
Authorization provider	Yes
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 has been deprecated in WebLogic Server 7.0 SP01, and you cannot develop custom Keystore providers.

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

- Configuration Steps for Security
- Setting the Default Security Realm
- Deleting a Security Realm

PKI

digital certificates (generate via certgen utility or certificate request generator servlet)

Single Sign-on

Today, a growing number of applications are being made available over the Web. These applications are typically comprised of different components, each of which may have its own authentication scheme or user registry. As are result, development and operations staffs are faced with the increasingly difficult problem of how to require their user community to authenticate or "sign-on" once, yet have access to each of the components of the application. The ability to require a user to sign-on to an application only once is called single sign-on.

BEA's WebLogic Server attempts to address a number of the issues concerning single sign-on support through a series of approaches, each focused at a different tier in which a portion of an application is housed, and addressing the problems that appear in that tier. The new security architecture provides essential integration points, called security providers, that allow best-in-breed solutions to be "plugged in" by commercial security vendors or customers themselves. In addition, WebLogic Server provides implementations for most of these providers.

Web Tier Single Sign-On

With a growing focus on the Web as the user interface to applications, the Web tier is fast becoming the most common place where the requirement for single sign-on appears. In the Web tier, the user of a browser is prompted to authenticate their identity to the application. This identity is then propagated to the application server and utilized in the authentication of the user. The result of a successful sign-on is a cookie that is scope to the DNS domain in which the application server is resident. This cookie is then returned to the browser where it is sent with each future request to the application server.

As a J2EE 1.3 conferment application server, WebLogic Server supports all of the required mechanisms for authentication and single sign-on. These include:

- **Basic authentication**, where the user's credentials are only protected by a simple base64 encoding.
- **Form-based authentication**, where the credentials are not protected and sent in clear text.
- **Certificate-based authentication**, where X.509 certificates held by the client are used with the SSL or TLS secure protocol to establish the identity of the user.

The first two forms of authentication, basic and form-based, typically require the use of a secure transport, such as SSL or TLS, in order to protect the rather weak security utilized to protect the user's credentials.

In addition to the J2EE prescribed mechanisms, WebLogic Server can also be configured to support the use of additional authentication technology. This is possible through the use of the extensible WebLogic Security Framework. The use of the WebLogic Security Framework allows for the consistent enforcement of authentication policies regardless of whether the client is a browser or an application. This also removes the requirement for each application having to provide its own servlet filter with which to perform authentication. A further benefit is that this allows the type of authentication mechanisms, such as the use of a stronger authentication mechanism, to be changed without requiring changes in the application itself.

Single Sign-On Extensions

WebLogic Server goes beyond the J2EE 1.3 specification with its support for single sign-on by supporting fail over and load balancing to other members of a cluster without the need to re-authenticate to each cluster member. Furthermore, if the session has been configured for file or JDBC persistence, then the single sign-on solution can be extended even further to other non-clustered servers within the same DNS domain. Finally, it is possible to both disable single sign-on, as well as to create a group of different applications or Web components that participate in a single sign-on group by specifying a name, which is used to control the scope of a cookie within the `weblogic.xml` deployment descriptor file. The use of single sign-on groups provides the ability for a single DNS domain to be home to multiple applications, each of which can control which other applications or Web components are allowed to participate as part of its particular single sign-on environment.

Cross-Domain Single Sign-On

Although the J2EE 1.3 specification does not address the concept of a single sign-on environment that spans multiple DNS domains, WebLogic Server is uniquely designed to support this type of environment through partnerships with other security vendors. Cross-Domain single sign-on allows users to authenticate once but access multiple applications, even if these applications reside in different DNS domains. This provides the ability to construct a network of affiliates or partners that can participate in a single sign-on domain.

Through the use of the Security Framework contained in WebLogic Server, it is possible for standards-based solutions to be utilized to provide single sign-on integration at both the affiliate, as well as the global level. In particular, emerging standards and de facto standards such as the Security Assertions Markup Language (SAML) from Oasis and Internet-wide solutions such as Microsoft Passport and the result of the Liberty Alliance can be integrated with WebLogic Server to create an even broader single sign-on solution.

Beyond the Web Tier

In today's enterprise applications, it is rare that all the components that make up the application are all contained in the Web tier or are hosted on the same application server. As a result, it is critical that a single sign-on solution support integration of those components as part of the overall offering.

WebLogic Server provides a single sign-on solution that extends beyond the Web tier in order to incorporate integration with legacy systems and other J2EE application servers as part of its standard product features. The mechanisms used to support this functionality can be extended by customers, security vendors, or independent software vendors (ISVs) to provide enhanced capabilities.

Single Sign-On with Legacy Systems

As part of the single sign-on solution provided by WebLogic Server, application adapters defined by the J2EE Connector Architecture are able to acquire the credentials necessary to authenticate with the other components that make up the business solution. The credential acquisition capabilities are provided as part of the new security framework in WebLogic Server. Through this framework, the Connector container is able to retrieve the appropriate set of credential information for the target,

based on the information in the deployment descriptor, and the mapping contained in the one of the Credential Mapping providers. The credential information is then passed to the adapter, where it can be used to authenticate to the target.

A Credential Mapping provider is responsible for providing a mapping between the combination of the requestor's identity and the desired resource to the credential set appropriate for authenticating with the desired resource for that requestor. In order to allow for easier administration, the identity of the requestor used in the mapping can be either the username under which the requestor authenticated or the name of a group in which the requestor must be a member.

Each Credential Mapping provider can be associated with one or more credential formats, including Kerberos tickets, SAML name assertions, or others. The format of the credentials supported by a given Credential Mapping provider is registered with WebLogic Server at the time the provider is instantiated. BEA provides a default WebLogic Credential Mapping provider that can yield credentials in the form of username and password. This provider is included in the WebLogic Server and later distribution software.

Single Sign-On with Other J2EE Application Servers

The conformance with Level 0 of the Common Secure Interoperability (CSI) v2 specification allows WebLogic Server to participate in a secure, single sign-on environment with Enterprise JavaBeans (EJBs) hosted in other conferment J2EE application servers over the IIOP protocol.

WebLogic Server supports the required Generic Security Service Username/Password (GSSUP) authentication mechanism as a means to authenticate to another conferment J2EE application server, as well as the ability act as the target of such authentication. It can also support the use of Identity Tokens, as described in the CSIv2 specification, when configured in a trust relationship between the two containers.

Message-Driven Beans

Another aspect of single sign-on amongst J2EE application servers that can occur is when an application utilizes Message-Driven Beans with a foreign Java Message Service (JMS) provider, such as IBM MQueue Series. In this scenario, the WebLogic Server EJB container must authenticate itself to the foreign JMS implementation in order to retrieve the queued messages that are to be dispatched to the application code.

As with J2EE Connection adapters, WebLogic Server utilizes the Credential Mapping providers as a means to obtain the necessary credentials to authenticate with the foreign JMS provider. The use of the Credential Mapping provider allows the mapping between the foreign JMS provider and the requestor's identity to the appropriate credential set to be defined using the WebLogic Server Administration Console. In addition, because Credential Mapping providers can be written by customers and external vendors, custom implementations that contain enhanced mappings are possible.

CHAPTER

3 The WebLogic Security Service Architecture

The following sections describe the WebLogic Server Security architecture:

An Open Architecture: Multi-Vendor and Multi-Protocol Support

The open, interface-based, security architecture in BEA WebLogic Server allows use of existing security products while taking advantage of new security technologies available in the marketplace. With this architecture, a security installation can support security vendors' *full value proposition*s, not just a subset. A user's choice of security products can be "mixed and matched" to create complete custom security solutions. In fact, a WebLogic Server installation can run *more than one* security plug-in for a given function, and users can set constraints that govern which product or protocol will be used in a given situation.

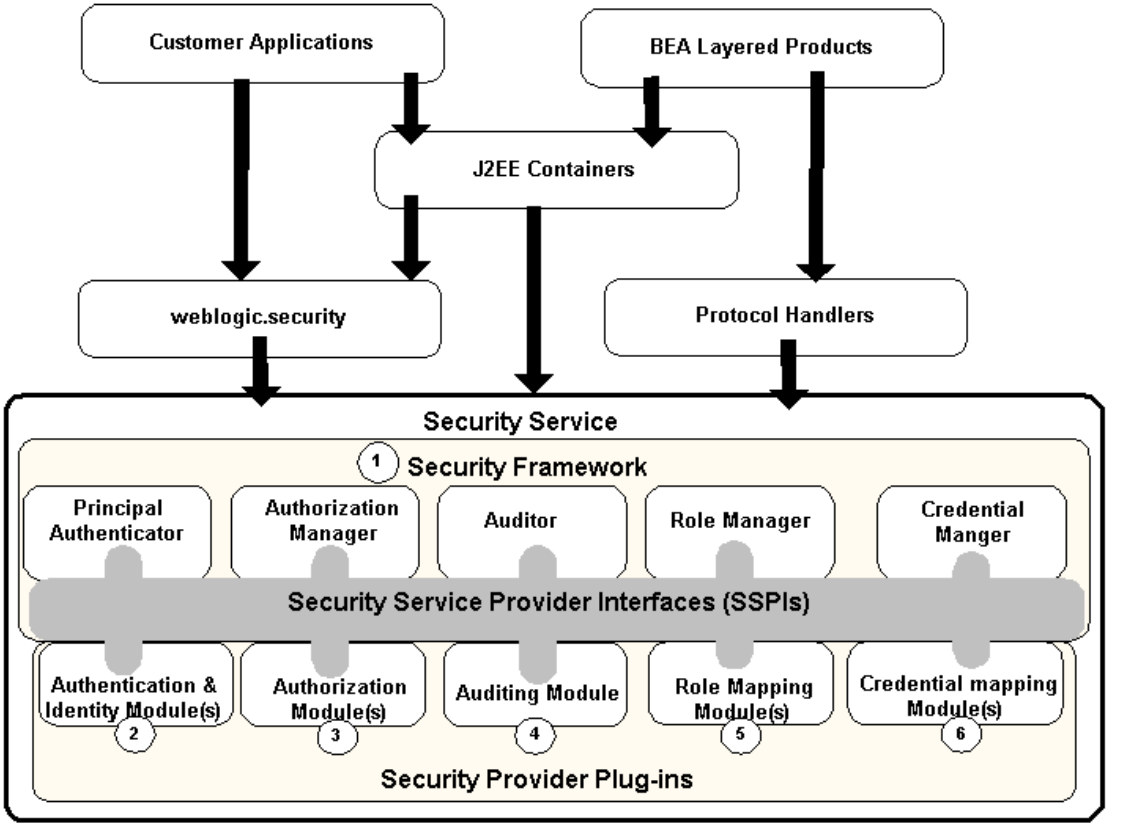
As users integrate new solutions or modify existing ones, administrators can set *security policy* for each security plug-in, using a built-in menu-driven policy tool (see Figure 1-1). Security policy governs authorization: the rules and constraints for accessing resources or assuming roles. More than one security plug-in can run concurrently, as part of a migration or transition scheme, and set security policy accordingly. The BEA-supplied Adjudicator function resolves any conflicts in interpretation when making authorization decisions.

The BEA WebLogic Server design for security services supports any choice of vendors and protocols because it cleanly separates the details of the security system from application code, simplifying application maintenance and management. Changing security system components or policies need not entail modifying applications. This unified architecture makes it easy to integrate best-of-breed security solutions, and to replace components of a security system with the latest technologies from third-party vendors, or from a development staff. The ability to swap in new security plug-ins and technologies, as needed, reduces the total cost of ownership and maximizes the return on investment in security technologies.

Architectural Overview

Figure 3-1 illustrates the architecture of the WebLogic Security Service.

Figure 3-1 WebLogic Security Service Architecture



1. Service-based framework that delegates requests to appropriate provider
2. Authenticates, verifies, and maps security tokens to internal format for single sign-on
3. Enforces authorization policies for resources that take business policy into consideration
4. Audits all security actions in support of non-repudiation
5. Maps roles to users/groups based on policy
6. Maps authentication credentials for a user to legacy application for single sing-on

Security Service Framework

Figure 3-1 shows the BEA WebLogic Server Security Service Framework. The primary function of the framework is to provide a simplified application programming interface (API) that can be used by WebLogic Server container developers can to define security services. Within that context, the framework also acts as an intermediary between the WebLogic Server containers and the security provider plug-ins.

The framework consists of the following components:

- Principal Authenticator
- Authorization Manager
- Role Manager
- Auditor
- Credential Manager

Principal Authenticator

The WebLogic Server containers call the Principal Authenticator to perform authentication via username and password. For example, a servlet container creates a callback handler to pass down the username and password and then calls authenticate in the Principal Authenticator. The principal authenticator creates a login context and calls login on the login context. The JAAS code calls the configured login modules. A subject is returned from the login context that the Principal authenticator uses it to perform auditing and user lockout functions.

The WebLogic Server containers can also call the Principal Authenticator to perform authentication via a certificate. For example, a servlet container has a certificate from a SSL connection. The container passes down the certificate and a token type of X509 to assertIdentity in the Principal Authenticator and gets back the subject. The principal authenticator finds the provider that has the token type active and gets the identity asserter for that provider. The principal authenticator then calls the provider's identity asserter. The identity asserter returns a callback handler which is used to retrieve the username that should be asserted. The principal authenticator uses the username to create a login context and calls login on the login context. The JAAS code calls the configured login modules. A subject is returned from the login context that the Principal authenticator uses to perform auditing and user lockout functions.

Authorization Manager

The WebLogic Server containers obtain a reference to the Authorization Manager by calling the Security Service Manager. With that reference the containers use either the `isAccessAllowed` or `isProtectedResource` method to call the Authorization Manager. The WebLogic Server containers use the Authorization Manager to determine whether a particular Subject has access to a particular Resource. For example, the container passes in authorization information to the Authorization Manager. The Authorization Manager uses the information to get a reference to the appropriate Authorization Provider that in turn holds a reference to the AccessDecision.

Additionally, the Authorization Manager performs the following functions:

- Provides mechanisms that enable WebLogic Server containers to deploy and undeploy policies.
- Checks to make sure that at least one Authorizer MBean is deployable.
- Attempts to get a reference to the Role Manager and throws an exception if it cannot.
- Attempts to get a reference to the Auditor. If a reference to an Auditor cannot be obtained, the Authorization Manager does not perform audits.

Role Manager

The WebLogic Server containers use the Role Manager and role mapping providers to determine which roles a Subject currently has with respect to a specific protected resource. The role mapping provider returns a role as a form of principal. The WebLogic Server containers obtain a reference to the Role Manager by calling the Security Service Manager. Once they have a reference to the Role Manager, the containers call the `RoleMapper.getRoles()` method on the role mapping provider. The `getRoles()` method returns a map of roles indexed by their names, representing the security roles associated with the specified resources that have been granted to the Subject.

WebLogic Server initializes the role mapping providers through the security service framework at boot time. The role mapping providers can also be initialized by an initialize call through the Role Manager or by the Administration Console

The Role Manager also attempts to get a reference to the Auditor. If a reference to an Auditor cannot be obtained, the Role Manager does not perform audits.

Auditor

When the server boots, during Security Service Manager initialization, if an Audit Provider is configured, the Auditor is instantiated.The Auditor provides a facility to record security events, with their pertinent data and outcomes. Logging of these events provides an audit trail with a purpose of non-repudiation.

The Auditor is called directly by security framework components, such as authentication and authorization to record date, time and event specific data. The Auditor can be called pre- and/or post-operationally. The Default security Audit Provider does not maintain context between pre- and post-operations.

The audit severity levels are as follows:

- Informational
- Warning
- Success
- Error
- Failure
- Audit_Failure

The Auditor is accessible directly via the security framework; a limited Auditor is accessible to security providers and management runtime MBeans. The Auditor is a back end for the Realm Adaptor Audit facility in Compatibility security. The Auditor fans out Auditor write operations (using the Security Provider Interface) to one or more configured Auditing Providers.

Credential Manager

The Credential Manager is used by WebLogic Server containers use to obtain Credential Mapping information. The Credential Manager makes calls through all the configured Credential Mapping providers to get all the relevant information. The Credential Manager is the only component that calls the Credential Mapping providers directly.

The Credential Manager uses the Credential Mapping provider to provide a specific type of credentials. Credential Mapping providers map a user in WebLogic Server to a proper set of credentials to be used to access some external system. For example, a Resource Adapter that plugs into WebLogic Server can provide connectivity between the WebLogic Server and an Enterprise

Information System (EIS). To make such connections possible, a Credential Mapping provider is implemented and configured to be used by the Credential Manager to map user credentials that the Resource Adapter can use to access the EIS. For example, the default Credential Mapping Provider that is provided as part of the WebLogic Server product is used to provide username/password credentials, and the provider uses the Embedded LDAP Server to store and manage the mappings.

Custom Credential Mapping providers may be created to provide other types of credentials, such as certificates or keys, for example. Also, a custom Credential Mapping provider could be created to store and manage username/password credentials outside of the WebLogic Server Embedded LDAP Server.

- Adapting existing third-party security technologies so that they are BEA-compliant.

The WebLogic Security Providers

This section provides descriptions of the WebLogic security providers that are included in the WebLogic Server product for your use. If the WebLogic security providers do not fully meet your security requirements, you can supplement or replace them. For more information, see *Developing Security Services for WebLogic Server*.

The following WebLogic security providers at included in the WebLogic Server product:

- WebLogic Authentication Provider
- WebLogic Identity Assertion Provider
- WebLogic Keystore Provider
- WebLogic Authorization Provider
- WebLogic Auditing Provider
- WebLogic Role Mapping Provider
- WebLogic Adjudication Provider
- WebLogic Credential Mapping Provider
- WebLogic Realm Adapter Providers

WebLogic Authentication Provider

The WebLogic Authentication provider supports delegated username/password and certificate authentication with WebLogic Server, and HTTP certificate authentication through external Web servers. The WebLogic Authentication provider utilizes an embedded LDAP server for the storage of user and group information.

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.

In addition, WebLogic Server provides as set of LDAP Authentication providers which access external LDAP stores (Open LDAP, Netscape iPlanet, Microsoft Active Directory, and Novell NDS).

WebLogic Identity Assertion Provider

Identity Assertion providers allow perimeter authentication. In perimeter authentication, a system outside of WebLogic Server establishes trust via tokens (as opposed to simple authentication whereby WebLogic Server establishes trust via usernames and passwords). An Identity Assertion provider verifies tokens and performs whatever actions are necessary to establish validity and trust in the token.

The WebLogic Identity Assertion provider validates X.509 and IIOP-CSIv2 tokens and maps a token to a WebLogic Server user.

WebLogic Keystore Provider

The WebLogic Keystore provider uses the reference Keystore implementation supplied by Sun Microsystems in the Java Development Kit. It utilizes the standard "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 CA Keystore file that it uses by default to locate the trusted CAs used by 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 only retrieves private keys from this keystore file, not certificates.

WebLogic Authorization Provider

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

WebLogic Auditing Provider

The WebLogic Auditing provider records information about security events, the data associated with a security events, and the outcome to a log file. The logging of security events provides an electronic trail of data that can be used for the purpose of non-repudiation. The WebLogic Auditing provider provides a customizable set of data for auditing security events such as failed login attempts, authentication requests, rejected digital certificates, and invalid roles.

The WebLogic Server Framework can call through to the WebLogic Auditing provider with a request before and after security operations (such as authentication or authorization) have been performed, enabling audit event recording. The decision by the WebLogic Auditing provider to audit a particular event is based on severity levels. During runtime, administrators can use the Administration Console to manage the WebLogic Auditing Provider's severity level.

WebLogic Role Mapping Provider

The WebLogic Role Mapping provider determines dynamic roles for a specific user (Subject) with respect to a specific protected resource. The provider maps roles to users or groups based on policy and determines the appropriate set of roles granted to a WebLogic Server user or group for a WebLogic resource. The role mapping provider returns a role as a form of principal.

Given a reference to the Role Manager, the WebLogic Server containers call the `RoleMapper.getRoles()` method on the role mapping provider. The `getRoles()` method returns a map of roles indexed by their names, representing the security roles associated with the specified resources that have been granted to the Subject.

WebLogic Server initializes the role mapping providers through the security service framework at boot time. The role mapping providers can also be initialized by an initialize call through the Role Manager or by the Administration Console

The WebLogic Role Mapping provider supports the deployment and undeployment of roles within the system. Using the Administration Console, administrators can enable or disable role deployment by the WebLogic Role Mapping provider. Role deployment is enabled by default.

The WebLogic Role Mapping provider uses the same policy engine as the WebLogic Authorization provider.

WebLogic Adjudication Provider

The WebLogic Adjudication provider is responsible for adjudicating between potentially differing decisions rendered by multiple Authorization providers. It is used when there are multiple Authorization providers. It adjudicates the results returned from multiple Authorization providers, that is, it examines the results returned and renders a final decision on whether or not access will be granted to a resource.

The default adjudication provider that is provided as part of the WebLogic Server product has an attribute, `RequireUnanimousPermit`, that governs its behavior. The status of this attribute is displayed on the Administration Console and can be set to true or false. By default, the `RequireUnanimousPermit` attribute is set to true, which means that all Authorization providers must return a Result of PERMIT in order for the WebLogic Adjudicator provider to return a `TRUE` verdict. In other words, if any Authorization providers return a verdict of either ABSTAIN or DENY, then the WebLogic Adjudicator provider will return a verdict of `FALSE`. If `RequireUnanimousPermit` is false and if all Results are either `PERMIT` or `ABSTAIN`, then the WebLogic Adjudicator provider will return `TRUE`. This means that only if at least one Authorization provider returns a verdict of DENY will the Adjudicator render a verdict of `FALSE`.

WebLogic Credential Mapping Provider

A Credential Mapping provider supports deploying policies on behalf of Resource Adapter deployment needs. Basically, credential mapping specifies which credential needs to be used by a certain WebLogic Server user when connecting to an Enterprise Information System (EIS) through a Resource Adapter. For example, the EIS may be a mainframe transaction processing, database systems, or legacy applications not written in the Java programming language.

When a Resource Adapter is deployed, credentials and mappings need to be created. Similarly, when the Resource Adapter is redeployed, the credentials and mappings need to be updated. And, when the Resource Adapter is undeployed, the credentials and mappings should be removed. This work is done by a Credential Mapping provider.

The default Credential Mapping provider that ships as part of the WebLogic Server product is used to associate, or map, a WebLogic Server user to the appropriate credentials to be used with a Resource Adapter plug-in to access an EIS system (for example, some relational database like Oracle, SQL Server, and so on.). The provider maps a user's authentication credentials to those required for legacy applications, so that the legacy application gets the necessary credential information. Using the Administration Console, administrators can enable or disable the deployment of the default Credential Mapping provider.

Credential Mappings are specific to a particular component (Resource Adapter) deployed within WebLogic Server. For example, WebLogic Server user "joeuser" may be mapped to the username/password credential "scott/tiger" for Component XYZ, and at the same time be mapped to "sa/admin" for Component ABC.

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. There are four WebLogic Realm Adapter providers:

- Authentication (includes an Identity Assertion provider)
- Authorization
- Auditing
- Adjudication

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.

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

The Realm Adapter Adjudication provider is designed to enable both the WebLogic Authorization provider and the Realm Adapter Authorization provider to be used together for a security realm in Compatibility security. In these cases, the WebLogic Authorization provider is used for new, security policies, while the Realm Adapter Authorization provider is used for mapping to 6.1 access control lists (ACLs).

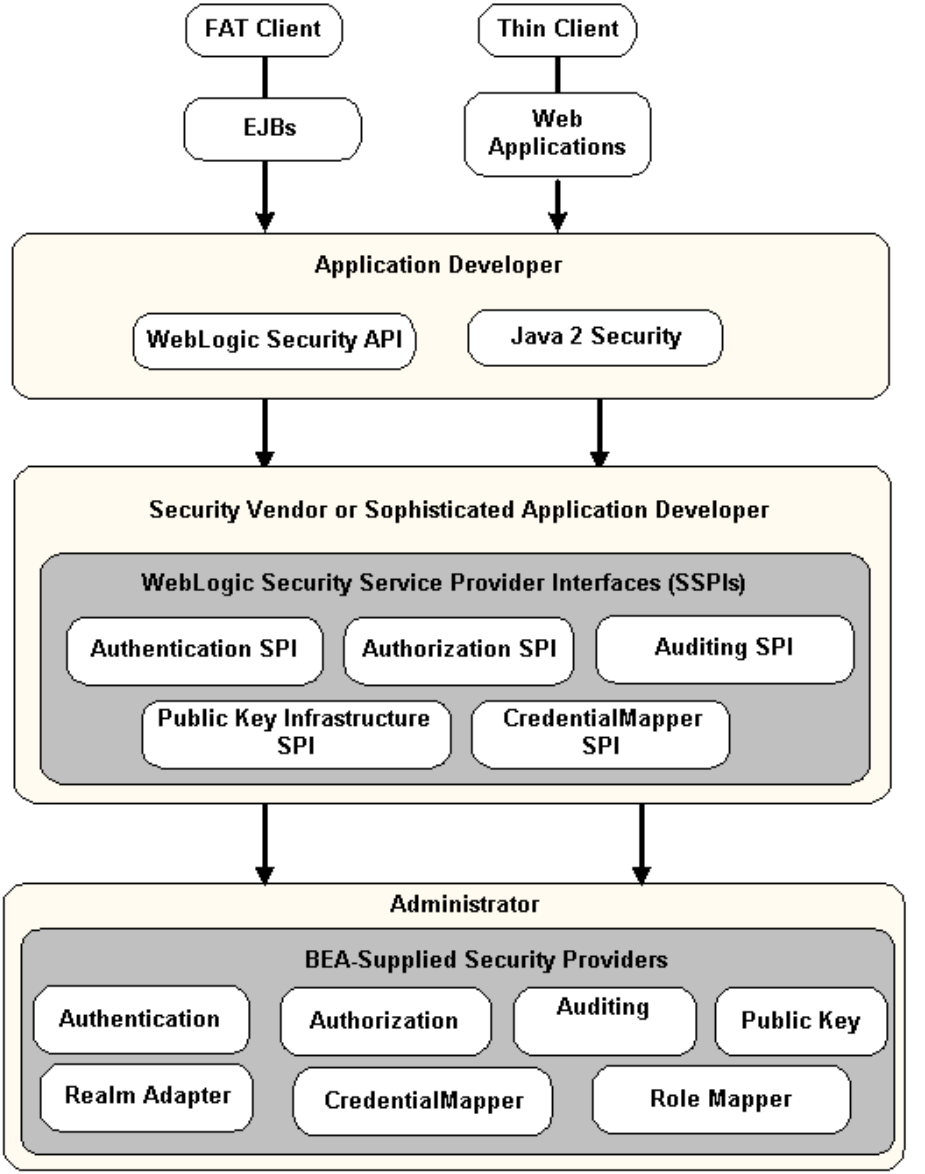
Although these security providers are configured using the WebLogic Server Administration Console, existing realms will continue to use the same MBeans and user interface present in WebLogic Server 6.1.

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

Advantages for Developers, Administrators, and Vendors

Figure 2 illustrates how different types of users would interact with the software architecture of the BEA WebLogic Server security services. The new security architecture has benefits for three categories of users: application developers, third-party security service vendors, and administrators.

Figure 3-2 Types of WebLogic Server Security Users



Benefits for Administrators

Administrators who install, configure, deploy and maintain BEA WebLogic Server can use their choice of BEA-supplied security plug-ins, customized security plug-ins, or third-party security products, and manage them all with the Administration Console.

Out of the box, a complete security solution can be implemented using the BEA-supplied security plug-ins. Administrators can use the menu-driven rule-based policy engine to create an authorization scheme that implements your company's business rules.

Benefits for Third-Party Security Service Providers

Most leading security service providers have announced plans to support BEA WebLogic Server 8.1. These providers are integrating their products with the WebLogic Server environment using the Security SPIs. As the underlying integration mechanism for BEA's security plug-ins, the Security SPIs permit development of customized security plug-ins for the WebLogic Server environment. Security SPIs are available for Authentication, Authorization, Auditing, Public Key Infrastructure, Credential Mapping, and Role Mapping. This allows third party vendors to provide tightly integrated solutions that are easy to implement. Reduced development requirements mean an increased return on investment when implementing an enterprise security management solution that includes WebLogic Server. Refer to the BEA Security Center (www.bea.com/products/security/index.shtml) for information on BEA-compliant security vendors.

Benefits for Application Developers

Since most security for Web applications 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. The 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.

CHAPTER

4 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 guard, or 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 this release of WebLogic Server, security policies replace the use of ACLs. ACLS are deprecated in this release of WebLogic Server, therefore, to continue to protect WebLogic resources with ACLs, you should use Compatibility security. See also permission (only used in 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, with specific parameters in an application. 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.

Adjudication provider and Adjudicator
: A security provider that tallies the results that multiple Access Decisions return, and determines the final `PERMIT` or `DENY` decision. The Adjudicator is a component of the Adjudication provider. It resolves conflicts between multiple Access Decisions by tallying each Access Decision and returning a final result. 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. 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 is typically done using username/password combinations, but may also be done using tokens. See also Authentication provider, LoginModule, Identity Assertion, perimeter authentication, token, user.

Authentication provider
: A security provider that enables 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. See also authentication, digital certificate, security provider, user.

authorization
: Process whereby the interactions between users and WebLogic resources are limited to ensure integrity, confidentiality, and availability. Authorization controls access to WebLogic resources based on user identity or other information. In this release of WebLogic Server, access to protected WebLogic resources is based on the context and target of the business request. See also Authorization provider, security policy, user, WebLogic resource.

Authorization provider
: A security provider that controls access to WebLogic resources based on user identity or other information. The WebLogic security architecture supports several types of authorization including parametric authorization, permissions-based authorization, and capabilities-based authorization. See also

capabilities-based authorization, parametric authorization, permissions-based authorization, security provider, user, WebLogic resource.

Caching realm

A WebLogic Server 6.x feature. A Caching realm is a temporary location in memory that contains frequently called ACLs, users, groups, etc. from the primary realm. A Caching realm is provided because in WebLogic Server 6.x, users, groups, and ACL objects are stored in a file (the file `realm.properties` file) and reading from a file can be very slow. The Caching realm sits as a communication layer on top of the primary realm and, by default, lookups are always done using the Caching realm first. If the Caching realm lookup fails, a lookup is performed on the primary realm. The Caching realm is only used in this release of WebLogic Server when you use Compatibility security. The use of the Caching realm is deprecated in this release. See also access control list (ACL), Compatibility security, group, user.

capabilities-based authorization

An arrangement whereby explicit permissions are given to a user (as in permissions-based authorization), but the objects and operations on those objects are also specified. For example, a user can be given permission to access a particular method of an Enterprise JavaBean (EJB). Thus, permissions may be specified at a high or low level. An example of capabilities-based authorization is the use of roles. See also Authorization provider, permissions-based authorization, security role, user.

certificate authority (CA)

Well-known and 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 digital certificate, entity, public key.

Compatibility realm

Security realm that is the default (active) security realm if you are using Compatibility security. The Compatibility realm uses your existing Authentication and Authorization providers and allows you to migrate to the providers in this release of WebLogic Server. The only security realm available in Compatibility security is the Compatibility realm. See also default realm, security provider, security realm, WebLogic security provider.

Compatibility security

Compatibility security refers to the capability to run security configurations from WebLogic Server 6.x in this release of WebLogic Server. In Compatibility security, you configure 6.x security realms, define users, groups,

and ACLs, manage the 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, group,user, Compatibility realm, security realm, Realm Adapter Authentication provider, Realm Adapter Authorization provider.

connection filter

A programmable filter that WebLogic Server uses to determine whether the server should allow incoming connections from network clients. Security policies allow you to secure WebLogic resources using some characteristic of a user, however, you can add an additional layer of security by using connection filters to filter network connections. See also security policy, user, WebLogic resource.

connector

See resource adapter

context handler

A high-performing WebLogic class that allows a variable number of arguments to be passed as strings to a method. The `ContextHandler` interface provides a way for an internal WebLogic container to pass additional information to a WebLogic Security Framework call, so that a security provider can obtain additional context information beyond what is provided by the arguments to a particular method. A context handler is essentially a name/value list and, as such, it requires a security provider to know what names to look for (that is, use of a context handler requires close cooperation between the WebLogic container and a 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, public key, subject.

credential mapping

Used by WebLogic Server for credentials used by a legacy or any remote system (for example, PeopleSoft, or Oracle). 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

A feature that allows users to authenticate once but access multiple applications, even if these applications reside in different DNS domains. This feature provides the ability to construct a network of affiliates or partners that can participate in a Single Sign-On domain.

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 J2EE1.3 containers correspond to Conformance Level 0 of the CSIv2 specification. The CORBA Security Attribute Service (SAS) is the protocol that is used in CSIv2. See `http://www.omg.org/technology/documents/formal/omg_security.htm` for more information.

custom security provider

Security provider written by a third-party security vendor or security developer that does not come with WebLogic Server. See also security provider, WebLogic security provider.

Custom security realm

In WebLogic Sever 6.x, if you wanted to customize authentication, you could write you own security realm and integrate it into the WebLogic Server environment. Additionally, in WebLogic Server 6.x, a security realm applied to a domain and you could only have one security realm in a domain. In this release of WebLogic Server, Custom security realms are only supported in 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 this release of WebLogic Server, 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, 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 (CA). 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 (CA), digital signature, public key, trusted (root) CA.

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 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.

dynamic role computation
: Late binding of principals to roles at runtime, which occurs just prior to an access decision for a protected WebLogic resource. See also Access Decision, authorization, principal, security role, WebLogic resource.

embedded LDAP server
: A server that contains user, group, 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 role, security policy.

entity
: Something that exists independently.

File realm

In WebLogic Server 6.x, a realm, which stores users, groups, encrypted passwords, and ACLs in a file. In this release, a File realm is used when running Compatibility security.

firewall

A protective perimeter against unauthorized access for a company's network that is connected to the Internet. Firewalls are also used within company networks to prevent unauthorized access. 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, WebLogic resource.

group

Collection of users that share some characteristics. 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 client or a WebLogic Server 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 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. See also digital certificate, host name verification, Secure Sockets Layer (SSL).

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 generating 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. Thus, identity assertion can be used to enable single sign-on. See also authentication, digital certificate, Identity Assertion provider, 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.

Java Authentication and Authorization Service (JAAS)

Set of packages that enable services to authenticate and enforce access controls upon users. It 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.

JAAS control flag

If a security realm has multiple Authentication providers configured, the Control Flag determines how the login sequence uses the Authentication provider. 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 Cryptography Architecture

A framework for accessing and developing cryptographic functionality for the Java platform. See

http://java.sun.com/j2se/1.3/docs/guide/security/CryptoSpec.html#Introduction for a description of JCA provided by Sun Microsystems, Inc. See also Java Cryptography Extensions (JCE)

Java Cryptography Extensions (JCE)

Extends the Java Cryptography Architecture API to include APIs for encryption, key exchange, and Message Authentication Code (MAC). See http://java.sun.com/j2se/1.3/docs/guide/security/CryptoSpec.html#Introduction for a description of JCE provided by Sun Microsystems, Inc. See also Java Cryptography Architecture.

Java Security Manager

A Java 2 security feature that prevents untrusted code from performing actions that are restricted by the Java security policy file. WebLogic Server supports the use of the Java Security Manager. The Java virtual machine (JVM) has security mechanisms built into it that allow you to define restrictions to code through a 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 not permit certain runtime operations. See also Java security policy file, permission (only used in Compatibility security).

Java security policy file

The Java Virtual Machine (JVM) has security mechanisms built into it that allow you to define restrictions to code through a 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 not permit certain runtime operations. See also Java Security Manager, permission (only used in Compatibility security).

keystore

A repository that contains a list of trusted certificate authorities and your private key and certificate pairs. The information is stored in PKCS12 format and is protected by a “passphrase.” In this release, the keystool utility is supported, however, server certificates must also be stored in, and accessed from, a flat file because accessing certificates from the keystore is not supported. See also private key, trusted (root) CA.

LDAP Authentication provider

Authentication provider that utilizes an 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. You can choose from the File realm or a set of alternative security realms including the Lightweight Data Access Protocol (LDAP), Windows NT, Unix, or RDBMS realms. The LDAP security realm provides authentication through a Lightweight Directory Access Protocol (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 this release, 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 your security providers. See also MBean type, security provider.

MBean Definition File (MDF)

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 and are installed into WebLogic Server. 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 (comment: needs more work)

A digitally created hash, or fingerprint, from which it was created. Message digests are very useful in helping to prevent man-in-the-middle attacks. Because there is only one digest for any given plain text, the digest can be used to verify the authenticity of the message. See also message digest algorithm (comment: needs more work).

message digest algorithm (comment: needs more work)

An algorithm that is used to produce a message digest of the bulk text of a message. Once a message digest is produced, other security mechanisms are used to encrypt and convey the digest. This process results in a digital signature of the message, which is used to provide non-repudiation and integrity services. See also message digest (comment: needs more work).

mutual authentication

Authentication that requires that both the client and the server present proof of identity. Two-way SSL authentication is a form of mutual authentication. See also authentication, digital certificate, trusted (root) CA, Secure Sockets Layer (SSL), two-way SSL authentication.

non-repudiation

Irrefutable evidence that a security event occurred.

one-way SSL authentication

SSL authentication can be one-way or two-way. With one-way SSL authentication, which WebLogic Server enables by default, the server is required to present a certificate to the client, but the client is not required to

present a certificate to the server. To successfully negotiate an SSL connection, the client must authenticate the server, but the server will accept any client into the connection. See also mutual authentication.

parametric authorization
: Authorization whereby access decisions on protected WebLogic resources take into account the context and target of the business request. See also authorization, WebLogic resource.

perimeter authentication
: Authentication that occurs outside of the application server. 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 pass phase, 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), the 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. The WebLogic Server security architecture supports Identity Assertion providers that perform perimeter authentication (Web server, firewall, VPN) and handle multiple security token types/protocols (SOAP, IIOP-CSIv2). See also authentication, Identity Assertion.

permission (only used in Compatibility security)
: In WebLogic Server 6.x, the means used by ACLs to grant or deny users and groups access to WebLogic resources. In this release of WebLogic Server, ACLs are used only when Compatibility security is used and when not using Compatibility security, roles are used instead of permissions to define who has permission to access a WebLogic resource and under what conditions. See also access control list (ACL), Compatibility security, WebLogic resource.

permissions-based authorization
: Authorization whereby explicit permissions are given to a user via security policies. Java 2 Enterprise Edition (J2EE) security uses this type of authorization by obtaining data from the `weblogic.policy` file. Although deprecated in this release of WebLogic Server, access control lists (ACLs) are an example of permissions-based authorization. See also access control list (ACL), Authorization provider, security policy.

policy statement
: An expression that specifies the users, groups, or roles on which the security policy is based.

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. See also public key.

private key algorithm
: The algorithm used to decode, or decrypt ciphertext. It is called private because it must be kept secret from everyone but the user. See also private key, public key, public key algorithm.

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

public key
: Value provided by some designated authority as an encryption key that, combined with a private key derived from the public key, can be used to effectively encrypt 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 private key.

public key cryptography
: See asymmetric key cryptography.

public key algorithm
: The algorithm used to encode, or encrypt, plain text. Only the private key can decrypt the ciphertext. See also private key, private key algorithm, public key.

RDBMS security realm
: A WebLogic Server 6.x security realm. In WebLogic Server 6.x, security realms provided authentication and authorization services. You can choose from the File realm or a set of alternative security realms including the Lightweight Data Access Protocol (LDAP), Windows NT, Unix, or RDBMS realms. The RDBMS Security realm is a BEA-provided custom security realm that stores Users, Groups and ACLs in a relational database. In this release of WebLogic Server, 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 Auditing provider
: To be supplied.

Realm Adapter provider
: Type of security provider used to access WebLogic Server 6.x security services when using Compatibility security in this release of WebLogic Server.

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.

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.

resource
: See WebLogic resource.

resource adapter
: System-level software driver (also called a connector) used by an application server such as WebLogic Server to connect to an enterprise information system (EIS).

role mapping
: Process whereby the groups and/or principals recognized by the EJB or Web container are associated with the security roles specified in a deployment descriptor. See also group, principal, security role.

Role Mapping provider
: A security provider that determines what 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.

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 Interface (SSPI).

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

secret key cryptography
: See symmetric key cryptography.

Security Assertion Markup Language (SAML)
: An XML-based framework for exchanging security information. SAML provides a standard way to profile information in XML documents and to define authentication and authorization. SAML implementations provide an interoperable XML-based security solution, where user information and corresponding authentication or authorization information can be exchanged by collaborating services irrespective of their existing security implementations. SAML is the key to enabling single sign-on in Web services. For a discussion of SAML, 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, user.

Secure Sockets Layer (SSL)
: Secure Sockets Layer (SSL) implements cryptography on the Web. SSL supports the use of public key cryptography to provide authentication, and secret key cryptography and digital signatures to provide for privacy and data integrity. Generally, the 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. See also authentication, digital signature, public key cryptography, symmetric key cryptography.

security policy

In the previous releases of WebLogic Server, ACLs were used to protect WebLogic resources. In this release of WebLogic Server, security policies are used instead of ACLs in protecting WebLogic resources. A security policy is created when you define an association between a WebLogic resource and one or more users, groups, or roles. A security policy may also include time constraints, that is, periods of time when the resource may be accessed. A WebLogic resource has no protection until you assign it a security policy. You assign security policies to an individual WebLogic resource or components of the WebLogic resource. See also access control list (ACL), group, security role, user, WebLogic resource.

security provider

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 may be categorized as WebLogic security providers and custom security providers. See also custom security provider, Security Service Provider Interface (SSPI), MBean, MBean type, runtime class, 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 provided authentication and authorization services. You chose from 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 wanted to customize authentication, you could write you own security realm and integrate it into the

WebLogic Server environment. A security realm applied to a domain and you could not have multiple security realms in a domain. See also File realm.

In this release of WebLogic Server, 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 in this release of WebLogic Server; rather, you configure a new security realm to provide the security services you want and then set the new security realm as the default security realm. See also Compatibility realm, default realm, Domain Configuration Wizard, security provider, WebLogic resource.

security role

Abstract, logical collections of users similar to a group. 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. 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 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 dynamic role computation, global role, group, scoped role, security policy, user, WebLogic resource.

Security Service Provider Interface (SSPI)

Interfaces implemented by security providers, both BEA and custom providers. The WebLogic Security Framework calls methods in these interfaces to perform the security operation. See also security provider, WebLogic Security Framework.

single sign-on

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, Identity Assertion, LoginModule, token, user.

SSL hardware accelerator

A peripheral Secure Socket 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 MBean type, custom security provider, WebLogic security provider.

subject

Container for authentication information, including principals, as specified by the Java Authentication and Authorization Service (JAAS). Both users and groups can be used as principals by application servers like WebLogic Server. A subject can contain any number of principals. See also authentication, group, Java Authentication and Authorization Service (JAAS), 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 used as part of the authentication process. When using Identify Assertion, a token must be presented to determine whether the user has been authenticated. Tokens come in many different types, including Kerberos and SAML. See also authentication, Identity Assertion, SSL tunneling, 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) CA

A 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 ceritificate authority (CA) is similar to that of a Notory Republic: to guarantee the identify of the individual or organization presenting the certificate. Thus, trusted CAs issue certificates that are used to sign other certificates. CAs are rerferred to as root CAs because their authority is recognized and, therefore, they do not need anyone to validate their identity. Trusted (root) CA certificates are installed into applications that authenticate certificates. For example, Web browsers are usually distributed with several common trusted (root) CAs certificates pre-installed. See also private key, public key.

two-way SSL authentication

Authentication that requires both the client and server 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, trusted (root) CA, Secure Sockets Layer (SSL).

user

Entities that use WebLogic Server, such as application end users, client applications, and other instances of WebLogic Server. Users may be placed into groups that are associated with roles, or be directly associated with roles. See also entity, group, security role.

UNIX security realm

A WebLogic Server 6.x security realm. In WebLogic Server 6.x, security realms provided authentication and authorization services. You can choose from the File realm or a set of alternative security realms including the Lightweight Data Access Protocol (LDAP), Windows NT, Unix, or RDBMS realms. 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 this release of WebLogic Server, you can only use the UNIX security realm when using Compatibility security. See also authentication, authorization, Compatibility security, group, security realm.

WebLogic container
To be supplied.

WebLogic MBeanMaker
Command-line utility that takes an MBean Definition File (MDF) as input and outputs 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
A security provider supplied by BEA Systems as part of the WebLogic Server product. See also custom security provider, security provider.

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. In WebLogic Server 6.x, security realms provided authentication and authorization services. You can choose from the File realm or a set of alternative security realms including the Lightweight Data Access Protocol (LDAP), Windows NT, Unix, or RDBMS realms. The Windows NT Security realm uses account information defined for a Windows NT domain to authenticate Users and Groups. In this release of WebLogic Server, you can only use the Windows NT security realm when using Compatibility security. See also authentication, authorization, Compatibility security, group, security realm, user.