BEA WebLogic Server™

Securing WebLogic Server

Version 9.0 BETA
Document Revised: December 15, 2004

Copyright

Restricted Rights Legend

Trademarks or Service Marks

Contents

4. Configuring WebLogic Security Providers

8. Configuring SSL

11.Using Compatibility Security

CHAPTER 1

Introduction and Roadmap

The following sections describe the contents and organization of this guide—*Securing WebLogic Server*.

Document Scope

This document explains how to configure security for WebLogic Server and how to use Compatibility security.

Document Audience

This document is intended for the following audiences:

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

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

Guide to this Document

This document is organized as follows:

- Chapter 2, "Overview of Security Management," explains the differences between security in previous releases of WebLogic Server and this release of WebLogic Server; describes the default security configuration in WebLogic Server; lists the configuration steps for security, and describes Compatibilty security.

- Chapter 3, "Customizing the Default Security Configuration," explains when to customize the default security configuration, the configuration requirements for a new security realm, and how to set a security realm as the default security realm.
- Chapter 4, "Configuring WebLogic Security Providers," describes the available configuration options for the security providers supplied by WebLogic Server and how to configure a custom security provider.
- Chapter 5, "Migrating Security Data," provides information about migrating security data between security realms and security providers.
- Chapter 6, "Managing the Embedded LDAP Server," describes the management tasks associated with the embedded LDAP server used by the WebLogic security providers.
- Chapter 7, "Configuring Identity and Trust," describes how to configure identity and trust for WebLogic Server.
- Chapter 8, "Configuring SSL," describes how to configure SSL for WebLogic Server.
- Chapter 9, "Configuring SSL for Node Manager," describes the SSL requirements for the Node Manager and describes two scenarios: using the demonstration identity and trust provided by WebLogic Server and using production-quality identity and trust.
- Chapter 10, "Configuring Security for a WebLogic Domain," describes how to set security configuration options for a WebLogic domain.
- Chapter 11, "Using Compatibility Security,"describes how to use Compatibility security.

Related Information

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

- *Understanding WebLogic Security*—This 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.
- *Developing Security Providers for WebLogic Server*—This document provides security vendors and application developers with the information needed to develop custom security providers that can be used with WebLogic Server.
- *Securing a Production Environment*—This document highlights essential security measures for you to consider before you deploy WebLogic Server into a production environment.

- *Securing WebLogic Resources*—This document introduces the various types of WebLogic resources, and provides information that allows you to secure these resources using WebLogic Server. The current version of this document primarily focuses on securing URL (Web) and Enterprise JavaBean (EJB) resources.

- *WebLogic Server Upgrade Guide*—This document provides procedures and other information you need to upgrade 6.x and earlier versions of WebLogic Server to this release of WebLogic Server 8.1. It also provides information about moving applications from a 6.x or earlier version of WebLogic Server to this release. For specific information on upgrading WebLogic Server security, see *Security* in the *WebLogic Server Upgrade Guide*.

- *Security FAQ*—This document gives answers to frequently asked questions about WebLogic Server security.

- *Javadocs for WebLogic Classes*—This document provides reference documentation for the WebLogic security packages that are provided with and supported by this release of WebLogic Server.

Security Samples and Tutorials

TBD

CHAPTER 2

Overview of Security Management

The following sections provide an overview of the security system for WebLogic Server, including the differences between the 6.x release of WebLogic Server and this release of WebLogic Server.

Note: Throughout this document, the term *6.x* refers to WebLogic Server 6.0 and 6.1 and their associated Service Packs.

How Security Changed in WebLogic Server

The security service in WebLogic Server simplifies the configuration and management of security while offering robust capabilities for securing your WebLogic Server deployment. This section describes how the security service changed from previous releases of WebLogic Server.

Change in Scope of Security Realms

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.

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, security 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*—Has the WebLogic Adjudication, Authentication, Identity Assertion, Authorization, Role Mapping, and Credential Mapping providers configured by default.
- *CompatibilityRealm*—Provides backward compatibility for 6.x security configurations. You can access an existing 6.x security configuration through the *CompatibilityRealm*.

Custom security realms written using the security application programming interfaces (APIs) are only supported in Compatibilty security. In this release of WebLogic Server, you customize authentication and authorization functions by configuring a new security realm to provide the security services you want and then set the new security realm as the default security realm.

For information about the default security configuration in WebLogic Server, see "The Default Security Configuration in WebLogic Server" on page 2-7.

For information about configuring a security realm and setting it as the default security realm, see Chapter 3, "Customizing the Default Security Configuration."

For information about using Compatibility security, see Chapter 11, "Using Compatibility Security."

Security Providers

Security providers are modular components that handle specific aspects of security, such as authentication and authorization. Although applications can leverage the services offered via the default WebLogic security providers, the WebLogic Security Service's flexible infrastructure also allows security vendors to write their own custom security providers for use with WebLogic Server. WebLogic security providers and custom security providers can be mixed and matched to create unique security solutions, allowing organizations to take advantage of new technology advances in some areas while retaining proven methods in others. The WebLogic Server Administration Console allows you to administer and manage all your security providers through one unified management interface.

The WebLogic Security Service supports the following types of security providers:

- **Authentication**—Authentication is the 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. The WebLogic Security Service supports the following types of authentication:
  - Username and password authentication
  - Certificate-based authentication directly with WebLogic Server
  - HTTP certificate-based authentication proxied through an external Web server

  Authentication providers supply these services.
- **Authorization**—Authorization is the process whereby the interactions between users and WebLogic resources are limited to ensure integrity, confidentiality, and availability. In other words, authorization is responsible for controlling access to WebLogic resources based on user identity or other information. An Authorization provider supplies these services.
- **Role Mapping**—Obtains a computed set of roles granted to a requestor for a given resource. Role Mapping providers supply Authorization providers with this information so that the Authorization provider can answer the "is access allowed?" question for WebLogic resources that use role-based security (for example, Web applications and Enterprise JavaBeans (EJBs). A Role Mapping supplies these services.
- **Identity Assertion**—An Authentication provider that performs perimeter authentication—a special type of authentication using tokens— is called an Identity Assertion provider. Identity assertion involves establishing a client's identity through the use of client-supplied tokens that may exist outside of the request. Thus, the function of an Identity Assertion provider is to validate and map a token to a username. Once this mapping is complete, an Authentication provider's LoginModule can be used to convert the username to principals. Identity Assertion providers supply these services.
- **Auditing**—Auditing is the process whereby information about security requests and the outcome of those security requests are collected, stored, and distributed for the purpose of non-repudiation. In other words, auditing provides an electronic trail of computer activity. An Auditing provider supplies these services.
- **Adjudication**—When multiple Authorization providers are configured in a security realm, each may return a different answer to the "is access allowed" question for a given resource. Determining what to do if multiple Authorization providers do not agree is the primary function of an Adjudication provider. Adjudication providers resolve authorization conflicts by weighing each Authorization provider's answer and returning a final decision.

- **Credential Mapping**—A credential map is a mapping of credentials used by WebLogic Server to credentials used in a legacy or 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. Credential Mapping providers map credentials in this way.
- **Keystore**—A keystore is a mechanism for creating and managing password-protected stores of private keys and certificates for trusted certificate authorities. The keystore is available to applications that may need it for authentication or signing purposes. In the WebLogic Server security architecture, the WebLogic Keystore provider is used to access keystores.

  Note: The WebLogic Server Keystore provider is deprecated in this release of WebLogic Server and is only supported for backward compatibility. Use keystores instead. For more information about configuring keystores, see "Configuring Keystores For Production" on page 7-13.

- **Certificate Lookup and Validation (CLV)**—X.509 certificates need to be located and validated for purposes of identity and trust. CLV providers receive certificates, certificate chains, or certificate references, complete the certificate path (if necessary), and validate all the certificates in the path. There are two types of CLV providers:
  - A CertPath Builder looks up and optionally completes the certificate path and validates the certificates.
  - CertPath Validator looks up and optionally completes the certificate path, validates the certificates, and performs extra validation (for example, revocation checking).
- **Certificate Registry**—A certificate registry is a mechanism for adding revocation checking to a security realm. The registry stores a list of valid certificates. Only registered certificates are valid. A certificate is revoked by removing it from the certificate registry. The registry is stored in the embedded LDAP server. The Certificate Registry is both a CertPath Builder and a CertPath Validator.

For information about the functionality provided by the WebLogic security providers, see Chapter 4, "Configuring WebLogic Security Providers."

For information about the default security configuration, see "The Default Security Configuration in WebLogic Server" on page 2-7.

For information about writing a custom security provider, see *Developing Security Providers for WebLogic Server*.

Security Policies Instead of ACLs

In WebLogic Server 6.x, access control lists (ACLs) and permissions were used to protect WebLogic resources. In this release of WebLogic Server, security policies replace ACLs and permissions. Security policies 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 a user, group, or security role. You can also optionally associate a time constraint with a security policy. A WebLogic resource has no protection until you assign it a security policy.

Creating security policies is a multi-step process with many options. To fully understand this process, read *Securing WebLogic Resources*. This document should be used in conjunction with *Managing WebLogic Security* to ensure security is completely configured for a WebLogic Server deployment.

WebLogic Resources

A WebLogic resource is a structured object used to represent an underlying WebLogic Server entity, which can be protected from unauthorized access. WebLogic Server defines the following resources:

- Administrative resources such as the WebLogic Server Administration Console and the `weblogic.Admin` tool.
- Application resources that represent Enterprise applications. This type of resource includes individual EAR (Enterprise Application aRchive) files and individual components, such as EJB JAR files contained within the EAR.
- Component Object Model (COM) resources that are designed as program component objects according to Microsoft's framework. This type of resource includes COM components accessed through BEA's bi-directional COM-Java (jCOM) bridging tool.
- Enterprise Information System (EIS) resources that are designed as connectors, which allow the integration of Java applications with existing enterprise information systems. These connectors are also known as resource adapters.
- Enterprise JavaBean (EJB) resources including EJB JAR files, individual EJBs within an EJB JAR, and individual methods on an EJB.
- Java DataBase Connectivity (JDBC) resources including groups of connection pools, individual connection pools, and multipools.
- Java Naming and Directory Interface (JNDI) resources.

- Java Messaging Service (JMS) resources.
- Server resources related to WebLogic Server instances, or servers. This type of resource includes operations that start, shut down, lock, or unlock servers.
- URL resources related to Web applications. This type of resource can be a Web Application aRchive (WAR) file or individual components of a Web application (such as servlets and JSPs).

  Note: Web resources are deprecated in this release of WebLogic Server. Use the URL resource instead.

- Web Services resources related to services that can be shared by and used as components of distributed, Web-based applications. This type of resource can be an entire Web service or individual components of a Web service (such as a stateless session EJB, particular methods in that EJB, the Web application that contains the `web-services.xml` file, and so on).
- Remote resource

Deployment Descriptors and the WebLogic Server Administration Console

The WebLogic Security Service can use information defined in deployment descriptors to grant security roles and define security policies for Web applications and EJBs. When WebLogic Server is booted for the first time, security role and security policy information stored in `web.xml`, `weblogic.xml`, `ejb-jar.xml`, or `weblogic-ejb-jar.xml` deployment descriptors is loaded into the Authorization and Role Mapping providers configured in the default security realm. Changes to the information can then be made through the WebLogic Server Administration Console.

To use information in deployment descriptors, at least one Authorization and Role Mapping provider in the security realm must implement the `DeployableAuthorizationProvider`, and `DeployableRoleProvider` Security Service Provider Interface (SSPI). This SSPI allows the providers to store (rather than retrieve) information from deployment descriptors. By default, the WebLogic Authorization and Role Mapping providers implement this SSPI.

If you change security role and security policy in deployment descriptors through the WebLogic Server Administration Console and want to continue to modify this information through the WebLogic Server Administration Console, you can set configuration options on the security realm to ensure changes made through the WebLogic Server Administration Console are not overwritten by old information in the deployment descriptors when WebLogic Server is rebooted.

For more information, see *Securing WebLogic Resources*.

The Default Security Configuration in WebLogic Server

To simplify the configuration and management of security in WebLogic Server, a default security configuration is provided. In the default security configuration, *myrealm* is set as the default security realm and the WebLogic Adjudication, Authentication, Identity Assertion, Authorization, Credential Mapping, and Role Mapping, CertPath providers are defined as the security providers. To use the default security configuration, you need to define users, groups, and security roles for the security realm, and create security policies to protect the WebLogic resources in the domain.

For a description of the functionality provided by the WebLogic Security providers, see the *Understanding WebLogic Security*. 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*.

If the default security configuration does not meet your requirements, you can create a new security realm with any combination of WebLogic and custom security providers and then set the new security realm as the default security realm. For more information, see Chapter 3, "Customizing the Default Security Configuration."

Configuration Steps for Security

Because the security features are closely related, it is difficult to determine where to start when configuring security. In fact, configuring security for your WebLogic Server deployment may be an iterative process. Although more than one sequence of steps may work, BEA Systems recommends the following procedure:

1. Determine whether or not to use the default security configuration by reading "Why Customize the Default Security Configuration?" on page 3-1
   - If you are using the default security configuration, begin at step 3.
   - If you are not using the default security configuration, begin at step 2.
2. Change the configuration of the security providers (for example, configure an LDAP Authentication provider instead of using the WebLogic Authentication provider) or configure custom security providers in the default security realm. This step is optional. By default, WebLogic Server configures the WebLogic security providers in the default

security realm (*myrealm*). For information about the circumstances that require you to customize the default security configuration, see "Why Customize the Default Security Configuration?" on page 3-1.

Note: You can also create a new security realm, configure security providers (either WebLogic or custom) in the security realm and set the new security realm as the default security realm. See Chapter 3, "Customizing the Default Security Configuration."

3. Optionally, configure the embedded LDAP server. By default, options for the embedded LDAP server are configured. However, you may want to change those options to optimize the use of the embedded LDAP server in your environment. For more information, see Chapter 6, "Managing the Embedded LDAP Server."
4. Ensure user accounts are properly secured. WebLogic Server provides a set of configuration options for protecting user accounts. By default, they are set for maximum security. However, during the deployment of WebLogic Server you may need to lessen the restrictions on user accounts. Before moving to production, check that the options on user accounts are set for maximum protection. If you are creating a new security realm, you need to set the user lockout options. For more information, see "How Passwords are Protected in WebLogic Server" on page 10-4 and "Protecting User Accounts" on page 10-5.
5. Protect WebLogic resources with security policies. Creating security policies is a multi-step process with many options. To fully understand this process, read *Securing WebLogic Resources*. This document should be used in conjunction with *Securing WebLogic Server* to ensure security is completely configured for a WebLogic Server deployment.
6. Configure identity and trust for WebLogic Server. (This step is optional but encouraged.) For more information, see Chapter 7, "Configuring Identity and Trust."
7. Enable SSL for WebLogic Server. (This step is optional but encouraged.) For more information, see Chapter 8, "Configuring SSL."

In addition, you can:

- Configure a connection filter. See "Using Connection Filters" on page 10-3.
- Enable interoperability between WebLogic domains. See "Enabling Trust Between WebLogic Server Domains" on page 10-1.

What Is 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 manage 6.x security

realms, users, groups, and ACLs, protect user accounts, and configure the Realm Adapter Auditing provider and optionally the Identity Assertion provider in the Realm Adapter Authentication provider.

The only security realm available in Compatibility security is the *CompatibilityRealm*. The Realm Adapter providers (Auditing, Adjudication, Authorization, and Authentication) in the Compatibility realm allow backward compatibility to the authentication, authorization, and auditing services in 6.x security realms. For more information, see Chapter 10, "Using Compatibility Security."

Management Tasks Available in Compatibility Security

Because Compatibility security only allows you to access authentication, authorization, and custom auditing implementations supported in WebLogic Server 6.x, not all 6.x security tasks are allowed in Compatibility security. Use Compatibility security to:

1. Configure the Realm Adapter Auditing provider. For more information, see "Configuring a Realm Adapter Auditing Provider" on page 10-4.
2. Configure the Identity Assertion provider in the Realm Adapter Authentication provider so that implementations of the `weblogic.security.acl.CertAuthenticator` class can be used. For more information, see "Configuring the Identity Assertion Provider in the Realm Adapter Authentication Provider" on page 10-3.

   Note: The Realm Adapter Adjudication and Authorization providers are configured by default in the *CompatibilityRealm* using information in an 6.x existing `config.xml` file. These providers can only be used in the *CompatibilityRealm*. The Realm Adapter Authentication provider is also automatically configured in the *CompatibilityRealm*. However, this provider can also be configured in other realms to provide access to users and groups stored in 6.x security realms. For more information, see "Configuring a Realm Adapter Authentication Provider" on page 4-16.
3. Change the password of the `system` user to protect your WebLogic Server deployment.
4. Manage the security realm in the *CompatibilityRealm*.
5. Define additional users for the security realm in the *CompatibilityRealm*. Organize users further by implementing groups in the security realm.
6. Manage ACLs and permissions for the resources in your WebLogic Server deployment.
7. Create security roles and security policies for WebLogic resources you add to the *CompatibilityRealm*. For more information, see *Securing WebLogic Resources*.

You can still configure identity and trust, use SSL, configure connection filters, and enable interoperability between domains; however, you use the security features available in this release of WebLogic Server to perform these tasks. For more information, see:

- Chapter 7, “Configuring Identity and Trust”
- Chapter 8, “Configuring SSL”
- Chapter 10, “Configuring Security for a WebLogic Domain”

CHAPTER 3

Customizing the Default Security Configuration

The following sections provide information about customizing the default security configuration and creating a new security realm:

For information about configuring security providers, see Chapter 4, "Configuring WebLogic Security Providers."

For information about migrating security data to a new security realm, see Chapter 5, "Migrating Security Data."

Why Customize the Default Security Configuration?

To simplify the configuration and management of security, WebLogic Server provides a default security configuration. In the default security configuration *myrealm* is set as the default (active) security realm, and the WebLogic Adjudication, Authentication, Identity Assertion, Authorization, Credential Mapping, Role Mapping, CertPath providers are defined as the security providers.

The easiest way to customize the default security configuration is to add the security providers you want to the default security realm (*myrealm*). Customize the default security configuration if you want to:

- Replace one of the WebLogic security providers with a custom security provider.

- Configure additional security providers in the default security realm. (For example, if you want to use two Authentication providers, one that uses the embedded LDAP server and one that uses a Windows NT store of users and groups.)
- Use an Authentication provider that accesses an LDAP server other than the embedded LDAP server.
- Use an existing store of users and groups (for example, a DBMS database) instead of defining users and groups in the WebLogic Authentication provider.
- Add an Auditing provider to the default security realm.
- Use an Identity Assertion provider that handles SAML assertions or Kerberos tokens.
- Use the Cert Registry to add revocation to the security realm.
- Change the default configuration settings of the security providers. See Chapter 4, "Configuring WebLogic Security Providers."

For information about configuring different types of security providers in a security realm, see Chapter 4, "Configuring WebLogic Security Providers."

You can also customize the default security configuration by creating an entirely new security realm, configuring configure security providers in that realm, and setting the new security realm as the default security realm. BEA recommends this process when upgrading a security configuration.

The remainder of this chapter explains describes the configuration decisions that need to be made when creating a new security realm and the main steps used to create a new security realm. Configuring a security realm is only one step in creating a new security configuration, you also need to configure security providers in that realm before in order for the security realm to be valid. For information about configuring different types of security providers in a security realm, see Chapter 4, "Configuring WebLogic Security Providers."

Configuration Decisions When Creating a New Security Realm

Before creating a new security realm, you need to make decisions about how the WebLogic Security service will use security information defined in deployment descriptors (DDs), the method for securing URLs and EJBs, how credential maps will be managed, and whether or not a deprecated WebLogic resource should be used..

When creating a new security realm, consider the following:

- Whether or not to set security roles and security policies for Web Applications and EJBs through deployment descriptors or through the WebLogic Server Administration Console.

  The Check Roles and Security Policies option determines how the WebLogic Security Service uses the security information defined in DDs. The option can be set as follows:

  – Web Applications and EJBs Protected in DD—specifies that the WebLogic Security Service only performs security checks on URL and EJB resources that have security specified in their associated deployment descriptors (DDs). This option is the default Check Roles and Policies setting.

  – All Web Applications and EJBs—specifies that the WebLogic Security Service performs security checks on all URL (Web) and EJB resources, regardless of whether there are any security settings in the deployment descriptors (DDs) for these WebLogic resources. If you change the setting of the Check Roles and Policies drop-down menu to All Web Applications and EJBs, specify Future Redeploys as specified in Step 2.

- How URL and EJB resources are to be secured. The Future Redeploy option determines how these WebLogic resources are to be secured. The option can be set as follows:

  – To secure URL and EJB resources using only the WebLogic Server Administration Console, select the `Ignore Roles and Policies From DD` (Deployment Descriptors) option.

  – To secure URL and EJB resources using only the deployment descriptors (that is, the `ejb-jar.xml`, `weblogic-ejb-jar.xml`, `web.xml`, and `weblogic.xml` files), select `Initialize roles and policies from DD` option.

- How to create and manage credential maps. You can load credential maps from `weblogic-ra.xml` deployment descriptor files into the embedded LDAP server and then use the WebLogic Server Administration Console to create new credential maps, or directly modify credential maps defined in the deployment descriptor.

  Once information from a `weblogic-ra.xml` deployment descriptor file is loaded into the embedded LDAP server, the original resource adapter remains unchanged. Therefore, if you redeploy the original resource adapter (which will happen if you redeploy it through the WebLogic Server Administration Console, modify it on disk, or restart WebLogic Server), the data will once again be imported from the `weblogic-ra.xml` deployment descriptor file and new credential mapping information may be lost.

- Whether or not to use the Web resource.

  The Web resource is deprecated in this release of WebLogic Server. If you are configuring a custom Authorization provider that uses the Web resource (instead of the URL resource) in the new security realm, enable Use Deprecated Web Resource on the new security

realm. This option changes the runtime behavior of the Servlet container to use a Web resource rather than a URL resource when performing authorization.

For more information, see "Create a new security realm" in the Administration Console online help.

Creating a New Security Realm:Main Steps

To create a new security realm:

1. Define a name and set configuration options for the security realm. See "Configuration Decisions When Creating a New Security Realm" on page 3-2.
2. Configure the required security providers for the security realm. A valid security realm requires an Authentication provider, an Authorization provider, an Adjudication provider, a Credential Mapping provider, and a Role Mapping provider. Otherwise, you will not be able to set the new security realm as the default security realm. See "Configuring WebLogic Security Providers" on page 4-1.
3. Optionally, define Identity Assertion, Auditing, and Cert Registry providers. See "Configuring WebLogic Security Providers" on page 4-1.
4. If you configured the WebLogic Authentication, Authorization, Credential Mapping or Role Mapping provider or the Certificate Registry in the new security realm, verify the default settings of the embedded LDAP server. See "Managing the Embedded LDAP Server" on page 6-1.
5. Optionally, improve the performance of the WebLogic or LDAP Authentication providers in the security realm. See "Improving the Performance of WebLogic and LDAP Authentication Providers" on page 4-13.
6. Protect WebLogic resources in the new security realm with security policies. Creating security policies is a multi-step process with many options. To fully understand this process, read *Securing WebLogic Resources*. This document should be used in conjunction with *Securing WebLogic Server* to ensure security is completely configured for a WebLogic Server deployment.
7. Protect user accounts in the new security realm.
8. Test the new security realm to ensure it is valid.
9. Set the new realm as the default security realm for the WebLogic domain.

For information:

- See “Create a new security realm” in the Administration Console online help.
- See “Test a new security realm” in the Administration Console online help.
- See “Set a new security realm as the default security realm” in the Administration Console online help.
- See “Delete a security realm” in the Administration Console online help.
- See “Revert to a previous security configuration” in the Administration Console online help.
- See “Configure security providers” in the Administration Console online help.
- You can also use the WebLogic Scripting tool or Java Management Extensions (JMX) APIs can to create a new security configuration. .

CHAPTER 4

Configuring WebLogic Security Providers

The following sections describe how to configure the security providers supplied by WebLogic Server and how to configure custom security providers.

- "Changing the Order of Authentication Providers" on page 4-22
- "Configuring a User Name Mapper" on page 4-22
- "Configuring a Custom User Name Mapper" on page 4-23
- "Configuring a WebLogic Authorization Provider" on page 4-23
- "Configuring a WebLogic Credential Mapping Provider" on page 4-24
- "Configuring a WebLogic Keystore Provider" on page 4-24
- "Configuring a WebLogic Role Mapping Provider" on page 4-25

Note: Only the Realm Adapter Auditing, Adjudication, and Authorization providers are available when you are using Compatibility Security.

When Do I Need to Configure a Security Provider?

By default, most configuration work for WebLogic security providers is done. However, the following circumstances require you to supply configuration information:

- Before using the WebLogic Identity Assertion provider, define the active token type. See "Configuring a WebLogic Identity Assertion Provider" on page 4-17.
- To map tokens to a user in a security realm, configure the user name mapper in the WebLogic Identity Assertion provider. See "Configuring a User Name Mapper" on page 4-22.
- To use auditing in the default (active) security realm, configure either the WebLogic Auditing provider or a custom Auditing provider. See "Configuring a WebLogic Auditing Provider" on page 4-4 or "Configure a custom security provider."
- To audit data for many types of ContextElements (such as HTTP servlet requests or EJB parameters). See "Configuring a WebLogic Auditing Provider" on page 4-4.
- To use identity assertion based on SAML assertions. See "Configuring a SAML Identity Assertion Provider" and "Configuring a SAML Credential Mapping Provider".
- To use HTTP and Kerberos-based authentication in conjunction with WebLogic Server. See "Configuring a Negotiate Identity Assertion Provider."
- To use certificate revocation. See "Configuring a Certificate Registry".

- To user, password, group, and group membership information stored in databases for authentication purposes. See “Configuring a DBMS Authentication Provider.” The DBMS Authentication providers can be used to upgrade from the RDBMS security realm.
- To use of Windows NT users and groups for authentication purposes. See “Configuring the Windows NT Authentication Provider.” The Windows NT Authentication provider is the upgrade path for the Window NT security realm.
- To use an LDAP server other than the embedded LDAP server, configure one of the LDAP Authentication providers. The LDAP authentication provider can be used instead of or in addition to the WebLogic Authentication provider. See “Configuring an LDAP Authentication Provider:Main Steps” on page 4-9.
- To use existing users and groups stored in an 6.x security realm (for example, the 6.x Windows NT, UNIX, RDBMS security realms or 6.x custom security realms) in *myrealm*, configure the Realm Adapter Authentication provider. The Realm Adapter Authentication provider can be used instead of or in addition to the WebLogic Authentication provider. See “Configuring a Realm Adapter Authentication Provider” on page 4-16.
- When creating a new security realm, configure security providers for that new realm. See “Creating a New Security Realm:Main Steps” on page 3-4.
- When adding a custom security provider to a security realm or replacing one of the WebLogic security providers with a custom security provider, configure options for the custom security provider. When writing a custom security provider, you can implement options that are configurable through the WebLogic Server Administration Console. However, those options are implementation-specific and are not addressed in this manual. See “Writing Console Extensions for Custom Security Providers” in *Developing Security Providers for WebLogic Server*.

You can use either theWebLogic-supplied security providers or a custom security provider in a security realm. For information about configuring a custom security provider, see “Configure a custom security provider.”

The remainder of this chapter contains conceptual information and configuration options for each security provider.

Configuring a WebLogic Adjudication Provider

When multiple Authorization providers are configured in a security realm, each may return a different answer to the "is access allowed" question for a given resource. This answer may be `PERMIT`, `DENY`, or `ABSTAIN`. Determining what to do if multiple Authorization providers do not agree on the answer is the primary function of the Adjudication provider. Adjudication providers

resolve authorization conflicts by weighting each Authorization provider's answer and returning a final decision.

Each security realm requires an Adjudication provider. You can use either a WebLogic Adjudication provider or a custom Adjudication provider in a security realm.

By default, most of the configuration options for the WebLogic Adjudication provider are defined. However, you can define the Require Unanimous Permit option to determine how the WebLogic Adjudication provider handles a combination of `PERMIT` and `ABSTAIN` votes from the configured Authorization providers.

- If the option is enabled (the default), all Authorization providers must vote `PERMIT` in order for the Adjudication provider to vote `true`.
- If the option is disabled, `ABSTAIN` votes are counted as `PERMIT` votes.

For more information:

- See "Configure the WebLogic Adjudication Provider" in the Administration Console online help.
- You can also use the WebLogic Scripting tool or Java Management Extensions (JMX) APIs can to create a new security configuration.

Configuring a WebLogic Auditing Provider

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 providers produce an electronic trail of computer activity. Configuring an Auditing provider is optional. The default security realm (*myrealm*) does not have an Auditing provider configured.

The WebLogic Auditing provider logs the events described in Table 4-1.

Table 4-1 WebLogic Auditing Provider Events

Audit Event	Indicates...
AUTHENTICATE	Simple authentication (username and password) occurred.
ASSERTIDENTITY	Perimeter authentication (based on tokens) occurred.

Table 4-1 WebLogic Auditing Provider Events

Audit Event	Indicates...
USERLOCKED	A user account is locked because of invalid login attempts.
USERUNLOCKED	The lock on a user account is cleared.
USERLOCKOUTEXPIRED	The lock on a user account expired.
AuditCreateConfigurationEvent	A user has created a resource in the domain. For this event to be logged, you nust have enabled the domain to produce configuration Audit Events. See "Configuration Auditing" in the *Administration Console Online Help*.
AuditSetAttributeConfigurationEvent	A user has modified the setting of a resource in the domain. For this event to be logged, you nust have enabled the domain to produce configuration Audit Events. See "Configuration Auditing" in the *Administration Console Online Help*.
AuditInvokeConfigurationEvent	A user has invoked an operation on a resource in the domain. For this event to be logged, you nust have enabled the domain to produce configuration Audit Events. See "Configuration Auditing" in the *Administration Console Online Help*.
AuditDeleteConfigurationEvent	A user has deleted a resource in the domain. For this event to be logged, you nust have enabled the domain to produce configuration Audit Events. See "Configuration Auditing" in the *Administration Console Online Help*.

By default, most configuration options for the WebLogic Auditing provider are defined. However, when configuring the WebLogic Auditing provider, you need to define the following on the Auditor-->Details page:

- Severity—Specifies the severity level appropriate for your WebLogic Server deployment. The WebLogic Auditing provider audits the specifies security event. Auditing can be initiated when the following levels of security events occur:
  - INFORMATION
  - WARNING
  - ERROR
  - SUCCESS
  - FAILURE
- Rotation Minutes—Specifies how many minutes to wait before creating a new `DefaultAuditRecorder.log` file. At the specified time, the audit file is closed and a new one is created. A backup file named `DefaultAuditRecorder.`*`YYYYMMDDHHMM`*`.log` (for example,`DefaultAuditRecorder.200405130110.log`) is created in the same directory.

All auditing information recorded by the WebLogic Auditing provider is saved in *`WL_HOME`*`\`*`yourdomain`*`\`*`yourserver`*`\DefaultAuditRecorder.log` by default. Although, an Auditing provider is configured per security realm, each server writes auditing data to its own log file in the server directory. You can specify a new directory location for the `DefaultAuditRecorder.log` file on the command line with the following Java startup option:

```
-Dweblogic.security.audit.auditLogDir=c:\foo
```

The new file location will be `c:\foo\`*`yourserver`*`\DefaultAuditRecorder.log`.

For more information, see "*Weblogic Server Command Reference*."

Warning: Using an Auditing provider affects the performance of WebLogic Server even if only a few events are logged.

For more information:

- See "Configure the WebLogic Auditing Provider" in the Administration Console online help.
- You can also use the WebLogic Scripting tool or Java Management Extensions (JMX) APIs can to create a new security configuration.

Choosing an Authentication Provider

Authentication is the process whereby the identity of users and 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.

The WebLogic Server security architecture supports: certificate-based authentication directly with WebLogic Server; HTTP certificate-based authentication proxied through an external Web server; perimeter-based authentication (Web server, firewall, VPN); and authentication based on multiple security token types and protocols.

Authentication is performed by an Authentication provider. WebLogic Server offers the following types of Authentication providers:

- *The WebLogic Authentication provider* accesses user and group information in the embedded LDAP server.
- *LDAP Authentication providers* access external LDAP stores. WebLogic Server provides LDAP Authentication providers which access Open LDAP, Netscape iPlanet, Microsoft Active Directory and Novell NDS stores.
- *The Realm Adapter Authentication provider* accesses user and group information stored in 6.x security realms.
- The *MedRec Authentication provider* implements authentication for the MedRec sample application. This Authentication provider can only be used with the MedRec sample application and should be removed from production domains. See the "Security Known Problems" section in the *WebLogic Server Release Notes*.
- *The WebLogic Identity Assertion provider* validates X.509 and IIOP-CSIv2 tokens and can use a user name mapper to map that token to a user in a WebLogic Server security realm.

In addition, you can use:

- Custom (non-WebLogic) Authentication providers, which offer different types of authentication technologies.
- Custom (non-WebLogic) Identity Assertion providers, which support different types of tokens.

Each security realm must have one at least one Authentication provider configured. The WebLogic Security Framework is designed to support multiple Authentication providers (and thus multiple LoginModules) for multipart authentication. Therefore, you can use multiple Authentication providers as well as multiple types of Authentication providers in a security realm. For example, if you want to use both a retina-scan and a username/password-based form of authentication to access a system, you configure two Authentication providers.

The way you configure multiple Authentication providers can affect the overall outcome of the authentication process. Use the JAAS Control Flag option to set up login dependencies between

Authentication providers and allow single-sign on between providers. See “Setting the JAAS Control Flag Option” on page 4-8.

Authentication providers are called in the order in which they are configured. Therefore, use caution when configuring Authentication providers. Use the Reorder the Configured Authentication Providers link to modify the configuration of the Authentication providers. See “Changing the Order of Authentication Providers” on page 4-22.

For more information, see:

- “Setting the JAAS Control Flag Option” on page 4-8
- “Configuring an LDAP Authentication Provider:Main Steps” on page 4-9
- “Configuring a WebLogic Authentication Provider” on page 4-13
- “Configuring a WebLogic Identity Assertion Provider” on page 4-17
- “How an LDAP X509 Identity Assertion Provider Works” on page 4-18

Setting the JAAS Control Flag Option

When you configure multiple Authentication providers, use the JAAS Control Flag option on the Authenticator-->General page to control how the Authentication providers are used in the login sequence.

The definitions for the JAAS Control Flag values are as follows:

- REQUIRED—The Authentication provider is always called, and the user must always pass its authentication test.
- SUFFICIENT—If the user passes the authentication test of the Authentication provider, no other Authentication providers are executed (except Authentication providers with the JAAS Control Flag set to REQUIRED) because the user was sufficiently authenticated.
- REQUISITE—If the user passes the authentication test of this Authentication provider, other providers are executed but can fail (except for Authentication providers with the JAAS Control Flag set to REQUIRED).
- OPTIONAL—The user is allowed to pass or fail the authentication test of this Authentication provider. However, if all Authentication providers configured in a security realm have the JAAS Control Flag set to OPTIONAL, the user must pass the authentication test of one of the configured providers.

When additional Authentication providers are added to an existing security realm, by default the Control Flag option is set to OPTIONAL. If necessary, change the setting of the Control Flag so that the Authentication provider works properly in the authentication sequence.

For more information:

- See “Set the JAAS control flag” in the Administration Console online help.
- You can also use the WebLogic Scripting tool or Java Management Extensions (JMX) APIs can to create a new security configuration..

Configuring an LDAP Authentication Provider:Main Steps

WebLogic Server does not support or certify any particular LDAP servers. Any LDAP v2 or v3 compliant LDAP server should work with WebLogic Server. The following LDAP directory servers have been tested:

- Netscape iPlanet version 4.1.3
- Active Directory shipped as part of Windows 2000
- Open LDAP version 2.0.7
- Novell NDS version 8.5.1

An LDAP Authentication provider can also be used to access other LDAP stores. However, you must choose a pre-defined LDAP provider and customize it. See “Accessing Other LDAP Servers” on page 4-11.

Configuring an LDAP Authentication provider is a multi-step process. The steps are as follows. (Detailed configuration procedures are in the Administration Console online help.)

1. Choose an LDAP Authentication provider. WebLogic Server provides the following LDAP Authentication providers:
   - iPlanet Authentication provider
   - Active Directory Authentication provider
   - Open LDAP Authentication provider
   - Novell Authentication provider.
2. Enable communication between the LDAP server and the LDAP Authentication provider. .

   For a more secure deployment, BEA recommends using the SSL protocol to protect communications between the LDAP server and WebLogic Server.

3. Configure options that control the cache for the LDAP server.
4. Configure options that control how searches of the LDAP directory work.
5. Specify where in the LDAP directory structure users are located.
6. Specify where in the LDAP directory structure groups are located.
7. Define how members of a group are located.

For more information:

- For configuration procedures, see "Configure an LDAP Authentication Provider" in the Administration Console online help.
- You can also use the WebLogic Scripting tool or Java Management Extensions (JMX) APIs can to create a new security configuration.
- "Requirements for Using an LDAP Authentication Provider" on page 4-10
- "Accessing Other LDAP Servers" on page 4-11
- "Configuring Failover for LDAP Authentication Providers" on page 4-11
- "Improving the Performance of WebLogic and LDAP Authentication Providers" on page 4-13

Requirements for Using an LDAP Authentication Provider

If an LDAP Authentication provider is the only configured Authentication provider for a security realm, you must have the `Admin` role to boot WebLogic Server and use a user or group in the LDAP directory. Do one of the following in the LDAP directory:

- By default in WebLogic Server, the `Admin` role includes the `Administrators` group. Create an `Administrators` group in the LDAP directory. Make sure the LDAP user who will boot WebLogic Server is included in the group.

  The Active Directory LDAP directory has a default group called `Administrators`. Add the user who will be booting WebLogic Server to the `Administrators` group and define Group Base Distinguished Name (DN) so that the `Administrators` group is found.

- If you do not want to create an `Administrators` group in the LDAP directory (for example, because the LDAP directory uses the `Administrators` group for a different purpose), create a new group (or use an existing group) in the LDAP directory and include the user from which you want to boot WebLogic Server in that group. In the WebLogic Server Administration Console, assign that group the `Admin` role.

Accessing Other LDAP Servers

The LDAP Authentication providers in this release of WebLogic Server work with the iPlanet, Active Directory, Open LDAP, and Novell NDS LDAP servers. You can use an LDAP Authentication provider to access new types of LDAP servers. Choose the existing LDAP provider that most closely matches the new LDAP server and customize the existing configuration for the new LDAP server.

See "Access other LDAP servers" in the Administration Console online help.

Configuring Failover for LDAP Authentication Providers

You can configure an LDAP provider to work with multiple LDAP servers and enable failover if one LDAP server is not available. Use the Host option on the Configuration-->LDAP page to specify the names of the additional LDAP servers. Each host name may include a trailing comma and a port number. In addition, set the Parallel Connect Delay and Connection Timeout options on the Configuration-->Details page:

- Parallel Connect Delay—Specifies the number of seconds to delay when making concurrent attempts to connect to multiple servers. An attempt is made to connect to the first server in the list. The next entry in the list is tried only if the attempt to connect to the current host fails. This setting might cause your application to block for an unacceptably long time if a host is down. If the value is greater than 0, another connection setup thread is started after the specified number of delay seconds has passed. If the value is 0, connection attempts are serialized.
- Connection Timeout—Specifies the maximum number of seconds to wait for the connection to the LDAP server to be established. If the set to 0, there is no maximum time limit and WebLogic Server waits until the TCP/IP layer times out to return a connection failure. Set to a value over 60 seconds depending upon the configuration of TCP/IP.

The following examples present scenarios that occur when LDAP options are set for LDAP failover.

Example 1

In the following scenario, WebLogic Server attempts to connect to `directory.knowledge.com`. After 10 seconds, the connect attempt times out and WebLogic Server attempts to connect to the next specified host (`people.catalog.com`). WebLogic Server then uses `people.catalog.com` as the LDAP Server for this connection.

LDAP Option	Value
Host	`directory.knowledge.com:1050 people.catalog.com 199.254.1.2` The LDAP servers are working as follows: `directory.knowledge.com:1050` is down `people.catalog.com` is up `199.254.1.2` is up
Parallel Connect Delay	0
Connect Timeout	10

Example 2

In the following scenario, WebLogic Server attempts to connect to `directory.knowledge.com`. After 1 second (specified by the Parallel Connect Delay option), the connect attempt times out and WebLogic Server tries to connect to the next specified host (`people.catalog.com`) and `directory.knowledge.com` at the same time. If the connection to `people.catalog.com` succeeds, WebLogic Server uses `people.catalog.com` as the LDAP Server for this connection.WebLogic Server cancels the connect to `directory.knowledge.com` after the connection to `people.catalog.com` succeeds.

LDAP Option	Value
Host	`directory.knowledge.com:1050 people.catalog.com 199.254.1.2` The LDAP servers are working as follows: `directory.knowledge.com:1050` is down `people.catalog.com` is up `199.254.1.2` is up
Parallel Connect Delay	1
Connect Timeout	10

See “Configure failover for LDAP Authentication providers” in the Administration Console online help.

Configuring a WebLogic Authentication Provider

The WebLogic Authentication provider allows you to edit, list, and manage users and group membership. User and group membership information for the WebLogic Authentication provider is stored in the embedded LDAP server. The WebLogic Authentication provider is case insensitive. Ensure user names are unique. By default, the WebLogic Authentication provider is configured. You should only have to configure a WebLogic Authentication provider when creating a new security realm.

By default, most of the configuration options for the WebLogic Authentication provider are defined. Use the Minimum Password Length option on the General page to specify the length of passwords defined for users that are stored in the embedded LDAP server. If you are using multiple Authentication providers, set the JAAS Control Flag option to determine how the WebLogic Authentication provider is used in the authentication process.

For more information:

- See “Configure a WebLogic Authentication provider” in the Administration Console online help.
- You can also use the WebLogic Scripting tool or Java Management Extensions (JMX) APIs can to create a new security configuration.
- “Setting the JAAS Control Flag Option” on page 4-8

Improving the Performance of WebLogic and LDAP Authentication Providers

You can improve the performance of the WebLogic and LDAP Authentication providers in the following ways:

- Configure the Active Directory Authentication provider to perform group membership lookups using the `tokenGroups` option. The `tokenGroups` option holds the entire flattened group membership for a user as an array of system ID (SID) values. The SID values are specially indexed in the Active Directory and yield extremely fast lookup response.
- Optimize the configuration the group membership caches used by the WebLogic and LDAP Authentication providers.

- Expose the internal PrincipalValidator cache and increase its thresholds.

The following sections describe how to implement these optimizations.

Configuring the Active Directory Authentication Provider to Improve Performance

To configure an Active Directory Authentication provider to use the `tokenGroups` option, set the following fields on the Active Directory--Details page:

- Use Token Groups for Group Membership Lookup—Indicates whether to use the Active Directory `tokenGroups` lookup algorithm instead of the standard recursive group membership lookup algorithm. By default, this option is not enabled.

  Note: Access to the `tokenGroups` option is required (meaning, the user accessing the LDAP directory must have privileges to read the `tokenGroups` option and the `tokenGroups` option must be in the schema for user objects).

- Enable SID to Group Lookup Caching—Indicates whether or not SID-to-group name lookup results are cached. This setting only applies if the Use Token Groups for Group Membership Lookup option is enabled.

- Max SID To Group Lookups In Cache—The maximum size of the Least Recently Used (LRU) cache for holding SID to group lookups. This setting applies only if both the Use Token Groups for Group Membership Lookup and Enable SID to Group Lookup Caching options are enabled.

See “Improve performance of the Active Directory Authentication provider” in the Administration Console online help.

Optimizing the Group Membership Caches

To optimize the group membership caches for WebLogic and LDAP Authentication providers, set the following options on the Authentication Provider-->Details page:

- Group Membership Searching—Controls whether group searches are limited or unlimited in depth. This option controls how deeply a search should recursive into nested groups. For configurations that use only the first level of nested group hierarchy, this option allows improved performance during user searches by limiting the search to the first level of the group.
  - If a limited search is defined, Max Group Membership Search Level must be defined.
  - If an unlimited search is defined, Max Group Membership Search Level is ignored.

- Max Group Membership Search Level—Controls the depth of a group membership search if Group Membership Searching is defined. Possible values are:
  - 0—Indicates only direct groups will be found. That is, when searching for membership in Group A, only direct members of Group A will be found. If Group B is a member of Group A, the members will not be found by this search.
  - Any positive number—Indicates the number of levels to search. For example, if this option is set to 1, a search for membership in Group A will return direct members of Group A. If Group B is a member of Group A, the members of Group B will also be found by this search. However, if Group C is a member of Group B, the members of Group C will not be found by this search.
- Enable Group Membership Lookup Hierarchy Caching—Indicates whether group membership hierarchies found during recursive membership lookup are cached. Each subtree found will be cached. The cache holds the groups to which a group is a member. This setting only applies if Group Membership is enabled. By default, it is disabled.
- Max Group Hierarchies in Cache—The maximum size of the Least Recently Used (LRU) cache that holds group membership hierarchies. This setting only applies if Enable Group Membership Lookup Hierarchy Caching is enabled.
- Group Hierarchy Cache TTL—The number of seconds cached entries stay in the cache. The default is 60 seconds.

For configuration porcedures, see "Optimize Group Membership Caches" in the Administration Console online help.

Optimizing the Principal Validator Cache

To improve the performance of a WebLogic or LDAP Authentication provider, the settings of the cache used by the WebLogic Principal Validation provider can be increased as appropriate. The Principal Validator cache used by the WebLogic Principal Validation provider caches signed WLSAbstractPrincipals. To optimize the performance of the Principal Validator cache, set options on the Authentication Provider-->General page:

- Enable WebLogic Principal Validator Cache—Indicates whether the WebLogic Principal Validation provider uses a cache. This setting only applies if Authentication providers in the security realm use the WebLogic Principal Validation provider and WLSAbstractPrincipals. By default, it is enabled.
- Max WebLogic Principals In Cache—The maximum size of the Last Recently Used (LRU) cache used for validated WLSAbstractPrincipals. The default setting is 500. This setting only applies if Enable WebLogic Principal Validator Cache is enabled.

For configuration procedures, see “Optimize Principal Validator Cache” in the Administration Console online help.

Configuring a Realm Adapter Authentication Provider

The Realm Adapter Authentication provider allows you to use users and groups from 6.x security realms with the security realms in this release of WebLogic Server. Use the Realm Adapter Authentication provider if you store users and groups in the 6.x Windows NT, UNIX, RDBMS security realms or 6.x custom security realm. A Realm Adapter Authentication provider can be configured instead of or in addition to the WebLogic Authentication provider.

When using Compatibility Security, a Realm Adapter Authentication provider is by default configured for the *CompatibilityRealm*. However, you can configure a Realm Adapter Authentication provider in any security realm. For information about using the Realm Adapter Authentication provider in the *CompatibilityRealm*, see “The Default Security Configuration in the CompatibilityRealm” on page 10-2.

The Realm Adapter Authentication provider also allows use of implementations of the `weblogic.security.acl.CertAuthenticator` class with this release of WebLogic Server. The Realm Adapter Authentication provider includes an Identity Assertion provider that asserts identity based on X.509 tokens. For information about using a CertAuthenticator with WebLogic Server, “Configuring the Identity Assertion Provider in the Realm Adapter Authentication Provider” on page 10-3.

When you add a Realm Adapter Authentication provider to a security realm with an Authentication provider already configured, the WebLogic Server Administration Console sets the JAAS Control Flag option on the Realm Adapter Authentication provider to OPTIONAL and checks for the presence of a `fileRealm.properties` file in the domain directory. The WebLogic Server Administration Console will not add the Realm Adapter Authentication provider to the security realm if the `fileRealm.properties` file does not exist.

Note: The subjects produced by the Realm Adapter Authentication provider do not contain principals for the groups to which a user belongs. Use the `weblogic.security.SubjectUtils.isUserInGroup()` method to determine whether a user is in a group. When you use subjects produced by the Realm Adapter Authentication provider there is no way to iterate the complete set of groups to which a user belongs.

For more information:

- For configuration procedures, see “Configure the Realm Adapter Authentication Provider” in the Administration Console online help.

- You can also use the WebLogic Scripting tool or Java Management Extensions (JMX) APIs can to create a new security configuration.

Configuring a WebLogic Identity Assertion Provider

If you are using perimeter authentication, you need to use an Identity Assertion provider. In perimeter authentication, a system outside of WebLogic Server establishes trust through tokens (as opposed to simple authentication, where WebLogic Server establishes trust through usernames and passwords). An Identity Assertion provider verifies the tokens and performs whatever actions are necessary to establish validity and trust in the token. Each Identity Assertion provider is designed to support one or more token formats.

Multiple Identity Assertion providers can be configured 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. In the Active Type field on the General page, define the active token type. The WebLogic Identity Assertion provider supports identity assertion with X509 certificates and CORBA Common Secure Interoperability version 2 (CSI v2). If you are using CSI v2 identity assertion, define the list of client principals in the Trusted Principals field.

If multiple Identity Assertion providers are configured in a security realm, they can all support the same token type. However, the token can be active for only one only provider at a time.

When using the WebLogic Identity Assertion provider in a security realm, you can use a user name mapper to map the tokens authenticated by the Identity Assertion provider to a user in the security realm. For more information about configuring a user name mapper, see "Configuring a User Name Mapper" on page 4-22.

If the authentication type in a Web application is set to `CLIENT-CERT`, the Web Application container in WebLogic Server performs identity assertion on values from request headers and cookies. If the header name or cookie name matches the active token type for the configured Identity Assertion provider, the value is passed to the provider.

The Base64 Decoding Required value on the Details page determines whether the request header value or cookie value must be Base64 Decoded before sending it to the Identity Assertion provider. The setting is enabled by default for purposes of backward compatibility; however, most Identity Assertion providers will disable this option.

For more information:

- See "Configure the WebLogic Identity Assertion provider" in the Administration Console online help.

- You can also use the WebLogic Scripting tool or Java Management Extensions (JMX) APIs can to create a new security configuration.

How an LDAP X509 Identity Assertion Provider Works

The LDAP X509 Identity Assertion provider receives an X509 certificate, looks up the LDAP object for the user associated with that certificate, ensures that the certificate in the LDAP object matches the presented certificate, and then retrieves the name of the user from the LDAP object.

[needs figure]

The LDAP X509 Identity Assertion provider works in the following manner:

1. An application is set up to use perimeter authentication (in other words, users or system process use tokens to assert their identity).
2. As part of the SSL handshake, the application presents it certificate. The Subject DN in the certificate can be used to locate the object that represents the user in the LDAP server. The object contains the user's certificate and name.
3. The LDAP X509 Identity Assertion provider uses the certificate in the Subject DN to construct an LDAP search to find the LDAP object for the user in the LDAP server. It gets the certificate from that object, ensures it matches the certificate it holds, and retrieves the name of the user.
4. The username is passed to the authentication providers configured in the security realm. The authentication providers ensure the user exists and locates the groups to which the user belongs.

Configuring an LDAP X509 Identity Assertion Provider:Main Steps

Typically, if you use the LDAP X509 Identity Assertion provider, you also need to configure an LDAP Authentication provider that uses an LDAP server. The authentication provider ensures the user exists and locates the groups to which the user belongs. You should ensure both providers are properly configured to communicate with the same LDAP server.

To use an LDAP X509 Identity Assertion provider:

1. Obtain certificates for users and putting them in an LDAP Server. See Chapter 7, "Configuring Identity and Trust."

There must be a correlation between the Subject DN in the certificate and the location of the object for that user in the LDAP server. The LDAP object for the user must also include configuration information for the certificate and the username that will be used in the Subject.

2. In the WebLogic Server Administration Console, configure the LDAP X509 Identity Assertion provider to find the LDAP object for the user in the LDAP directory given the certificate's Subject DN.
3. Configure the LDAP X509 Identity Assertion provider to search the LDAP server to locate the LDAP object for the user. This requires the following pieces of data.
   - A base LDAP DN from which to start searching. The Certificate Mapping option for the LDAP X509 Identity Assertion provider tells the identity assertion provider how to construct the base LDAP DN from the certificate's Subject DN. The LDAP object must contain an attribute that holds the certificate.
   - A search filter that only returns LDAP objects that match a defined set of options. The filter narrows the LDAP search. Configure User Filter Search to construct a search filter from the certificate's Subject DN.
   - Where in the LDAP directory to search for the base LDAP DN. The LDAP X509 Identity Assertion provider searches recursively (one level down). This value must match the values in the certificate's Subject DN.
4. Configure the Certificate field of the LDAP X509 Identity Assertion provider to specify how the LDAP object for the user holds the certificate. The LDAP object must contain an attribute the holds the certificate.
5. Configure the Username field of the LDAP X509 Identity Assertion provider to specify which of the LDAP object's attributes holds the username that should appear in the Subject DN.
6. Configure the LDAP server connection for the LDAP X509 Identity Assertion provider. The LDAP server information should be the same as the information defined for the LDAP Authentication provider configured in this security realm.
7. Configure an LDAP Authentication provider for use with the LDAP X509 Identity Assertion provider. The LDAP server information should be the same the information defined for the LDAP X509 Identity Assertion provider configured in Step 6.

For more information:

- See "Configure an LDAP X509 Identity Assertion provider" in the Administration Console online help.

- You can also use the WebLogic Scripting tool or Java Management Extensions (JMX) APIs can to create a new security configuration.

Ordering of Identity Assertion for Servlets

When an HTTP request is sent, there may be multiple matches that can be used for identity assertion. However, identity assertion providers can only consume one of the active token types at a time. As a result there is no way to provide a set of tokens that can be consumed with one call. Therefore, the servlet contained in WebLogic Server is forced to choose between multiple tokens to perform identity assertion. The following ordering is used:

1. An X.590 digital certificate (signifies two-way SSL to client or proxy plug-in with two-way SSL between the client and the Web server) if X.509 is one of the active token types configured for the Identity Assertion provider in the default security realm.
2. Headers with a name in the form `WL-Proxy-Client-<TOKEN>` where `<TOKEN>` is one of the active token types configured for the Identity Assertion provider in the default security realm.

   Note: This method is deprecated and should only be used for the purpose of backward compatibility.
3. Headers with a name in the form `<TOKEN>` where `<TOKEN>` is one of the active tokens types configured for the Identity Assertion provider in the default security realm.
4. Cookies with a name in the form `<TOKEN>` where `<TOKEN>` is one of the active tokens types configured for the Identity Assertion provider in the default security realm.

For example, if an Identity Assertion provider in the default security realm is configured to have the `FOO` and `BAR` tokens as active token types (for the following example, assume the HTTP request contains nothing relevant to identity assertion except active token types), identity assertion is performed as follows:

- If a request comes in with a `FOO` header over a two-way SSL connection, X.509 is used for identity assertion.
- If a request comes in with a `FOO` header and a `WL-Proxy-Client-BAR` header, the `BAR` token is used for identity assertion.
- If a request comes in with a `FOO` header and a `BAR` cookie, the `FOO` token will be used for identity assertion.

The ordering between multiple tokens at the same level is undefined, therefore:

- If a request comes in with a FOO header and a BAR header, then either the FOO or BAR token is used for identity assertion, however, which one is used is unspecified.
- If a request comes in with a FOO cookie and a BAR cookie, then either the FOO or BAR token is used for identity assertion, however, which one is used is unspecified.

Configuring Identity Assertion Performance in the Server Cache

When using an Identity Assertion provider (either for an X.509 certificate or some other type of token), Subjects (a grouping of related information for a single entity including an identity and its security-related configuration options) are cached within the server. This greatly enhances performance for servlets and EJB methods with `<run-as>` tags as well as for other places where identity assertion is used but not cached in the HTTPSession (for example, signing and encrypting XML documents).

Note: Caching can violate the desired semantics.

You can change the lifetime of items in this cache by setting the maximum number of seconds a Subject can live in the cache via the `-Dweblogic.security.identityAssertionTTL` command-line argument. The default for this command-line argument is 300 seconds (that is, 5 minutes). Possible values for the command-line argument are:

- Less than 0—Disables the cache.
- 0—Caching is enabled and the identities in the cache never timeout. Any changes in the user database of cached entities requires a server reboot in order for the server to pick them up.
- Greater than 0—Caching is enabled and the cache is reset at the specified number of seconds.

To improve the performance of identity assertion, specify a higher value for this command-line argument. Note that as identity assertion performance improves, the Identity Assertion provider is less responsive to changes in the configured Authentication provider. For example, a change in the user's group will not be reflected until the Subject is flushed from the cache and recreated. Setting a lower value for the command-line argument makes authentication changes more responsive at a cost for performance.

Changing the Order of Authentication Providers

The way you configure multiple Authentication providers can affect the overall outcome of the authentication process, which is especially important for multipart authentication. Authentication providers are called in the order in which they are configured. The Authentication Providers table lists the authentication providers in the order they were configured. The Re-order the Configured Authentication Provider link on the Authentication Providers table is used to change the order of Authentication providers.

For more information:

- Be aware that the way each Authentication provider's Control Flag is set affects the outcome of the authentication process. For more information, see "Setting the JAAS Control Flag Option" on page 4-8.
- See "Configure the WebLogic Identity Assertion provider" in the Administration Console online help.
- You can also use the WebLogic Scripting tool or Java Management Extensions (JMX) APIs can to create a new security configuration.

Configuring a User Name Mapper

WebLogic Server verifies the digital certificate of the Web browser or Java client when establishing a two-way SSL connection. However, the digital certificate does not identify the Web browser or Java client as a user in the WebLogic Server security realm. If the Web browser or Java client requests a WebLogic Server resource protected by a security policy, WebLogic Server requires the Web browser or Java client to have an identity. The WebLogic Identity Assertion provider allows you to enable a user name mapper that maps the digital certificate of a Web browser or Java client to a user in a WebLogic Server security realm.

The user name mapper must be an implementation of the `weblogic.security.providers.authentication.UserNameMapper` interface. This interface maps a token to a WebLogic Server user name according to whatever scheme is appropriate for your needs. By default, WebLogic Server provides a default implementation of the `weblogic.security.providers.authentication.UserNameMapper` interface. You can also write your own implementation.

The WebLogic Identity Assertion provider calls the user name mapper for the following types of identity assertion token types:

- X.509 digital certificates passed via the SSL handshake

- X.509 digital certificates passed via CSIv2
- X.501 distinguished names passed via CSIv2

The default user name mapper uses the subject DN of the digital certificate or the distinguished name to map to the appropriate user in the WebLogic Server security realm. For example, the user name mapper can be configured to map a user from the Email attribute of the subject DN (`smith@bea.com`) to a user in the WebLogic Server security realm (`smith`). Use Default User Name Mapper Attribute Type and Default Username Mapper Attribute Delimiter on the Details page to define this information:

- Default User Name Mapper Attribute Type—The subject distinguished name (DN) in a digital certificate used to create a username. Valid values are: `C`, `CN`, `E`, `L`, `O`, and `OU`.
- Default User Name Mapper Attribute Delimiter—Ends the username. The user name mapper uses everything to the left of the value to create a username.

For more information:

- See "Configure a user name mapper" in the Administration Console online help.
- You can also use the WebLogic Scripting tool or Java Management Extensions (JMX) APIs can to create a new security configuration.

Configuring a Custom User Name Mapper

You can also write a custom user name mapper to map a token to a WebLogic Server user name according to whatever scheme is appropriate for your needs. The custom user name mapper must be an implementation of the `weblogic.security.providers.authentication.UserNameMapper` interface. You then configure the custom user name mapper in the active security realm.

For more information:

- See "Configure a user name mapper" in the Administration Console online help.
- You can also use the WebLogic Scripting tool or Java Management Extensions (JMX) APIs can to create a new security configuration.

Configuring a WebLogic Authorization Provider

Authorization is the process whereby the interactions between users and resources are limited to ensure integrity, confidentiality, and availability. In other words, authorization is responsible for controlling access to resources based on user identity or other information. By default, the

WebLogic Authorization provider is configured. You should only have to configure a WebLogic Authorization provider when creating a new security realm.

See "Configure a WebLogic Authorization provider."

Configuring a WebLogic Credential Mapping Provider

Credential mapping is the process whereby the authentication and authorization mechanisms of a remote system (for example, a legacy system or application) are used to obtain an appropriate set of credentials to authenticate users to a target WebLogic resource.

For information about creating credential maps, see *Programming WebLogic J2EE Connector*.

By default, most configuration options for the WebLogic Credential Mapping provider are defined. However, you have the option of setting Credential Mapping Deployment Enabled which specifies whether or not this Credential Mapping provider imports credential maps from deployment descriptors (`weblogic-ra.xml` files) into the security realm. This setting is enabled by default.

In order to support Credential Mapping Deployment Enabled, a Credential Mapping provider must implement the `DeployableCredentialProvider` SSPI. The credential mapping information is stored in the embedded LDAP server.

For more information:

- See Implementing the DeployableCredentialMappingProvider SSPI in *Developing Security Services for WebLogic Server*
- See "Configure a WebLogic Credential Mapping provider" in the Administration Console online help.
- You can also use the WebLogic Scripting tool or Java Management Extensions (JMX) APIs can to create a new security configuration.

Configuring a WebLogic Keystore Provider

Note: The WebLogic Keystore provider is deprecated in this release of WebLogic Server. It is only supported for backward compatibility.

Configuring a WebLogic Role Mapping Provider

Role Mapping providers compute the set of roles granted to a subject for a given resource. Role Mapping providers supply Authorization providers with this role information so that the Authorization Provider can answer the "is access allowed?" question for WebLogic resources.

By default, most configuration options for the WebLogic Role Mapping provider are defined. However, you can set Role Mapping Deployment Enabled which specifies whether or not this Role Mapping provider imports information from deployment descriptors for Web applications and EJBs into the security realm. This setting is enabled by default.

In order to support Role Mapping Deployment Enabled, a Role Mapping provider must implement the `DeployableRoleProvider` SSPI. Roles are stored in the embedded LDAP server.

For more information:

- See Role Mapping Providers in *Developing Security Services for WebLogic Server.*
- See "Configure a WebLogic Role Mapping provider" in the Administration Console online help.
- You can also use the WebLogic Scripting tool or Java Management Extensions (JMX) APIs can to create a new security configuration.

Migrating Security Data

The following sections provide information about migrating security data between security realms and security providers.

Overview of Security Data Migration

Several WebLogic security providers support security data migration. This means you can export users and groups (for the WebLogic Authentication provider), security policies (for the WebLogic Authorization provider), security roles (for the WebLogic Role Mapping provider), or credential maps (for the WebLogic Credential Mapping provider) from one security realm, and import them into a new security realm. You can migrate security data for each security provider individually, or migrate security data for all the WebLogic security providers at once (that is, security data for an entire security realm). You migrate security data through the WebLogic Server Administration Console or by using the WebLogic Scripting Tool (WLST)

Migrating security data may be helpful when:

- Transitioning from development to production mode.

- Proliferating production mode security configurations to security realms in new WebLogic Server domains.
- Moving data from one security realm to a new security realm in the same WebLogic Server domain, where one or more of the WebLogic security providers will be replaced with custom security providers.

The remainder of this chapter describes the concepts you need to understand when migrating security data, the formats and constraints supported by the WebLogic security providers, and how to use WLST to migrate security data.

For more information about migrating security data from the WebLogic Server Administration Console, see the following topics in the online help:

- Export data from security realms
- Import data into security realms
- Export data from security providers
- Import data into security providers

Migration Concepts

A **format** is simply a data format that specifies how security data should be exported or imported. **Supported formats** are the list of data formats that a given security provider understands how to process.

Constraints are key/value pairs that specify options to the export or import process. Use constraints to control which security data is exported to or imported from the security provider's database (in the case of the WebLogic Server security providers, the embedded LDAP server). For example, you may want to export only users (not groups) from an Authentication provider's database. **Supported constraints** are the list of constraints you may specify during the migration process for a particular security provider. For example, an Authentication provider's database may be used to import users and groups, but not security policies.

Export files are the files to which security data is written (in the specified format) during the export portion of the migration process. **Import files** are files from which security data is read (also in the specified format) during the import portion of the migration process. Both export and import files are simply temporary storage locations for security data as it is migrated from one security provider's database to another.

Formats and Constraints Supported by the WebLogic Security Providers

WebLogic Server does not provide any standard, public formats for developers of security providers. Therefore, in order for security data to be exported and imported from one security provider to another, both security providers must understand how to process the same format.

Notes: Because the data format used for the WebLogic Server security providers is unpublished, you cannot currently migrate security data from a WebLogic security provider to a custom security provider, or visa versa.

WebLogic security providers support the following formats and constraints.

Table 5-1 Formats and Constraints Supported by the WebLogic Security Providers

WebLogic Provider	Supported Format	Supported Constraints
WebLogic Authentication Provider	DefaultAtn	Users, groups
WebLogic Authorization Provider	DefaultAtz	None
WebLogic Role Mapping Provider	DefaultRoles	None
WebLogic Credential Mapping Provider	DefaultCreds	Passwords

In the WebLogic Server Administration Console, the constraints are only displayed for the WebLogic Authentication provider because you have the option of exporting or importing users and groups, only users, or only groups.

When exporting credential maps from the WebLogic Credential Mapping provider, you need to specify whether or not the passwords for the credentials are exported in clear text. The mechanism used to encrypt passwords in each WebLogic Server domain is different, therefore, you want to export passwords in clear text if you plan to use them in a different WebLogic Server domain. After the credential maps are imported into the WebLogic Credential Mapping provider in the new WebLogic Server domain, the passwords are encrypted. Carefully protect the directory and file in which you export credential maps in clear text as secure data is available on your system during the migration process.

Migrating Data Using WLST

Migrating Data Using weblogic.admin

Note: The `weblogic.Admin` utility is deprecated in this release of WebLogic Server. Use WLST instead.

You can also use the `weblogic.Admin` utility to export and import security data between security realms and security providers. The format of the command is:

```
java weblogic.Admin -username username -password password \
INVOKE -mbean mbeanname \
-method methodname dataformat filename constraints
```

where

username—Name of the Admin user

password—Password of the Admin user

mbeanname—Name of the Security provider MBean.

methodname—`exportData` or `importData`

dataformat—`DefaultAtn`, `DefaultAtz`, `DefaultRoles`, `DefaultCreds`

filename—The directory location and filename in which to export or import the security data.

constraints—" "

Note: The directory and file into which you export the security data should be carefully protected with operating system security as they contain secure information about your deployment.

For example:

```
java weblogic.Admin -username system -password weblogic INVOKE -mbean
Security:Name=myrealmDefaultAuthenticator -method importData DefaultAtn
d:\temp\security.info " "
```

CHAPTER 6

Managing the Embedded LDAP Server

The embedded LDAP server is the default security provider database for the WebLogic Authentication, Authorization, Credential Mapping and Role Mapping providers.The following sections explain how to manage the embedded LDAP server.

Configuring the Embedded LDAP Server

The embedded LDAP server contains user, group, group membership, security role, security policy, and credential map information. By default, each WebLogic Server domain has an embedded LDAP server configured with the default values set for each type of information. The WebLogic Authentication, Authorization, Credential Mapping, and Role Mapping providers use the embedded LDAP server as their database. If you use any of these providers in a new security realm, you may want to change the default values for the embedded LDAP server to optimize its use in your environment.

Note: The performance of the embedded LDAP server works best with less than 50,000 users.

For more information:

- See “Configure the embedded LDAP server” in the Administration Console online help.
- See “Configure backups for the embedded LDAP server” in the Administation Console online help.

Embedded LDAP Server Replication

When you use the embedded LDAP Server in a WebLogic Server domain, updates are sent to a master LDAP server. The master LDAP server maintains a log of all changes. The master LDAP server also maintains a list replicated servers and the current change status for each one. The master LDAP server sends appropriate changes to each replicated server and updates the change status for each server. This process occurs when an update is made to the Master LDAP server. However, depending on the number of updates, it may take several seconds or more for the change to be replicated to the Managed Server. The master LDAP server is the embedded LDAP server on the Administration Server. The replicated servers are all the Managed Servers in the WebLogic Server domain.

Note: Deleting and modifying the configured security providers using the WebLogic Server Administration Console may require manual clean up of the embedded LDAP server. Use an external LDAP browser to delete unnecessary information.

Viewing the Contents of the Embedded LDAP Server from an LDAP Browser

To view the contents of the embedded LDAP server through an LDAP browser:

1. Down-load an external LDAP browser from the following location:

   `http:/www.iit.edu/~gawojar/ldap/`

2. To view the contents, change the embedded LDAP credentials.

   a. Expand the Domain node (for example, Examples).

   b. Select the Security-->Embedded LDAP page.

   c. Change Credential. See Table 6-1.

   d. Click Apply.

   e. Reboot WebLogic Server.

3. Enter the following command at a command prompt to start the LDAP browser:

```
lbe.sh
```

4. Configure a new connection in the LDAP browser:
   a. Set the host field to *localhost*.
   b. Set the port field to `7001` (`7002` if SSL is being used).
   c. Set the Base DN field to *dc=mydomain* where *mydomain* is the name of the WebLogic Server domain you are using.
   d. Uncheck the Anonymous Bind option.
   e. Set the User DN field to `cn=Admin`.
   f. Set the Password field to the password you specified in Step 1.
5. Click the new connection.

   Use the LDAP browser to navigate the hierarchy of the embedded LDAP server.

Exporting and Importing Information in the Embedded LDAP Server

In this release of WebLogic Server, the Migration page of each security provider can be used to import or export data from the embedded LDAP server. Optionally, an LDAP browser can be used to export and import data stored in the embedded LDAP Server. Table 6-1 summarizes where data is stored in the hierarchy of the embedded LDAP server.

Table 6-1 Location of Security Data in the Embedded LDAP Server

Security Data	Embedded LDAP Server DN
Users	ou=people,ou=*myrealm*,dc=*mydomain*
Groups	ou=groups,ou=*myrealm*,dc=*mydomain*
Security roles	ou=ERole,ou=*myrealm*,dc=*mydomain*
Security policies	ou=EResource,ou=*myrealm*,dc=*mydomain*

To export security data from the embedded LDAP server:

1. Enter the following command at a command prompt to start the LDAP browser:

```
lbe.sh
```

2. Specify the data to be exported (for example, to export users specify `ou=people,ou=`*`myrealm`*`,dc=`*`mydomain`*).
3. Select the LDIF-->Export option.
4. Select Export all children.
5. Specify the name of the file into which the data will be exported.

To import security data from the embedded LDAP server:

1. Enter the following command at a command prompt to start the LDAP browser:

   ```
   lbe.sh
   ```

2. Specify the data to be imported (for example, to import users specify `ou=people,ou=`*`myrealm`*`,dc=`*`mydomain`*).
3. Select the LDIF-->Import option.
4. Select Update/Add.
5. Specify the name of the file from which the data will be imported.

LDAP Access Control Syntax

The embedded LDAP server supports the IETF LDAP Access Control Model for LDAPv3. This section describes how those access control is implemented within the embedded LDAP server. These rules can be applied directly to entries within the directory as intended by the standard or can be configured and maintained by editing the access control file (`acls.prop`).

Note: The default behavior of the embedded LDAP server is to only allow access from the Admin account in WebLogic Server. The WebLogic security providers only use the Admin account to access the embedded LDAP server. If you are not planning to access the embedded LDAP server from an external LDAP browser or if you are planning only to use the Admin account, you do not need to edit the `acls.prop` file. You may ignore the procedures in this section.

The Access Control File

The access control file (`acls.prop`) maintained by the embedded LDAP server contains the complete list of access control lists (ACLs) for an entire LDAP directory. Each line in the access

control file contains a single access control rule. An access control rule is made up of the following components:

- Location in the LDAP directory where the rule applies
- Scope within that location to which the rule applies
- Access rights (either grant or deny)
- Permissions (either grant or deny)
- Attributes to which the rule applies
- Subject being granted or denied access

Listing 6-1 shows a sample access control file.

Listing 6-1 Sample acl.props File

```
[root]|entry#grant:r,b,t#[all]#public

ou=Employees,dc=octetstring,dc=com|subtree#grant:r,c#[all]#public:
ou=Employees,dc=octetstring,dc=com|subtree#grant:b,t#[entry]#public:
ou=Employees,dc=octetstring,dc=com|subtree#deny:r,c#userpassword#public:
ou=Employees,dc=octetstring,dc=com|subtree#grant:r#userpassword#this:
ou=Employees,dc=octetstring,dc=com|subtree#grant:w,o#userpassword,title,
description,
postaladdress,telephonenumber#this:
cn=schema|entry#grant:r#[all]#public:
```


Access Control Location

Each access control rule is applied to a given location in the LDAP directory. The location is normally a distinguished name (DN) but the special location `[root]` can be specified in the `acls.prop` file if the access control rule applies to the entire directory.

If an entry being accessed or modified on the LDAP server does not equal or reside below the location of the access control rule, the given access control rule is not evaluated further.

Access Control Scope

The following access control scopes are defined:

- Entry—An ACL with a scope of Entry is only evaluated if the entry in the LDAP directory shares the same DN as the location of the access control rule. Such rules are useful when a single entry contains more sensitive information than parallel or subentries entries.
- Subtree—A scope of Subtree is evaluated if the entry in the LDAP directory equals or ends with the location of this access control. This scope protects means the location entry and all subentries.

If an entry in the directory is covered by conflicting access control rules (for example, where one rule is an Entry rule and the other is a Subtree rule), the Entry rule takes precedence over rules that apply because of the Subtree rule.

Access Rights

Access rights apply to an entire object or to attributes of the object. Access can be granted or denied. Either of the actions `grant` or `deny` may be used when creating or updating the access control rule.

Each of the LDAP access rights are discrete. One right does not imply another right. The rights specify the type of LDAP operations that can be performed.

Attribute Permissions

The following permissions apply to actions involving attributes.

Table 6-2 Attribute Permissions

Permission	Description
`r` Read	Read attributes. If granted, permits attributes and values to be returned in a Read or Search operation.
`w` Write	Modify or add attributes. If granted, permits attributes and values to be added in a Modify operation.
`o` Obliterate	Modify and delete attributes. If granted, permits attributes and values to be deleted in a Modify operation.
`s` Search	Search entries with specified attributes. If granted, permits attributes and values to be included in a Search operation.

Table 6-2 Attribute Permissions

Permission	Description
c Compare	Compare attribute values. If granted, permits attributes and values to be included in a Compare operation.
m Make	Make attributes on a new LDAP entry below this entry.

The m permission is required for all attributes placed on an object when it is created. Just as the w and o permissions are used in the Modify operation, the m permission is used in the Add operation. The w and o permissions have no bearing on the Add operation and m has no bearing on the Modify operation. Since a new object does not yet exist, the a and m permissions needed to create it must be granted to the parent of the new object. This requirement differs from w and o permissions which must be granted on the object being modified. The m permission is distinct and separate from the w and o permissions so that there is no conflict between the permissions needed to add new children to an entry and the permissions needed to modify existing children of the same entry. In order to replace values with the Modify operation, a user must have both the w and o permissions.

Entry Permissions

The following permissions apply entire LDAP entries.

Table 6-3 Entry Permissions

Permission	Description
a Add	Add an entry below this LDAP entry. If granted, permits creation of an entry in the DIT subject to control on all attributes and values placed on the new entry at the time of creation. In order to add an entry, permission must also be granted to add at least the mandatory attributes.
d Delete	Delete this entry. If granted, permits the entry to be removed from the DIT regardless of controls on attributes within the entry.

Table 6-3 Entry Permissions

Permission	Description
`e` Export	Export entry and all sub entries to new location. If granted, permits an entry and its sub entries (if any) to be exported; that is, removed from the current location and placed in a new location subject to the granting of suitable permission at the destination. If the last RDN is changed, `Rename` permission is also required at the current location. In order to export an entry or its subentries, there are no prerequisite permissions to the contained attributes, including the RDN attribute. This is true even when the operation causes new attribute values to be added or removed as the result of the changes to the RDN.
`i` Import	Import entry and subordinates from specified location. If granted, permits an entry and its subentries (if any) to be imported; that is, removed from one other location and placed at the specified location (if suitable permissions for the new location are granted). When importing an entry or its subentries, there are no prerequisite permissions for the contained attributes, including the RDN attributes. This is true even when the operation causes new attribute values to be added or removed as the result of the changes of RDN.

Table 6-3 Entry Permissions

Permission	Description
`n` RenameDN	Change the DN of an LDAP entry. Granting the `Rename` permission is necessary for an entry to be renamed with a new RDN, taking into account consequential changes to the DN of subentries. If the name of the superior entry is unchanged, the grant is sufficient. When renaming an entry, there are no prerequisite permissions to contained attributes, including the RDN attributes. This is true even when the operation causes new attribute values to be added or removed as the result of the changes of RDN.
`b` BrowseDN	Browse the DN of an entry. If granted, permits entries to be accessed using directory operations that do not explicitly provide the name of the entry.
`t` ReturnDN	Allows DN of entry to be disclosed in an operation result. If granted, allows the distinguished name of the entry to be disclosed in the operation result.

Attributes Types

The attribute type(s) to which an access control rule applies should be listed where necessary. The following keywords are available:

- `[entry]` indicates the permissions apply to the entire object. This could mean actions such as delete the object, or add a child object.
- `[all]` indicates the permissions apply to all attributes of the entry.

If the keyword `[all]` and another attribute are both specified within an ACL, the more specific permission for the attribute overrides the less specific permission specified by the `[all]` keyword.

Subject Types

Access control rules can be associated with a number of subject types. The subject of an access control rule determines whether the access control rule applies to the currently connected session.

The following subject types are defined:

- `AuthzID`—Applies to a single user that can be specified as part of the subject definition. The identity of that user in the LDAP directory is typically defined as a DN.
- `Group`—Applies to a group of users specified by one of the following object classes:
  - `groupOfUniqueNames`
  - `groupOfNames`
  - `groupOfUniqueURLs`

  The first two types of groups contain lists of users, while the third type allows users to be included in the group automatically based on defined criteria.
- `Subtree`—Applies to the DN specified as part of the subject and all subentries in the LDAP directory tree.
- `IP Address`—Applies to a particular Internet address. This subject type is useful when all access must come through a proxy or other server. Applies only to a particular host, not to a range or subnet.
- `Public`—Applies to anyone connected to the directory, whether they are authenticated or not.
- `This`—Applies to the user whose DN matches that of the entry being accessed.

Grant/Deny Evaluation Rules

The decision whether to grant or deny a client access to an information in an entry is based on many factors related to the access control rules and the entry being protected. Throughout the decision making process, there are guiding principles.

- More specific rules override less specific ones (for example, individual user entries in an ACL take precedence over a group entry).
- If a conflict still exists in spite of the specificity of the rule, the subject of the rule determines which rule will be applied. Rules based on an `IP Address` subject are given the most precedence, followed by rules that are applied to a specific `AuthzID` or `This`

subject. Next in priority are rules that apply to `Group` subjects. Last priority is given to rules that apply to `Subtree` and `Public` subjects.

- When there are conflicting ACL values, Deny takes precedence over Grant.
- Deny is the default when there is no access control information. Additionally, an entry scope takes precedence over a subtree scope.

CHAPTER 7

Configuring Identity and Trust

This following sections describe how to configure identity and trust for WebLogic Server:

Before performing the steps in this chapter, review the "Identity and Trust" section in *Understanding WebLogic Security*.

Private Keys, Digital Certificates, and Trusted Certificate Authorities

Private keys, digital certificates, and trusted certificate authorities establish and verify server identity and trust.

SSL uses public key encryption technology for authentication. With public key encryption, a public key and a *private key* are generated for a server. The keys are related such that data encrypted with the public key can only be decrypted using the corresponding private key and vice versa. The private key is carefully protected so that only the owner can decrypt messages that were encrypted using the public key.

The public key is embedded into a *digital certificate* with additional information describing the owner of the public key, such as name, street address, and e-mail address. A private key and digital certificate provide *identity* for the server.

The data embedded in a digital certificate is verified by a certificate authority and digitally signed with the certificate authority's digital certificate. Well-know certificate authorities include Verisign and Entrust.net. The trusted certificate authority (CA) certificate establishes *trust* for a certificate.

An application participating in an SSL connection is authenticated when the other party evaluates and accepts the application's digital certificate. Web browsers, servers, and other SSL-enabled applications generally accept as genuine any digital certificate that is signed by a trusted certificate authority and is otherwise valid. For example, a digital certificate can be invalidated because it has expired or the digital certificate of the certificate authority used to sign it expired. A server certificate can be invalidated if the host name in the digital certificate of the server does not match the URL specified by the client.

Configuring Identity and Trust:Main Steps

To create identity and trust for a server:

1. Obtain digital certificates, private keys, and trusted CA certificates from the Cert Gen utility, Sun Microsystem's `keytool` utility, or a reputable vendor such as Entrust or Verisign. You can also use the digital certificates, private keys, and trusted CA certificates provided by the WebLogic Server kit. The demonstration digital certificates, private keys, and trusted CA certificates should be used in a development environment only.

2. Store the private keys, digital certificates, and trusted CA certificates. Private keys and trusted CA certificates are stored in a keystore.

   Note: This release of WebLogic Server supports private keys and trusted CA certificates stored in files, or in the WebLogic Keystore provider for the purpose of backward compatibility only.

3. Configure the Identity and Trust keystores for WebLogic Server in the WebLogic Server Administration Console.

The remainder of this chapter describes these steps.

Supported Formats for Identity and Trust

When using the deprecated file-based private keys, digital certificates, and trusted CAs, WebLogic Server can use digital certificates in either privacy-enhanced mail (PEM) or distinguished encoding rules (DER) format.

A `.pem` format file begins with this line:

```
----BEGIN CERTIFICATE----
```

and ends with this line:

```
----END CERTIFICATE----
```

A `.pem` format file supports multiple digital certificates (for example, a certificate chain can be included). The order is important (include the files in the order of trust). The server digital certificate should be the first digital certificate in the file. The issuer of that digital certificate should be the next file and so on until you get to the self-signed root certificate authority certificate.

A `.der` format file contains binary data. A `.der` file can be used only for a single certificate, while a `.pem` file can be used for multiple certificates.

Microsoft is often used as a certificate authority. Microsoft issues trusted CA certificates in p7b format. The trusted CA certificates must be converted to PEM before they can be used with WebLogic Server. For more information, see "Converting a Microsoft p7b Format to PEM Format" on page 7-8.

Private key files (meaning private keys not stored in a keystore) must be in PKCS#5/PKCS#8 PEM format.

You can still use private keys and digital certificates used with other versions of WebLogic Swith with this version of WebLogic Server. Convert the private key and digital certificate from privacy-enhanced mail (PEM) format to distinguished encoding rules (DER) format. For more information, see the description of the der2pem utility in *Using WebLogic Server Java Utilities*.

After converting the files, ensure the digital certificate file has the `-----BEGIN CERTIFICATE-----` header and the `-----END CERTIFICATE-----` footer. Otherwise, the digital certificate will not work.

Obtaining Private Keys, Digital Certificates, and Trusted Certificate Authorities

Servers need a private key, a digital certificate containing the matching public key, and a certificate for at least one trusted certificate authority. WebLogic Server supports private keys, digital certificates, and trusted CA certificates from the following sources:

- The demonstration digital certificates, private keys, and trusted CA certificates in the *WL_HOME*\server\lib directory.

  The demonstration digital certificates, private keys, and trusted CA certificates should be used in a development environment only.

- Sun Microsystem's `keytool` utility can also be used to generate a private key, a self-signed digital certificate for WebLogic Server, and a Certificate Signing Request (CSR).
  - Submit the CSR to a certificate authority to obtain a digital certificate for WebLogic Server.
  - Use the keytool utility to update the self-signed digital certificate with a new digital certificate.
  - Use the keytool utility to obtain trust and identity when using WebLogic Server in a production environment.

  For more information about Sun's `keytool` utility, see **keytool—Key and Certificate Management Tool**.

  Note: WebLogic Server does not support the use of the Digital Signature Algorithm (DSA). When using the keytool utility, the default key pair generation algorithm is DSA. Specify another key pair generation and signature algorithm when using WebLogic Server.

- The Cert Gen utility generates digital certificates and private keys that should be used only for demonstration or testing purposes and not in a production environment. Use the Cert Gen utility if you want to set an expiration date in the digital certificate or specify a correct host name in the digital certificate so that you can use host name verification. (The demonstration digital certificate provided by WebLogic Server uses the machine's default host name as the host name.) For more information about using the Cert Gen utility to obtain private keys and digital certificates, see "Using the Cert Gen Utility."

Note: The Certificate Request Generator servlet is deprecated in this release of WebLogic Server. Use the keytool utility from Sun Microsystems in place of the Certificate Request Generator servlet. For more information, see "Using the Common Keytool Commands."

Common Keytool Commands

Table 7-1 the `keytool` commands when creating and using JKS keystores with WebLogic Server.

Note: The `keytool` utility is a product of Sun Microsytems. Therefore, BEA Systems does not provide complete documentation on the utility. For more information, see keytool-Key and Certificate Management Tool.

Table 7-1 Commonly Used keytool Commands

Command	Description
`keytool -genkey -keystore` *`keystorename`* `-storepass` *`keystorepassword`*	Generates a new private key entry and self-signed digital certificate in a keystore. If the keystore does not exist, it is created.
`keytool -import -alias` *`aliasforprivatekey`* `-file` *`privatekeyfilename.pem`* `-keypass` *`privatekeypassword`* `-keystore` *`keystorename`* `-storepass` *`keystorepassword`*	Updates the self-signed digital certificate with one signed by a trusted CA.
`keytool -import -alias` *`aliasfortrustedca`* `-trustcacerts -file` *`trustedcafilename.pem`* `-keystore` *`keystorename`* `-storepass` *`keystorepassword`*	Loads a trusted CA certificate into a keystore. If the keystore does not exist, it is created.
`-certreq -alias alias -sigalg` *`sigalg`* `-file` *`certreq_file`* `-keypass` *`privatekeypassword`* `-storetype` *`keystoretype`* `-keystore` *`keystorename`* `-storepass` *`keystorepassword`*	Generates a CSR, using the PKCS#10 format. Sent the CSR to be sent to a trusted CA. The trusted CA authenticates the certificate requestor and returns a digital certificate to replace the existing self-signed digital certificate in the keystore.
`keytool -list -keystore` *`keystorename`*	Displays what is in the keystore.

Table 7-1 Commonly Used keytool Commands

Command	Description
`keytool -delete -keystore` *`keystorename`* `-storepass` *`keystorepassword`* `-alias` *`privatekeyalias`*	Delete a private key/digital certifcate pair for the specified alias from the keystore.
`keytool -help`	Provides online help for keytool.

Using the Cert Gen Utility

Note: The Cert Gen utility generates digital certificates and private keys that should only be used for demonstration or testing purposes and not in a production environment.

The CertGen utility creates a private key and digital certificate signed by the demonstration certificate authority (`CertGenCAB`). The digital certificates generated by the Cert Gen utility have the host name of the machine on which they were generated as the common name. Therefore, if you use host name verification, you must generate a digital certificate for every machine on which you wish to use SSL.

The CertGen utility generates two `.pem` files and two `.der` files. View the `.der` files in a Web browser to view the details of the generated digital certificate. Use the `.pem` files when you boot WebLogic Server or use the digital certificates with a client.

To generate a certificate:

1. Copy the following files to the directory in which you run the CertGen utility:
   - `WL_HOME\server\lib\CertgenCA.der`—The digital certificate for a certificate authority trusted by WebLogic Server.
   - `WL_HOME\server\lib\CertGenCAKey.der`—The private key for a certificate authority trusted by WebLogic Server.
2. Enter the following command at a command prompt:

   prompt> `java utils.CertGen` ***`password certfile keyfile`*** `[export] [`*`hostname`*`] [`*`genCA`*`]`

   where

   - ***`password`*** is the password for the private key.

- *certfile* is the name of the digital certificate file. The file is put in the working directory.
- *keyfile* is the name of the generated private key file. The file is put in the working directory.
- *hostname* is the name of the machine for which you are obtaining a digital certificate. This option allows you to use host name verification.
- *genCA* create a trusted CA for demonstration purposes only.

3. Use the `ImportPrivateKey` utility load the digital certificate and private key into a keystore.

   By default, the CertGen tool generates domestic strength certificates. Specify the `[export]` option if you want the tool to generate export strength certificates. If you want to export domestic strength digital certificates that use a host name, specify `[domestic]`.

The CertGen tool uses the JDK version 1.3 `InetAddress.getLocalHost().getHostname()` method to get the hostname it puts in the Subject common name. The `getHostName()` method works differently on different platforms. It returns a fully qualified domain name (FQDN) on some platforms (for example, Solaris) and a short host name on other platforms (for example, Windows NT). If WebLogic Server is acting as a client (and by default hostname verification is enabled), you need to ensure the hostname specified in the URL matches the Subject common name in the certificate. Otherwise, your connection fails because the host names do not match.

On Solaris, when you type `hostname` on the command line the server looks at the `/etc/hosts` file and gets a short host name. When you invoke `java.net.InetAddress.getHostName()`, the host goes to the `/etc/nsswitch.conf` file and depending on how the host is configured returns a FQDN or a short host name.

If the host entry is configured as:

```
hosts:   dns nis [NOTFOUND=return]
```

The host performs a name service look up first and uses the information `/etc/hosts` file only if DNS is not available. In this case, a FQDN is returned.

If the host entry is configured as:

```
hosts:   files dns nis [NOTFOUND=return]
```

The host goes to the `/etc/hosts` file first and then goes to DNS. In this case, a short hostname is returned.

Using Your Own Certificate Authority

Many companies act as their own certificate authority. To use those trusted CA certificates with WebLogic Server:

1. Ensure the trusted CA certificates are in PEM format.
   - If the trusted CA certificate is in DER format, use the utils.der2pem utility to convert them.
   - If the trusted CA certificate was issued by Microsoft, see "Converting a Microsoft p7b Format to PEM Format" on page 7-8.
   - If the trusted CA certificate has a custom file type, use the steps in "Converting a Microsoft p7b Format to PEM Format" on page 7-8 to convert the trusted CA certificate to PEM format.
2. Create a Trust keystore. For more information, see "How WebLogic Server Locates Trust" on page 7-13.
3. Store the trusted CA certificate in the Trust keystore. For more information, see "How WebLogic Server Locates Trust" on page 7-13.
4. Configure WebLogic Server to use the Trust keystore. For more information, see "Configuring Keystores For Production" on page 7-13.

Converting a Microsoft p7b Format to PEM Format

Microsoft is often used as a certificate authority. The digital certificates issued by Microsoft are in a format (p7b) that cannot be used by WebLogic Server. To convert a digital certificate in p7b format to PEM format:

1. In Windows Explorer on Windows 2000, select the file (*filename*.p7b) you want to convert.

   A Certificates window appears.
2. In the left pane of the Certificates window, expand the file you want to convert.
3. Select the Certificates option.

   A list of certificates in the p7b file appear.
4. Select the certificate to convert to PEM format.

   The Certificate Export wizard appears.

5. Click Next.
6. Select the `Base64 Encoded Cert` option (exports PEM formats).
7. Click Next.
8. Enter a name for the converted digital certificate.
9. Click Finish.

Note: For p7b certificate files that contain certificate chains, you need to concatenate the issuer PEM digital certificates to the certificate file. The resulting certificate file can be used by WebLogic Server.

Obtaining a Digital Certificate for a Web Browser

Low-security browser certificates are easy to acquire and can be done from within the Web browser, usually by selecting the Security menu item in Options or Preferences. Go to the Personal Certificates item and ask to obtain a new digital certificate. You will be asked for some information about yourself.

The digital certificate you receive contains public information, including your name and public key, and additional information you would like authenticated by a third party, such as your email address. Later you will present the digital certificate when authentication is requested.

As part of the process of acquiring a digital certificate, the Web browser generates a public-private key pair. The private key should remain secret. It is stored on the local file system and should never leave the Web browser's machine, to ensure that the process of acquiring a digital certificate is itself safe. With some browsers, the private key can be encrypted using a password, which is not stored. When you encrypt your private key, you will be asked by the Web browser for your password at least once per session.

Note: Digital certificates obtained from Web browsers do not work with other types of Web browsers or on different versions of the same Web browser.

Using Certificate Chains (Deprecated)

Note: The use of file-based certificate chains is deprecated in this release of WebLogic Server. Now the whole certificate chain is imported into a keystore. The steps in this section are provided for the purpose of backward compatibility only.

To use certificate chains with WebLogic Server:

1. Ensure that all the digital certificates are in PEM format. If they are in DER format, you can convert them using the utils.der2pem utility. If you are using a digital certificate issued by Microsoft, see "Converting a Micosoft p7b Format to PEM Format." You can use the steps in the section to convert other types of digital certificates. Save the digital certificate in Base 64 format.
2. Open a text editor and include all the digital certificate files into a single file. The order is important. The server digital certificate should be the first digital certificate in the file. The issuer of that digital certificate should be the next file and so on until you get to the self-signed root certificate authority certificate. This digital certificate should be the last certificate in the file.

   You cannot have blank lines between digital certificates.
3. Specify the file in Server Certificate File Name in the SSL Configuration portion of the Keystores and SSL page in the WebLogic Server Administration Console.

Listing 7-1 shows a sample certificate chain.

Listing 7-1 Sample File with Certificate Chain

```
-----BEGIN CERTIFICATE-----
MIICyzCCAjSgAwIBAgIBLDANBgkqhkiG9w0BAQQFADCBtjELMAkGA1UEBhMCVVMxEzARBgNVBA
gTCkNhbGlmb3JuaWExFjAUBgNVBAcTDVNhbiBGcmFuY2lzY28xFTATBgNVBAoTDEJFQSBXZWJM
b2dpYzERMA8GA1UECxMIU2VjdXJpdHkxLzAtBgNVBAMTJkRlbW8gQ2VydGlmaWNhdGUgQXV0aG
9yaXR5IENvbnN0cmFpbnRzMR8wHQYJKoZIhvcNAQkBFhBzZWN1cml0eUBiZWEuY29tMB4XDTAy
MTEwMTIwMDIxMloXDTA2MTAxNTIwMDIxMlowgZ8xCzAJBgNVBAYTAlVTMRMwEQYDVQQIEwpDYW
xpZm9ybmlhMRYwFAYDVQQHEw1TYW4gRnJhbmNpc2NvMRUwEwYDVQQKEwxCRUEgV2ViTG9naWMx
ETAPBgNVBAsTCFNlY3VyaXR5MRkwFwYDVQQDExB3ZWJsb2dpYy5iZWEuY29tMR4wHAYJKoZIhv
cNAQkBFg9zdXBwb3J0QGJlYS5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMJX8nKU
gsFej8pEu/1IVcHUkwY0c2JbBzOryu3sce4QjX+rGxiCjoPm2MY=yts2BvonuJ6CztdZf8B/LB
EWCz+qRrtdFn9mKSZWGvrAkmMPz2RhXEOThpoRo5kZz2FQ9XF/PxIJXTYCM7yooRBwXoKYjquR
wiZNtUiU9kYi6Z3prAgMBAAEwDQYJKoZIhvcNAQEEBQADgYEAh2eqQGxEMUnNTwEUD
0tBq+7YuAkjecEocGXvi2G4YSoWVLgnVzJoJuds3c35KE6sxBe1luJQuQkE9SzALG/6lDIJ5ct
PsHFmZzZxY7scLl6hWj5ON8oN2YTh5Jo/ryqjvnZvqiNIWe/gqr2GLIkajC0mz4un1LiYORPig
3fBMH0=
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIC+jCCAmOgAwIBAgIBADANBgkqhkiG9w0BAQQFADCBtjELMAkGA1UEBhMCVVMxEzARBgNVBA
gTCkNhbGlmb3JuaWExFjAUBgNVBAcTDVNhbiBGcmFuY2lzY28xFTATBgNVBAoTDEJFQSBXZWJM
b2dpYzERMA8GA1UECxMIU2VjdXJpdHkxLzAtBgNVBAMTJkRlbW8gQ2VydGlmaWNhdGUgQXV0aG
9yaXR5IENvbnN0cmFpbnRzMR8wHQYJKoZIhvcNAQkBFhBzZWN1cml0eUBiZWEuY29tMB4XDTAy
MTEwMTIwMDIxMVoXDTA2MTAxNjIwMDIxMVowgbYxCzAJBgNVBAYTAlVTMRMwEQYDVQQIEwpDYW
xpZm9ybmlhMRYwFAYDVQQHEw1TYW4gRnJhbmNpc2NvMRUwEwYDVQQKEwxCRUEgV2ViTG9naWMx
```

```
ETAPBgNVBAsTCFNlY3VyaXR5MS8wLQYDVQQDEyZEZW1vIENlcnRpZmljYXRlIEF1dGhvcml0eS
BDb25zdHJhaW50czEfMB0GCSqGSIb3DQEJARYQc2VjdXJpdHlAYmVhLmNvbTCBnzANBgkqhkiG
9w0BAQEFAAOBjQAwgYkCgYEA3ynD8l5JfLob4g6d94dNtI0Eep6QNl9bblmswnrjIYz1BVjjRj
NVal9fRs+8jvm85kIWlerKzIMJgiNsj50WlXzNX6orszggSsW15pqV0aYE9Re9K
CNNnORlsLjmRhuVxg9rJFEtjHMjrSYr2IDFhcdwPgIt0meWEVnKNObSFYcCAwEAAaMWMBQwEgY
DVR0TAQH/BAgwBgEB/wIBATANBgkqhkiG9w0BAQQFAAOBgQBS+0oqWxGyqbZO028zf9tQT2RKo
jfuwywrDoGW96Un5IqpFnBHIu5atliJo3OUpiH18KkwLN8DVP/3t3K3O3kXdIuLbqAL0i5xyBl
Ahr7gE5eVhIyeMg7ETBPLyGO2BF13Y24LlsO+MX9jW7fxMraPN608QeJXkZw0E0cGwrw2AQ==
-----END CERTIFICATE-----
```

Storing Private Keys, Digital Certificates, and Trusted Certificate Authorities

Once you have obtained private keys, digital certificates, and trusted CA certificates, you need to store them so that WebLogic Server can use them to find and verify identity. Private keys, their associated digital certificates, and trusted CA certificates are stored in keystores. The keystores can be configured through the WebLogic Server Administration Console **or specified on the command-line.** Use the Keystore Configuration section of the Keystores and SSL page of the WebLogic Server Administration Console to configure Identity and Trust keystores for WebLogic Server.

For the purpose of backward compatibility, private keys and trusted CA certificates can be stored in a file or in a JKS keystore accessed via the WebLogic Keystore provider. In addition, trusted CA certificates can be stored in a JKS keystore. Use the SSL Configuration section of the Keystores and SSL page of the WebLogic Server Administration Console to specify identity and trust options when using a file or a JKS keystore accessed via the WebLogic Keystore provider.

Guidelines for Using Keystores

When configuring SSL you have to decide how identity and trust will be stored. Although one keystore can be used for both identity and trust, for the following reasons, BEA recommends using separate keystores for both identity and trust:

The Identity keystore (private key/digital certificate pairs) and the Trust keystore (trusted CA certificates) may have different security requirements. For example:

- The Identity keystore may be prohibited by company policy from ever being put in the network while the Trust keystore can be distributed over the network.

- The Identity keystore may be protected by the operating system for both reading and writing by non-authorized users while the Trust keystore only needs to be write protected.
- The password for the trust keystore is generally known by more people than the password for the Identity keystore.

For identity, you only have to put the certificates (non-sensitive data) in the keystore while for trust, you have to put the certificate and private key (sensitive data) in the keystore.

Machines tend to have the same trust rules across an entire domain (meaning, they use the same set of trusted CAs), while they tend to have per server identity. Identity requires a private key and private keys should not be copied from one machine to another. Therefore, separate keystores per machine are created each containing only the server identity needed for that machine. However, trust keystores can be copied from machine to machine thus making it easier to standardizes trust rules.

Identity is more likely to be store in hardware keystores such as nCipher. Trust can be stored in a file-based JDK keystore without having security issues since trust only has certificates not private keys.

Creating a Keystore and Loading Private Keys and Trusted Certificate Authorities into the Keystore

A keystore is a mechanism designed to create and manage private keys/digital certificate pairs and trusted CA certificates. Use the following mechanisms to create a keystore and load private keys and trusted CA certificates into the keystore:

- The WebLogic `ImportPrivateKey` utility. The `ImportPrivateKey` utility allows you to take private key and digital certificate files and load them into a keystore. For more information, see **utils.ImportPrivateKey** in the *WebLogic Server Administration Guide*.
- Sun Microsystem's `keytool` utility. Use the `keytool` utility to generate a private key/digital certificate pair and then import the signed private key into the keystore. For more information, see "How WebLogic Server Locates Trust" on page 7-13. While you can use the `keytool` utility to generate new private keys and digital certificates and add them to a keystore, the utility does not allow you to take an existing private key from a file and import it into the keystore. Instead, use the WebLogic `ImportPrivateKey` utility.

  Note: The `keytool` utility does allow you to import trusted CA certificates in a file into a keystore.
- Custom utilities. This release of WebLogic Server can use keystores created with custom tools or utilites. How to create and use these utilities is outside the scope of this document.

All private key entries in a keystore are accessed by WebLogic Server via unique aliases. You specify the alias when loading the private key into the keystore. Aliases are case-insensitive; the aliases `Hugo` and `hugo` would refer to the same keystore entry. Aliases for private keys are specified in Private Key Alias when configuring SSL.

All certificate authorities in a keystore identified as trusted by WebLogic Server are trusted. Although WebLogic Server does not use the alias to access trusted CA certificates, the keystore does require an alias when loading a trusted CA certificate into the keystore.

How WebLogic Server Locates Trust

WebLogic Server uses the following algorithm when it loads its trusted CA certificates:

1. If the keystore is specified by the `-Dweblogic.security.SSL.trustedCAkeystore` command-line argument, load the trusted CA certificates from that keystore.
2. Else if the keystore is specified in the configuration file (`config.xml`), load trusted CA certificates from the specified keystore. If the server is configured with DemoTrust, trusted CA certificates will be loaded from the `WL_HOME\server\lib\DemoTrust.jks` and the JDK `cacerts` keystores.
3. Else if the trusted CA file is specified in the configuration file (`config.xml`), load trusted CA certificates from that file (this is only for compatibility with 6.x SSL configurations).
4. Else load trusted CA certificates from `WL_HOME\server\lib\cacerts` keystore.

Configuring Keystores For Production

By default, WebLogic Server is configured with two keystores:

- `DemoIdentity.jks`—Contains a demonstration private key for WebLogic Server. This keystore contains the identity for WebLogic Server.
- `DemoTrust.jks`—Contains the trusted certificate authorities from the `WL_HOME\server\lib\DemoTrust.jks` and the JDK `cacerts` keystores. This keystore establishes trust for WebLogic Server.

These keystores are located in the *`WL_HOME`*`\server\lib` directory. For testing and development purposes, the keystore configuration is complete. However, the demonstration keystores should not be used in a production environment. All the digital certificates and trusted CA certificates in the keystores are signed by a WebLogic Server demonstration certificate authority. Therefore, all WebLogic Server installations trust each other. This will leave your SSL connections wide open to a number of security attack.

To configure keystores for use in a production environment:

1. Obtain private keys and digital certificates from a reputable certificate authority such as Verisign, Inc. or Entrust.net. For more information, see "Obtaining Private Keys, Digital Certificates, and Trusted Certificate Authorities" on page 7-4.
2. Create Identity and Trust keystores. For more information, see "Creating a Keystore and Loading Private Keys and Trusted Certificate Authorities into the Keystore" on page 7-12.
3. Load the private keys and trusted CAs into the Identity and Trust keystores. For more information, see "Creating a Keystore and Loading Private Keys and Trusted Certificate Authorities into the Keystore" on page 7-12.
4. Use the WebLogic Server Administration to configure the Identity and Trust keystores.

For more information:

- See "Configuring Keystores" in the Administration Console online help.
- You can also use the WebLogic Scripting tool or Java Management Extensions (JMX) APIs can to create a new security configuration.

CHAPTER 8

Configuring SSL

The following sections describe how to configure SSL for WebLogic Server:

Note: This chapter applies to WebLogic Server deployments using the security features in this release of WebLogic Server as well as deployments using Compatibility Security.

Configuring SSL is an optional step; however, BEA recommends using SSL in a production environment.

SSL: An Introduction

Secure Sockets Layer (SSL) provides secure connections by allowing two applications connecting over a network connection to authenticate the other's identity and by encrypting the data exchanged between the applications. Authentication allows a server and optionally a client to verify the identity of the application on the other end of a network connection. Encryption makes data transmitted over the network intelligible only to the intended recipient.

SSL in WebLogic Server is an implementation of the SSL 3.0 and Transport Layer Security (TLS) 1.0 specifications.

WebLogic Server supports SSL on a dedicated listen port which defaults to 7002. To establish an SSL connection, a Web browser connects to WebLogic Server by supplying the SSL listen port and the HTTPs schema in the connection URL, for example, `https://myserver:7002`.

Using SSL is computationally intensive and adds overhead to a connection. Avoid using SSL in development environments when it is not necessary. However, always use SSL in a production environment.

One-Way and Two-Way SSL

SSL can be configured one-way or two-way:

- With one-way SSL, 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. One-way SSL is common on the Internet where customers want to create secure connections before they share personal data. Often, clients will also use SSL to log on so that the server can authenticate them.
- With two-way SSL, the server presents a certificate to the client and the client presents a certificate to the server. WebLogic Server can be configured to require clients to submit valid and trusted certificates before completing the SSL connection.

Setting Up SSL:Main Steps

To set up SSL:

1. Obtain an identity (private key and digital certificates) and trust (certificates of trusted certificate authorities) for WebLogic Server. Use the digital certificates, private keys, and trusted CA certificates provided by the WebLogic Server kit, the Cert Gen utility, Sun

Microsystem's `keytool` utility, or a reputable vendor such as Entrust or Verisign to perform this step.

2. Store the identity and trust. Private keys and trusted CA certificates which specify identity and trust are stored in a keystore.

   Note: This release of WebLogic Server supports private keys and trusted CA certificates stored in files, or in the WebLogic Keystore provider for the purpose of backward compatibility only.

3. Configure the Identity and Trust keystores for WebLogic Server in the WebLogic Server Administration Console.

4. Set SSL configuration options for the private key alias and password in the WebLogic Server Administration Console. Optionally, set configuration options that require the presentation of client certificates (for two-way SSL).

For more information:

- See the following topics in the Administration Console online help:
  - "Configure SSL"
  - "Configure two-way SSL"
  - "Disable the SSL port" (
- For information on configuring identity and trust for WebLogic Server, see "Obtaining Private Keys, Digital Certificates, and Trusted Certificate Authorities" on page 7-4 and "Storing Private Keys, Digital Certificates, and Trusted Certificate Authorities" on page 7-11

Using Host Name Verification

A host name verifier ensures the host name in the URL to which the client connects matches the host name in the digital certificate that the server sends back as part of the SSL connection. A host name verifier is useful when an SSL client or an SSL server acting as a client connects to an application server on a remote host. It helps to prevent man-in-the-middle attacks.

By default, WebLogic Server has host name verification enabled. As a function of the SSL handshake, WebLogic Server compares the common name in the SubjectDN in 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. The SSL client is the actual party that drops the SSL connection if the names do not match.

If anything other than the default behavior is desired, either turn off host name verification or configure a custom host name verifier. Turning off host name verification leaves WebLogic Server vulnerable to man-in-the-middle attacks. BEA recommends leaving host name verification on in production environments.

In this release of WebLogic Server, the host name verification feature is updated so that if the host name in the certificate matches the local machine's host name, host name verification passes if the URL specifies *localhost*, `127.0.01` or the default IP address of the local machine.

For more information:

- See "Configuring Keystores" in the Administration Console online help.
- See "Verify Host Name Verification is enabled" in the Administration Console online help.
- See "Disabling Host Name Verification" on page 8-4
- See "Configuring a Custom Host Name Verifier" on page 8-4

Disabling Host Name Verification

Host name verification can be disabled on the WebLogic Server command-line or by using the WebLogic Server Administration Console.

Note: When using Java clients, host name verification must be set on the command-line.

On the command line of the SSL client, enter the following argument:

```
-Dweblogic.security.SSL.ignoreHostnameVerification=true
```

See the "Disable Host Name Verification" topic in the Administration Console online help.

Configuring a Custom Host Name Verifier

You can write a custom host name verifier. For more information, see *Programming WebLogic Security*. If you write a custom host name verifier, the name of the class that implements the host name verifier must be specified in the CLASSPATH of WebLogic Server (when acting as an SSL client) or a Java client acting as an SSL client.

Configuring a custom host name verifier for a server is done using the WebLogic Server Administration Console. See the "Configure a Custom Host Name Verifier" topic in the online help.

When using Java clients, a custom host name verifier must be specified on the command-line using the following argument:

```
-Dweblogic.security.SSL.HostnameVerifier=classname
```

where *classname* specifies the implementation of the `weblogic.security.SSL.HostnameVerifier` interface.

Enabling SSL Debugging

SSL debugging provides more detailed information about the SSL events that occurred during an SSL handshake. The SSL debug trace displays information about:

- Trusted certificate authorities
- SSL server configuration information
- Server identity (private key and digital certificate)
- The strength that is allowed by the license in use
- Enabled ciphers
- SSL records that were passed during the SSL handshake
- SSL failures detected by WebLogic Server (for example, trust and validity checks and the default host name verifier)
- I/O related information

Use the following command-line properties to enable SSL debugging:

```
-Dssl.debug=true -Dweblogic.StdoutDebugEnabled=true
```

The SSL debugging properties can be included in the start script of the SSL server, the SSL client, and the Node Manager. For a Managed Server started by the Node Manager, specify this command-line argument on the Remote Start page for the Managed Server.

SSL debugging dumps a stack trace whenever an ALERT is created in the SSL process. The types and severity of the ALERTS are defined by the TLS specification.

The stack trace dumps information into the log file where the ALERT originated. Therefore, when tracking an SSL problem, you made need to enable debugging on both sides of the SSL connection (on both the SSL client or the SSL server). The log file contains detailed information about where the failure occurred. To determine where the ALERT occurred, confirm whether there is a trace message after the ALERT. An ALERT received after the trace message indicates the failure occurred on the peer. To determine the problem, you need to enable SSL debugging on the peer in the SSL connection.

When tracking an SSL problem, review the information in the log file to ensure:

- The correct `config.xml` file was loaded
- The license (domestic or export) is correct
- The trusted certificate authority was valid and correct for this server.
- The host name check was successful
- The certificate validation was successful

Note: Sev 1 type 0 is a normal close ALERT, not a problem.

SSL Session Behavior

WebLogic Server allows SSL sessions to be cached. Those sessions live for the life of the server.

Clients that use SSL sockets directly can control the SSL session cache behavior. The SSL session cache is specific to each SSL context. All SSL sockets created by SSL socket factory instances returned by a particular SSL context can share the SSL sessions.

Clients default to resuming sessions at the same IP address and port. Multiple SSL sockets that use the same host and port share SSL sessions by default assuming the SSL sockets are using the same underlying SSL context.

Clients that don't want SSL sessions to be used at all need to explicitly call `setEnableSessionCreation(false)` on the SSL socket to ensure that no SSL sessions are cached. This setting only controls whether an SSL session is added to the cache, it does not stop an SSL socket from finding an SSL session that was already cached (for example, SSL socket 1 caches the session, SSL socket 2 sets `setEnableSessionCreation` to `false` but it can still reuse the SSL session but can still reuse the SSL session from SSL socket 1 since that session was put in the cache.)

SSL sessions exist for the lifetime of the SSL context; they are not controlled by the lifetime of the SSL socket. Therefore, creating a new SSL socket and connecting to the same host and port can resume a previous session as long as the SSL socket is created using an SSL socket factory from the SSL context that has the SSL session in its cache.

By default, clients that use HTTPS URLs get a new SSL session for each URL because each URL uses a different SSL context and therefore SSL sessions can not be shared or reused. The SSL session can be retrieved using the `weblogic.net.http.HttpsClient` class or the `weblogic.net.http.HttpsURLConnection` class. Clients can also resume URLs by sharing a SSLSocket Factory between them.

In WebLogic Server 6.x, one SSL session is cached in the thread of execution. In this release of WebLogic Server, session caching is maintained by the SSL context which can be shared by threads. A single thread has access to the entire session cache not just one SSL session so multiple SSL sessions can be used and shared in a single (or multiple) thread.

In addition, how SSL session behavior worked changed in this release of WebLogic Server. The `weblogic.security.SSL.sessionCache.size` and `weblogic.security.SSL.sessionCache.ttl` command-line arguments are now ignored.

Configuring RMI over IIOP with SSL

Use SSL to protect IIOP connections to RMI remote objects. SSL secures connections through authentication and encrypts the data exchanged between objects.

To use SSL to protect RMI over IIOP connections, do the following:

1. Configure WebLogic Server to use SSL.
2. Configure the client Object Request Broker (ORB) to use SSL. Refer to the product documentation for your client ORB for information about configuring SSL.
3. Use the `host2ior` utility to print the WebLogic Server IOR to the console. The `host2ior` utility prints two versions of the interoperable object reference (IOR), one for SSL connections and one for non-SSL connections. The header of the IOR specifies whether or not the IOR can be used for SSL connections.
4. Use the SSL IOR when obtaining the initial reference to the CosNaming service that accesses the WebLogic Server JNDI tree.

For more information about using RMI over IIOP, see *Programming WebLogic RMI* and *Programming WebLogic RMI over IIOP*.

SSL Certificate Validation

In previous releases, WebLogic Server did not ensure each certificate in a certificate chain was issued by a certificate authority. This problem meant anyone could get a personal certificate from a trusted certificate authority, use that certificate to issue other certificates, and WebLogic Server would not detect the invalid certificates. Now all X509 V3 CA certificates used with WebLogic Server must have the Basic Constraint extension defined as CA, thus ensuring all certificates in a certificate chain were issued by a certificate authority. By default, any certificates for certificate authorities not meeting this criteria are rejected. This section describes the command-line argument that controls the level of certificate validation.

If WebLogic Server is booted with a certificate chains that will not pass the certificate validation, an information message is logged noting that clients could reject it.

Controlling the Level of Certificate Validation

By default WebLogic Server rejects any certificates in a certificate chain that do not have the Basic Constraint extension defined as CA. However, you may be using certificates that do not meet this requirement or you may want to increase the level of security to conform to the IETF RFC 2459 standard. Use the following command-line argument to control the level of certificate validation performed by WebLogic Server:

`-Dweblogic.security.SSL.enforceConstraints=`*`option`*

Table 8-1 describes the options for the command-line argument.

Table 8-1 Options for -Dweblogic.security.SSL.enforceConstraints

Option	Description

`strong` or `true`	Use this option to check that the Basic Constraints extension on the CA certificate is defined as CA.
	For example:
	`-Dweblogic.security.SSL.enforceConstraints=strong`
	or
	`-Dweblogic.security.SSL.enforceConstraints=true`
	By default, WebLogic Server performs this level of certificate validation.

Table 8-1 Options for -Dweblogic.security.SSL.enforceConstraints

Option	Description
`strict`	Use this option to check the Basic Constraints extension on the CA certificate is defined as CA and set to critical. This option enforces the IETF RFC 2459 standard. For example: `-Dweblogic.security.SSL.enforceConstraints=strict` This option is not the default because a number of commerically available CA certificates do not conform to the IETF RFC 2459 standard.
`off`	Use this option to disable certificate validation. Use this option carefully. For example, if you purchased CA certificates from a reputable commerical certificate authority and the certificates do not pass the new validation, use this option. However, CA certificates from most commerical certificate authorities should work with the default `strong` option. For example: `-Dweblogic.security.SSL.enforceConstraints=off` BEA does not recommend use of this option in production environment. Instead, purchase new CA certificates that comply with the IETF RFC 2459 standard.

Checking Certificate Chains

WebLogic Server provides a ValidateCertChain command-line utility to check whether or not an existing certificate chain will be rejected by WebLogic Server. The utility uses certificate chains from PEM files, PKCS-12 files, PKCS-12 keystores, and JKS keystores. A complete certificate chain must be used with the utility. The following is the syntax for the ValidateCertChain command-line utility:

```
java utils.ValidateCertChain -file pemcertificatefilename
java utils.ValidateCertChain -pem pemcertificatefilename
java utils.ValidateCertChain -pkcs12store pkcs12storefilename
java utils.ValidateCertChain -pkcs12file pkcs12filename password
java utils.ValidateCertChain -jks alias storefilename [storePass]
```

Example of valid certificate chain:

```
java utils.ValidateCertChain -pem zippychain.pem

Cert[0]: CN=zippy,OU=FOR TESTING
ONLY,O=MyOrganization,L=MyTown,ST=MyState,C=US

Cert[1]: CN=CertGenCAB,OU=FOR TESTING
ONLY,O=MyOrganization,L=MyTown,ST=MyState,C=US

Certificate chain appears valid
```

Example of invalid certificate chain:

```
java utils.ValidateCertChain -jks mykey mykeystore

Cert[0]: CN=corba1,OU=FOR TESTING ONLY,
O=MyOrganization,L=MyTown,ST=MyState,C=US

CA cert not marked with critical BasicConstraint indicating it is a CA
Cert[1]: CN=CACERT,OU=FOR TESTING ONLY,
O=MyOrganization,L=MyTown,ST=MyState,C=US

Certificate chain is invalid
```

Troubleshooting Problems with Certificate Validation

If SSL communications were working properly in a previous release of WebLogic Server and start failing unexpectedly, the problem is mostly likely because the certificate chain used by WebLogic Server is failing the validation.

Determine where the certificate chain is being rejected, and decide whether to update the certificate chain with one that will be accepted or change the setting of the `-Dweblogic.security.SSL.enforceConstraints` command-line argument.

To troubleshoot problems with certificates, use one of the following methods:

- If you know where the certificate chains for the processes using SSL communication are located, use the ValidateCertChain command-line utility to check whether the certificate chains will be accepted.
- Turn on SSL debug tracing on the processes using SSL communication. The syntax for SSL debug tracing is:

  ```
  -Dssl.debug=true -Dweblogic.StdoutDebugEnabled=true
  ```

The following message indicates the SSL failure is due to problems in the certificate chain:

```
<CA certificate rejected. The basic constraints for a CA certificate
were not marked for being a CA, or were not marked as critical>
```

When using one-way SSL, look for this error in the client log. When using two-way SSL, look for this error in the client and server logs.

Using the nCipher JCE Provider with WebLogic Server

Note: JCE providers are written using the application programming interfaces (APIs) in the Java Cryptography Extension (JCE) available in JDK 1.4. This type of provider is different from the providers written using the WebLogic Security Service Provider Interfaces (SSPIs). WebLogic Server does not provide a JCE provider by default.

SSL is a key component in the protection of resources available in Web servers. However, heavy SSL traffic can cause bottlenecks that impact the performance of Web servers. JCE providers offload SSL processing from Web servers freeing the servers to process more transactions. They also provide strong encryption and cryptographic process to preserve the integrity and secrecy of keys.

WebLogic Server supports the use of the following JCE providers:

- The JDK JCE provider (`SunJCE`) in the JDK 1.4.1. For more information about the features in the JDK JCE provider, see http://java.sun.com/products/jce.

  By default, the JCE provider in the JDK 1.4.1 has export strength jurisdiction policy files. After filling out the appropriate forms, the domestic strength jurisdiction policy files are downloadable from Sun Microsystem at http://java.sun.com/products/jce/index-14.html#UnlimitedDownload.

  The BEA license will continue to control the strength of the cryptography used by the WebLogic Server Application Programming Interfaces (APIs). Client code without the appropriate domestic strength cryptography license will only be able to use the J2SE export strength default cryptography. On the server, there will always be a BEA license that will enable either export or domestic strength cryptography.

- The nCipher JCE provider. For more information about the nCipher JCE provider, see http://www.ncipher.com/solutions/webservers.html.

To install the nCipher JCE provider:

1. Install and configure the hardware for the nCipher JCE provider per the product's documentation.

2. Install the files for the nCipher JCE provider. The following files are required:
   - JCE 1.2.1 framework JAR
   - Jurisdiction policy files
   - JCE provider
   - Certificate that signed the JAR file

   Note: This step may have been performed as part of installing the hardware for nCipher JCE provider. In that case, verify that the files are correctly installed.

   The files are installed in one of the following ways:

   - As an installed extension. Copy the files to one of the following locations:

     Windows NT

     `%JAVA_HOME%\jre\lib\ext`

     For example:

     `%WL_HOME%\jdk141\jre\lib\ext`

     UNIX

     `$JAVA_HOME/jre/lib/ext`

     For example:

     `$WL_HOME/jdk141/jre/lib/ext`

   - In the CLASSPATH of the server.

3. Edit the Java security properties file (`java.security`) to add the nCipher JCE provider to the list of approved JCE providers for WebLogic Server. The Java security properties file is located in:

   Windows NT

   `%JAVA_HOME%\jre\lib\security\java.security`

   UNIX

   `$JAVA_HOME/jre/lib/security/java.security`

   Specify the nCipher JCE provider as:

   `security.provider.`*n*`=com.ncipher.provider.km.mCipherKM`

   where

n specifies the preference order which determines the order in which providers are searched for requested algorithms when no specific provider is requested. The order is 1-based; 1 is the most preferred, followed by 2, and so on.

The nCipher JCE provider must follow the RSA JCA provider in the security properties file. For example:

```
security.provider.1=sun.security.provider.Sun
security.provider.2=com.sun.rsajca.Provider
security.provider.3=com.ncipher.provider.km.mCipherKM
```

4. Boot WebLogic Server.
5. To ensure the nCipher JCE provider is working properly, enable debugging per the nCipher product documentation.

Specifying the Version of the SSL Protocol

WebLogic Server supports both the SSL V3.0 and TLS V1.0 protocols. By default, Weblogic Server uses the SSL V3.0 protocol. While in most cases the SSL V3.0 protocol is acceptable there are circumstances (compatibility, SSL performance, and environments with maximum security requirements) where the TLS V1.0 protocol is desired. The `weblogic.security.SSL.protocolVersion` command-line argument allows you to specify what protocol is used for SSL connections.

When WebLogic Server acts as a client, the SSL handshake starts with an SSL V2.0 hello from WebLogic Server. The peer must respond with an SSL V3.0 or TLS V1.0 message or the SSL connection is dropped. This behavior is the default.

Note: The SSL V3.0 and TLS V1.0 protocols can not be interchanged. Only use the TLS V1.0 protocol if you are certain all desired SSL clients are capable of using the protocol.

The following command-line argument can be specified so that WebLogic Server supports only SSL V3.0 or TLS V1.0 connections:

- `-Dweblogic.security.SSL.protocolVersion=SSL3`—Only SSL V3.0 messages are sent and accepted.
- `-Dweblogic.security.SSL.protocolVersion=TLS1`—Only TLS V1.0 messages are sent and accepted.
- `-Dweblogic.security.SSL.protocolVersion=ALL`—This is the default behavior.

Using the SSL Protocol to Connect to WebLogic Server from weblogic.Admin

Note: The weblogic.Admin utility is deprecated in WebLogic Server 9.0. Use the WebLogic Scripting Tool (WLST) for equivalent functionality. For more information, see *WebLogic Scripting Tool*.

Using the SSL protocol to connect to WebLogic Server from `weblogic.Admin` requires you to disable two-way SSL on the server, use a secure server port in the URL for the client, specify trust for the client, and configure how the client uses host name verification. The following sections describe these steps in detail.

Ensure Two-Way SSL Is Disabled on the SSL Server

There is no way to specify identity when using `weblogic.Admin`. Identity (private key and digital certificate or certificate chain) is required when the SSL server is configured for two-way SSL. Therefore, two-way SSL cannot be enabled when using `weblogic.Admin`. Before establishing an SSL connection from `weblogic.Admin` to an SSL server, ensure that the SSL server is not configured to use two-way SSL. If two-way SSL is enabled on the SSL server, the SSL connection will fail.

Disabling Two-Way SSL is done using the WebLogic Server Administration Console. See the "Disable Two-Way SSL" topic in the online help

Use a Secure Port in the URL

To use the SSL protocol to make a connection, specify a secure protocol and port in the URL for `weblogic.Admin`. For example:

```
weblogic.Admin -url t3s://localhost:9002
```

Specify Trust for weblogic.Admin

All SSL clients need to specify trust. Trust is a set of CA certificates that specify which trusted certificate authorities are trusted by the client. In order to establish an SSL connection the client needs to trust the certificate authorities that issued the server's diglossia certificates.

When using `weblogic.Admin`, the trusted CA certificates must be stored in a keystore. By default, all the trusted certificate authorities available from the JDK (`...\jre\lib\security\cacerts`) are trusted by `weblogic.Admin`. Optionally, use the following command-line argument to specify a password for the JDK cacerts trust keystore:

```
-Dweblogic.security.JavaStandardTrustKeystorePassPhrase=password
```

where *password* is the password for the Java Standard Trust keystore. This password is defined when the keystore is created.

You also have the option of specifying the one of the following types of trust:

- Demo Trust—The trusted CA certificates in the demonstration Trust keystore (`DemoTrust.jks`) located in the `WL_HOME\server\lib` directory. In addition, the trusted CAs in the JDK `cacerts` keystore are trusted. To use the Demo Trust, specify the following command-line argument:

  ```
  -Dweblogic.security.TrustKeyStore=DemoTrust
  ```

  Optionally, use the following command-line argument to specify a password for the JDK cacerts trust keystore:

  ```
  -Dweblogic.security.JavaStandardTrustKeystorePassPhrase=password
  ```

  where *password* is the password for the Java Standard Trust keystore. This password is defined when the keystore is created.

- Custom Trust—A trust keystore you create. To use Custom Trust, specify the following command-line arguments:

  - `weblogic.security.TrustKeyStore=CustomTrust`

    This required command-line argument specifies the use of Custom Trust

  - `weblogic.security.CustomTrustKeyStoreFileName=filename`

    This required command-line argument specifies the fully qualified path to the trust keystore

  - `weblogic.security.TrustKeystoreType=keystore_type`

    This optional command-line argument specifies the type of the keystore. Generally, this value for type is the default value, `jks`.

  - `weblogic.security.CustomTrustKeystorePassPhrase=password`

    This optional command-line argument specifies the password defined when creating the keystore.

Specify Host Name Verification for weblogic.Admin

By default, `weblogic.Admin` performs a host name verification check. It compares the CN field in the digital certificate received from the server with the server name in the URL the client used to connect to the server. The CN field and the server name must match to pass the host name

verification check. This check is performed to prevent man-in-the-middle attacks. In this release of WebLogic Server, the default host name verifier handles the case where the URL contains localhost or an IP address and the CN field of the digital certificate matches the name of the local host.

It is possible to disable the check by specifying the following command-line argument:

```
-Dweblogic.security.SSL.ignoreHostnameVerification=true
```

Note: If the SSL server specified an IP address in its URL, disable the host name verification check.

Use the following command-line argument to specify a custom host name verifier:

```
-Dweblogic.security.SSL.hostnameVerifier=classname
```

where *`classname`* specifies the implementation of the `weblogic.security.SSL.HostnameVerifier` interface.

CHAPTER 9

Configuring SSL for Node Manager

This chapter describes SSL requirements for the Node Manager and describes two scenarios: using the demonstration identity and trust provided by WebLogic Server and using production-quality identity and trust.

The following sections describe how to configure SSL for the Node Manager:

Before You Begin

The Node Manager uses two-way SSL to protect communications with Administration and Managed Servers. The configuration of SSL involves obtaining identity and trust for the Node Manager and each Administration and Managed Server with which the Node Manager will be communicating and then configuring the Node Manager, the Administration Server, and any Managed Servers with the proper identity and trust. In addition, the use of the Administration port must be taken into consideration.

Before performing the steps in this chapter, review the following information:

- "Configuring Identity and Trust"
- "Configuring the SSL Protocol"

SSL Requirements for Administration Servers

To use SSL, Administration Servers require:

- Identity—Administration Servers use private keys and digital certificates stored in keystores. The digital certificate must specify a host name rather than an IP address. The location of the Identity keystore is specified in the WebLogic Server Administration Console.

  By default, Administration Servers are configured to use the demonstration Identity keystore (`DemoIdentity.jks`). For testing and development purposes the Identity keystore configuration is complete.

- Trust—Administration Servers use trusted CA certificates stored in a keystore. The location of the Trust keystore is specified in the WebLogic Server Administration Console.

  By default, Administration Servers are configured to use the demonstration Trust keystore (`DemoTrust.jks`) and the Java Standard Trust keystore (*`JAVA_HOME`*`\jre\lib\security\cacerts`). For testing and development purposes the Trust keystore configuration is complete.

  In a production environment, a custom Trust keystore (a keystore you create) or the Java Standard Trust keystore can be used with the Administration Server. The Administration Server needs to trust the certificate authorities for all Node Managers running on machines in this domain and any Managed Servers (meaning those trusted CA certificates must be in the Trust keystore of the Administration Server). This requirement applies only if the Administration port is being used.

- Host name verification—By default, host name verification is enabled on Administration Servers.
- Administration port—The use of the Administration port is optional when configuring SSL. The Administration port uses SSL internally to protect all Administration communications. By default, there is no Administration port enabled.

  BEA recommends using the Administration port in a production environment. All Administration Servers, Managed Servers, and Node Managers on the same machine must use unique numbers for the Listen port.

SSL Requirements for Managed Servers

To use SSL, Managed Servers require:

- Identity—Managed Servers use private keys and digital certificates stored in keystores. The digital certificate must specify a host name rather than an IP address. The location of the Identity keystore is specified in the WebLogic Server Administration Console.

  By default, Managed Servers are configured to use the demonstration Identity keystore (`DemoIdentity.jks`). For testing and development purposes the Identity keystore configuration is complete. However, the demonstration keystore should not be used in a production environment.
- Trust—Managed Servers use trusted CA certificates stored in a keystore. The location of the Trust keystore is specified in the WebLogic Server Administration Console.

  By default, Managed Servers are configured to use the demonstration Trust keystore (`DemoTrust.jks`) and the Java Standard Trust keystore (*`JAVA_HOME`*`\jre\lib\security\cacerts`). For testing and development purposes the Trust keystore configuration is complete.

  In a production environment, a custom Trust keystore (a keystore you create) or the Java Standard Trust keystore can be used with a Managed Server. A Managed Server needs to trust the certificate authorities for all Node Managers running on machines in this domain and the Administration Server (meaning those trusted CA certificates must be in the trust keystore of the Managed Server). This requirement applies only if the Administration port is being used.
- Host name verification—By default, host name verification is enabled on Managed Servers.

- Administration port—The use of the Administration port is optional when configuring SSL. The Administration port uses SSL internally to protect all Administration communications. By default, there is no Administration port enabled.

  BEA recommends using the Administration port in a production environment. All Administration Servers, Managed Servers, and Node Managers on the same machine must use unique numbers for the Administration port.

SSL Requirements for the Node Manager

To use SSL, the Node Manager requires:

- Keystores—Specify the type of keystore configuration to be used by the Node Manager. By default, the Node Manager is configured to use DemoIdentityandDemoTrust as the keystore configuration.

  The keystore configuration is specified in the `nodemanager.properties` file by the following properties:

  `Keystores=CustomIdentityandCustomTrust`

  `Keystores=CustomIdentityandJavaStandardTrust`

  For more information, see Node Manager Properties in *Configuring and Managing WebLogic Server.*

- Identity—The Node Manager uses private keys and digital certificates stored in keystores. The digital certificate must specify a host name rather than an IP address.

  By default, the Node Manager is configured to use the demonstration Identity keystore (`DemoIdentity.jks`). For testing and development purposes the Identity keystore configuration is complete. However, this demonstration Identity keystore should not be used in a production environment.

  The Identity keystore is specified in the `nodemanager.properties` file by the following properties:

  `CustomIdentityKeystoreType`

  `CustomIdentityAlias`

  `CustomIdentityKeystoreFileName`

  `CustomIdentityKeystorePassPhrase`

  `CustomIdentityPrivateKeyPassPhrase`

  For more information, see Node Manager Properties in *Configuring and Managing WebLogic Server.*

- Trust—The Node Manager uses trusted CA certificates stored in a keystore. You can either use a keystore you create (custom) or the keystore in the JDK (Java Standard).

  By default, the Node Manager is configured to use the demonstration Trust keystore (`DemoTrust.jks`) and the Java Standard Trust keystore (*`JAVA_HOME`*`\jre\lib\security\cacerts`). For testing and development purposes the Trust keystore configuration is complete.

  The Trust keystore is specified in the `nodemanager.properties` file by the following properties:

  Custom Trust keystore:

  `CustomTrustKeystoreType`

  `CustomTrustKeystoreFileName`

  `CustomTrustKeystorePassPhrase`

  Java Standard Trust keystore:

  `JavaStandardTrustKeystorePassphrase`

  For more information, see Node Manager Properties in *Configuring and Managing WebLogic Server.*

  In a production environment, the Node Manager must also trust the certificate authorities used by the Administration Server and any Managed Servers running on its machine (meaning those trusted CA certificates must be in the Trust keystore of the Node Manager).

- Trusted Hosts—The Node Manager accepts commands only from Administration Servers that reside on a trusted hosts. The trusted hosts for a Node Manager process are identified by IP address or DNS name in a file. By default, the file is named `nodemanager.hosts` and installed in the *`WL_HOME`*`\common\nodemanager\config` directory.

  There is one Node Manager per machine, however, domains managed by the Node Manager can have multiple machines. Make sure the `nodemanager.hosts` file lists the machines (hosts) for the Administration Servers for any domain you want to run from this machine. By default, the `nodemanger.hosts` file always defaults to `localhost`.

  The hosts may be specified by IP address or name. If you want to use host names, specify the `ReverseDNSEnabled` property in the `nodemanager.properties` file.

  For more information, see Set Up the Node Manager Hosts File.

Host Name Verification Requirements

In previous releases of WebLogic Server, you had to enable host name verification. In this release of WebLogic Server, host name verification is enabled by default. BEA recommends using host name verification to ensure the host you are connecting to is the intended host. To avoid errors when using host name verification, check the following:

- All digital certificates have host names rather than IP addresses.
- All digital certificates have the correct host name.
- All URLs use host names rather than IP addresses.
- The Listen Address defined for the Node Manager, Administration Server, and any Managed Servers matches the host name in the digital certificate.
- Host name verification is enabled on the Node Manager, the Administration Server, and any Managed Servers.

Identity and Trust: Demonstration Versus Production

By default, the Node Manager, the Administration Server, and any Managed Servers, are configured to use the demonstration Identity keystore (`DemoIdentity.jks`), the demonstration Trust keystore (`DemoTrust.jks`), and the Java Standard Trust keystore (*`JAVA_HOME`*`\jre\lib\security\cacerts`). For testing and development purposes the Identity and Trust keystore configuration is complete.

However, the demonstration keystores should not be used in a production environment. All the digital certificates and trusted CA certificates in the keystores are signed by a WebLogic Server demonstration certificate authority. Therefore, all WebLogic Server installations trust each other. This will leave your SSL connections wide open to a number of security attack.

Keystores are configured on a per machine basis. You can share keystores and identity and trust for Administration Servers, Managed Servers, and Node Managers that run on the same machine.

:Perform the following steps to configure identity and trust for a production environment:

1. Obtain private keys and digital certificates for the Node Manager, Administration Servers, and Managed Servers from a reputable certificate authority. For more information, see “Obtaining Private Keys, Digital Certificates, and Trusted Certificate Authorities.”

2. Ensure Identity and Trust keystores for the Node Manager, Administration Server, and any Managed Servers exist. In previous releases, WebLogic Server only supported JKS keystores. In this release, WebLogic Server can access private keys and trusted CA certificates from any type of keystore. When you configure the keystore in the WebLogic Server Administration Console, you optionally specify its type.

   In a production environment, you can also use the Java Standard Trust keystore (*JAVA_HOME*`\jre\lib\security\cacerts`) as the Trust keystore for the Node Manager, the Administration Server, or any Managed Servers.

3. Load the private keys and certificates into the Identity keystore and trusted CA certificates into the Trust keystore.

4. Configure the keystore location and passwords for the Administration Server and Managed Server in the WebLogic Server Administration Console.

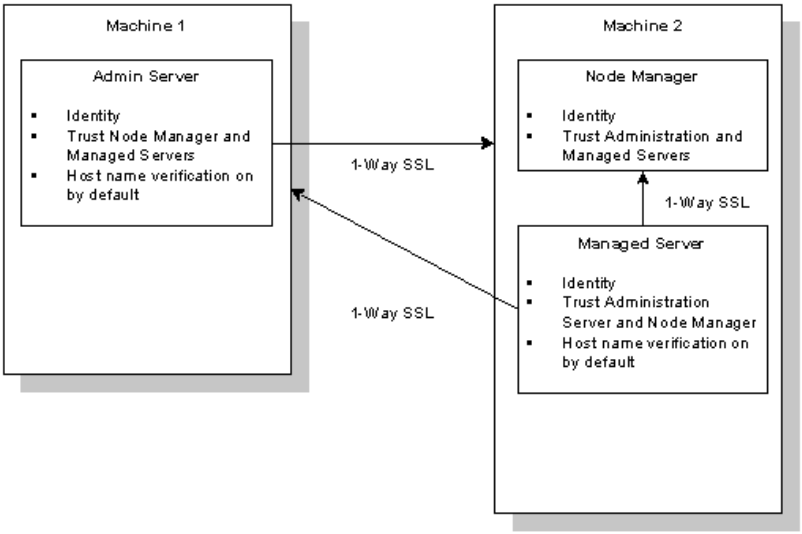
5. Edit the `nodemanager.properties` file to specify the keystore location and passwords for the Node Manager.

Node Manager SSL Demonstration Configuration: Main Steps

Using the demonstration Identity and Trust keystores provided by WebLogic Server to configure SSL for the Node Manager involves verifying the default settings for the keystore configuration options and ensuring that the Administration Server and any Managed Servers are listening for SSL communications on different ports.

Figure 9-1 illustrates the SSL demonstration configuration for the Node Manager.

Figure 9-1 SSL Demonstration Configuration for the Node Manager



Note: The following procedure assumes the Node Manager, the Administration Server, and all Managed Servers are running on the same machine.

To configure the Node Manager to use SSL and the demonstration Identity and Trust keystores:

1. Run the Configuration Wizard and create a new WebLogic domain.
2. Start the Administration Server. Do not enable the Administration port.
3. On the Server-->Keystores and SSL page, verify the settings of the following Identity keystore configuration options for the Administration Server:
   - Demo Identity Keystore—`DemoIdentity.jks`
   - Type—`JKS`
   - Passphrase—`DemoIdentityKeyStorePassPhrase`
4. On the Server-->Keystores and SSL page, verify the settings of the following Trust keystore configuration options for the Administration Server:
   - Demo Trust Keystore—`DemoTrust.jks`
   - Type—`JKS`
   - Passphrase—`DemoTrustKeyStorePassPhrase`

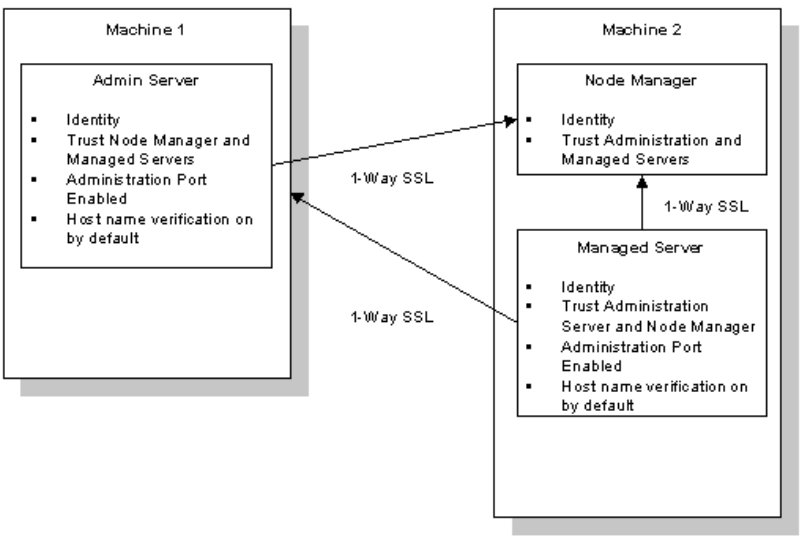
5. On the Server-->Keystores and SSL page, verify the settings of the following Identity keystore configuration options for the Managed Server:
   - Demo Identity Keystore—`DemoIdentity.jks`
   - Type—`JKS`
   - Passphrase—`DemoIdentityKeyStorePassPhrase`
6. On the Server-->Keystores and SSL page, verify the settings of the following Trust keystore configuration options for the Managed Server:
   - Demo Trust Keystore—`DemoTrust.jks`
   - Type—`JKS`
   - Passphrase—`DemoTrustKeyStorePassPhrase`

   Note: No changes to the `nodemanager.properties` file are required. The Node Manager will automatically default to the demonstration Identity and Trust keystores.

Node Manager SSL Production Configuration: Main Steps

Figure 9-2 illustrates the SSL production configuration for the Node Manager.

Figure 9-2 SSL Production Configuration for the Node Manager



Note: The following procedure assumes the Node Manager, the Administration Server, and all Managed Servers are running on the same machine.

Warning: When you configure keystores through the WebLogic Administration Servers, passwords are available in clear text during the configuration process. The passwords will be encrypted automatically when the configuration is complete and the Node Manager is started. For a more secure deployment, BEA recommends taking the machine on which you are configuring the Node Manager off the Internet or ensure the machine is protected behind a firewall so that passwords can not be snooped.

To configure SSL for the Node Manager:

1. Obtain identity and trust for the Node Manager, the Administration Server, and any Managed Servers.

Note: When obtaining identity and trust for the Administration Server, any Managed Server(s), and the Node Manager, ensure the digital certificates include the machine's host name (so that the digital certificate matches the URL).

2. Create Identity keystores for the Node Manager, the Administration Server, and any Managed Servers. If the Node Manager, the Administration Server, and the Managed Servers run on the same machine, they can share the Identity keystore. See "Storing Private Keys, Digital Certificates, and Trusted Certificate Authorities."
3. Load the private keys into the Identity keystores. See "Storing Private Keys, Digital Certificates, and Trusted Certificate Authorities."
4. Create Trust keystores for the Node Manager, Administration Server, and any Managed Servers. If the Node Manager, the Administration Server, and the Managed Servers run on the same machine, they can share the Trust keystore. See "Storing Private Keys, Digital Certificates, and Trusted Certificate Authorities."
5. Load the trusted CA certificates into the Trust keystore. See "Storing Private Keys, Digital Certificates, and Trusted Certificate Authorities."
6. Run the Configuration Wizard and create a new WebLogic domain.
7. Start the Administration Server.
8. Enable the Administration port for the WebLogic domain. See Enabling the Domain-Wide Administration Port.

9. Set the Listen Address for the Administration Server. The Listen Address is the host name of the machine on which the Administration Server runs. This host name should match the host name in the CN field of the digital certificate for the Administration Server. If you don't set the Listen Address correctly, you may encounter host name verification errors.

   Use the following command to determine the host name specified in the CN field of the digital certificate:

   `keytool -list -v -keystore` *`fulldirectorypathtokeystore\keystorename`*

10. Configure the Identity keystore for the Administration Server. See "Configuring Keystores."
11. Configure the Trust keystore for the Administration Server. See "Configuring Keystores."
12. If you have multiple servers running on the same machine, ensure the Administration Port for each server is unique. Use Local Administration Port Override on the Server-->General page to override the Administration Port number. Specify a port number other than the Administration port number used by the Administration Server.
13. Set the Listen Address for the Managed Server to the host name of the machine on which the Managed Server runs. This host name should match the host name in the digital certificate for the Managed Server.
14. Configure the Identity keystore for the Managed Server. See "Configuring Keystores."
15. Configure the Trust keystore for the Managed Server. See "Configuring Keystores."
16. Create a machine for the Node Manager.
    a. Set the Listen Address of the Node Manager to the host name of the machine on which the Node Manager runs. This host name should match the host name in the digital certificate for the Node Manager.
    b. Set the Listen Port for the Node Manager. Ensure that this port number is different from the Listen port number, the SSL port number, and the Administration port number of all the servers that run on this machine.
    c. Add any Managed Servers to the machine.

       For more information, see Configuring, Starting, and Stopping the Node Manager in *Configuring and Managing WebLogic Server.*
17. Edit the `nodemanager.properties` file for the Node Manager. Edit the `nodemanager.properties` file.

a. Specify the keystore configuration to be used:

   `Keystores=CustomIdentityandCustomTrust`

   `Keystores=CustomIdentityandJavaStandardTrust`

b. Specify information about the Identity keystore:

   `CustomIdentityKeystoreType`

   `CustomIdentityAlias`

   `CustomIdentityKeystoreFileName`

   `CustomIdentityKeystorePassPhrase`

   `CustomIdentityKeystoreType` is optional and defaults to the keystore type defined in the security policy file for the JDK.

   `CustomIdentityKeystorePassPhrase` is optional depending on the type of keystore. All keystores require the passphrase in order to write to the keystore. Some keystores do not require the passphrase to read from the keystore. Whether or not you define this property depends on the requirements of the keystore. For example, WebLogic Server only reads from the keystore so a passphrase is not required, however, WebLogic Integration writes to keystores and therefore requires a passphrase.

   `CustomIdentityPrivateKeyPassPhrase`

c. Specify information about the Trust keystore.

   If you use a custom Trust keystore, specify:

   `CustomTrustKeystoreType`

   `CustomTrustKeystoreFileName`

   `CustomTrustKeystorePassPhrase`

   `CustomTrustKeystoreType` is optional and defaults to the keystore type defined in the security policy file for the JDK.

   `CustomTrustKeystorePassPhrase` is optional depending on the type of keystore. All keystores require the passphrase in order to write to the keystore. Some keystores do not require the passphrase to read from the keystore. Whether or not you define this property depends on the requirements of the keystore. For example, WebLogic Server only reads from the keystore so a passphrase is not required, however, WebLogic Integration writes to keystores and therefore requires a passphrase.

   If you use the Java Standard Trust keystore, specify:

   `Keystores=CustomIdentityandJavaStandardTrust`

`JavaStandardTrustKeystorePassPhrase`

`JavaStandardTrustKeystorePassPhrase` is optional depending on the type of keystore. All keystores require the passphrase in order to write to the keystore. Some keystores do not require the passphrase to read from the keystore. Whether or not you define this property depends on the requirements of the keystore. For example, WebLogic Server only reads from the keystore so a passphrase is not required; however, WebLogic Integration writes to keystores and therefore requires a passphrase.

d. Specify a Listen Address and Listen Port for the Node Manager:

```
ListenAddress
ListenPort
```

Use the same host name as specified when creating the machine in step 16.

e. If you want use host names rather than IP addresses, specify:

```
ReverseDnsEnabled
```

See Node Manager Properties in *Configuring and Managing WebLogic Server*.

Note: When adding properties to the `nodemanager.properties` file, use forward slashes or double backslashes in file and directory names.

18. Edit the `nodemanager.hosts` file located in *`WL_HOME`*`\common\nodemanager\config` to list the machines on which the Administration Servers that communicate with the Node Manager run. The Administration Servers are specified by IP address or host name if Reverse DNS is used.

    Ensure that each machine in a domain has an updated `nodemanager.hosts` file. By default, the `nodemanger.hosts` file defaults to `localhost`.

    See Set Up the Node Manager Hosts File in *Configuring and Managing WebLogic Server*.

19. Start the Node Manager.
20. Stop the Administration Server.
21. Start the Administration Server.

Using Files and the WebLogic Keystore Provider

Note: For backward compatibility, WebLogic Server supports using files and the WebLogic Keystore provider as a way to store identity and trust when configuring the Node Manager to use SSL. However, both of these methods are deprecated in this release. Also, private keys stored in files may or may not be password protected. Private keys that

are not password protected can be vulnerable to security attacks. BEA recommends upgrading to keystores as a way to store identity and trust for the Node Manager, the Administration Server, and any Managed Servers.

The SSL requirements for identity and trust are as follows:

- Administration Servers and Managed Servers use private keys stored in JKS keystores accessed through the WebLogic Keystore provider or in a file.
- The digital certificate for an Administration Server and a Managed Server must be stored in a file.
- Administration Servers and Managed Servers can use trusted CA certificates stored:
  - In a JKS keystore specified by the `-Dweblogic.security.SSL.trustedCAKeyStore` command-line in the start script for the Administration Server or Managed Server.
  - In a JKS keystore accessed by the WebLogic Keystore provider
  - A file containing the PEM-encoded trusted certificate authorities.

  If no trusted CA certificate is located in either of these storage mechanisms, WebLogic Server assumes anyAdministration Server and Managed Servers trust all the certificate authorities in the `cacerts` files in the *`WL_HOME`*`\server\lib`.
- The Node Manager can only use digital certificates and private keys stored in files. Identity is specified by the command-line arguments for the Node Manager.
- The Node Manager uses trusted CA certificates stored in a JKS keystore for trust. The Trust keystore is specified by the command-line arguments for the Node Manager

Note: Perform the following steps on the Administration Server and each Managed Server you plan to use.

To use files or the WebLogic Keystore provider to store identity and trust for an Administration Server or a Managed Server:

1. Expand the Servers node.
2. Select the name of the server for which you want to configure identity and trust (for example, exampleserver).
3. Select the Configuration-->Keystores and SSL page.

   Information about the demonstration Identity and Trust keystores is displayed.
4. Click the Change... link in the SSL Configuration portion of the page.

The Configure SSL page appears.

5. Choose the Files or Keystore Providers option.
6. Click Continue.
7. Specify information about the location of identity and trust for WebLogic Server.
   – Private Key File Name—The directory location of the private key for WebLogic Server. Specify a value only if you stored the private key for WebLogic Server in a file (versus a WebLogic Keystore provider).

Configuring Security for a WebLogic Domain

The following sections describe how to set security configuration options for a WebLogic domain:

Note: This chapter applies to WebLogic Server deployments using the security features in this release of WebLogic Server as well as deployments using Compatibility Security.

Enabling Trust Between WebLogic Server Domains

Note: BEA Systems does not recommend enabling trust between WebLogic Server domains in a production environment unless the servers have a secure means of communication such as a dedicated line or are protected by a firewall.

Trust between domains is established so that principals in a Subject from one WebLogic Server domain are accepted as principals in another domain. When this feature is enabled, identity is passed between WebLogic Server domains over an RMI connection without requiring authentication in the second domain (for example: login to Domain 1 as Joe, make an RMI call to Domain 2 and Joe is still authenticated). When inter-domain trust is enabled, transactions can commit across domains. A trust relationship is established when the Domain Credential for one domain matches the Domain Credential for another domain.

The domain credential is randomly created the first time a WebLogic Server domain is started. This process ensures that by default no two WebLogic Server domains have the same credential. To enable trust between two WebLogic Server domains, you must explicitly specify the same value for the credential in both WebLogic Server domains. Use the configuration options on the Security Configuration-->Advanced page under the Domains node to set domain credentials.

By default, when you boot an Administration Server for the first time, the Domain Credential is not defined. As the Administration Server boots, it notices that the Domain Credential is not defined and generates a random credential.

WebLogic Server signs Principals with the Domain Credential as Principals are created. When a Subject is received from a remote source, its Principals are validated (the signature is recreated and if it matches, the remote domain has the same Domain Credential). If validation fails, an error is generated. If validation succeeds, the Principals are trusted as if they were created locally.

Note: Any credentials in clear text are encrypted the next time the `config.xml` file is persisted to disk.

If you want a WebLogic Server 6.x domain to interoperate with a WebLogic Server domain, change Domain Credential in the WebLogic Server domain to the password of the `system` user in the WebLogic Server 6.x domain.

If you are enabling trust between domains in a managed server environment, you must stop the Administration server and all the Managed Servers in both domains and then restart them. If this step is not performed, servers that were not rebooted will not trust the servers that were rebooted.

Keep the following points in mind when enabling trust between WebLogic Server domains:

- Because a domain will trust remote Principals without requiring authentication, it is possible to have authenticated users in a domain that are not defined in the domain's authentication database. This situation can cause authorization problems.
- Any authenticated user in a domain can access any other domain that has trust enabled with the original domain without re-authenticating. There is no auditing of this login and group membership is not validated. Therefore, if Joe is a member of the Administrators group in the original domain where he authenticated, he is automatically a member of the Administrators group for all trusted domains to which he makes RMI calls.
- If Domain 2 trusts both Domain 1 and Domain 3, Domain 1 and Domain 3 now implicitly trust eachother. Therefore, members of the Administrators Group in Domain 1 are members of the Administrators group in Domain 3. This may not be a desired trust relationship.

- If you extended the WLSUser and WLSGroup Principal classes, the custom Principal classes must be installed in the CLASSPATH in all domains that share trust.

For more information:

- See "Enable trust between domains" in the Administration Console online help.
- You can also use the WebLogic Scripting tool or Java Management Extensions (JMX) APIs can to create a new security configuration.

Using Connection Filters

Connection filters allow you to deny access at the network level. They can be used to protect server resources on individual servers, server clusters, or an entire internal network or Intranet. For example, you can deny any non-SSL connections originating outside of your corporate network. Network connection filters are a type of firewall in that they can be configured to filter on protocols, IP addresses, and DNS node names.

Connection filters are particularly useful when using the Administration port. Depending on your network firewall configuration, you may be able to use a connection filter to further restrict administration access. A typical use might be to restrict access to the Administration port to only the servers and machines in the domain. An attacker who gets access to a machine inside the firewall, still cannot perform administration operations unless the attacker is on one of the permitted machines.

WebLogic Server provides a default connection filter called `weblogic.security.net.ConnectionFilterImpl`. This connection filter accepts all incoming connections and also provides static factory methods that allow the server to obtain the current connection filter. To configure this connection filter to deny access, simply enter the connection filters rules in the WebLogic Server Administration Console.

You can also use a custom connection filter by implementing the classes in the `weblogic.security.net` package. For information about writing a connection filter, see "Using Network Connection Filters" in *Programming WebLogic Security*. Like the default connection filter, custom connection filters are configured in the WebLogic Server Administration Console.

To configure a connection filter:

1. Enable the logging of accepted messages. This Connection Logger Enabled option logs successful connections and connection data in the server. This information can be used to debug problems relating to server connections.

2. Choose which connection filter is to be used in the domain.
   - To configure the default connection filter, specify `weblogic.security.net.ConnectionFilterImpl` in Connection Filter.
   - To configure a custom connection filter, specify the class that implements the network connection filter in Connection Filter. This class must also be specified in the CLASSPATH for WebLogic Server.
3. Enter the syntax for the connection filter rules.

For more information:

- See "Configure connection filters" in the Administration Console online help.
- For information about connection filter rules and writing a custom connection filter, see "Using Network Connection Filters" and "Using Network Connection Filters" in *Programming WebLogic Security.*
- You can also use the WebLogic Scripting tool or Java Management Extensions (JMX) APIs can to create a new security configuration.

Viewing MBean Attributes

The Anonymous Admin Lookup Enabled option specifies whether anonymous, read-only access to WebLogic Server MBeans should be allowed from the `MBeanHome` API. With this anonymous access, you can see the value of any MBean attribute that is not explicitly marked as protected by the Weblogic Server MBean authorization process. This option is checked by default to sure backward compatibility.

To verify the setting of the Anonymous Admin Lookup Enabled option through the WebLogic Server Administration Console, see the "View MBean attributes" topic in the online help.

How Passwords are Protected in WebLogic Server

It is important to protect passwords that are used to access resources in a WebLogic Server domain. In the past, usernames and passwords were stored in clear text in a WebLogic security realm. Now all the passwords in a WebLogic Server domain are hashed. The `SerializedSystemIni.dat` file contains the hashes for the passwords. It is associated with a specific WebLogic Server domain so it cannot be moved from domain to domain.

If the `SerializedSystemIni.dat` file is destroyed or corrupted, you must reconfigure the WebLogic Server domain. Therefore, you should take the following precautions:

- Make a backup copy of the `SerializedSystemIni.dat` file and put it in a safe location.
- Set permissions on the `SerializedSystemIni.dat` file such that the system administrator of a WebLogic Server deployment has write and read privileges and no other users have any privileges.

Protecting User Accounts

WebLogic Server defines a set of configuration options to protect user accounts from intruders. In the default security configuration, these options are set for maximum protection. When creating a new security realm, you need to define options on the Security-->Realm User Lockout page.

As a system administrator, you have the option of turning off all the configuration options, increasing the number of login attempts before a user account is locked, increasing the time period in which invalid login attempts are made before locking the user account, and changing the amount of time a user account is locked. Remember that changing the configuration options lessens security and leaves user accounts vulnerable to security attacks.

Notes: The User Lockout options apply to the default security realm and all its security providers. The User Lockout options do not work with custom security providers in a security realm other than the default security realm. To use the User Lockout options with custom security providers, configure the custom security providers in the default security realm. Include the customer providers in the authentication process after the WebLogic Authentication provider and the WebLogic Identity Assertion provider. This ordering may cause a small performance hit.

If you are using an Authentication provider that has its own mechanism for protecting user accounts, disable Lockout Enabled.

If a user account becomes locked and you delete the user account and add another user account with the same name and password, the UserLockout configuration options will not be reset.

Unlocking a locked user account on a managed server cannot be done through the WebLogic Server Administration Console. The unlock information is propagated through a multicast message which is only configured in a cluster environment. Use the following command instead:

```
java weblogic.Admin -url url -username adminuser
-password passwordforadminuser -type
weblogic.mangement.security.authentication.UserLockoutManager
-method clearlockout lockedusername
```

You can also wait the time specified in the Lockout Duration attribute, the user account will be unlocked after the specified time.

For more information:

- See “Creating Users” in *Securing WebLogic Resources*.
- See “Set lockout options” in the Administration Console online help.
- See “Unlock a user account” in the Administration Console online help.
- You can also use the WebLogic Scripting tool or Java Management Extensions (JMX) APIs can to create a new security configuration.

CHAPTER 11

Using Compatibility Security

The following sections describe how to configure Compatibility security:

Note: Compatibility security is deprecated in this release of WebLogic Server. You should only use Compatibility security while upgrading your WebLogic Server deployment to the security features in this release of WebLogic Server.

Running Compatibility Security: Main Steps

To set up Compatibility security:

1. Make a backup copy of your 6.x WebLogic domain (including your `config.xml` file) before using Compatibility security.
2. Add the following to the 6.x config.xml file if it does not exist:

```
<Security Name="mydomain" Realm="mysecurity"/>
<Realm Name="mysecurity" FileRealm="myrealm"/>
<FileRealm Name="myrealm"/>
```

3. Install WebLogic Server in a new directory location. Do not overwrite your existing 6.x installation directory. For more information, see *Installing WebLogic Platform*.
4. Modify the start script for your 6.x server to point to the new WebLogic Server installation. Specifically, you need to modify:
   - The classpath to point to the `weblogic.jar` file in the new WebLogic Server installation.
   - `The JAVA_HOME` variable to point to the new WebLogic Server installation.
5. Use the start script for your 6.x server to boot WebLogic Server.

To verify whether you are correctly running Compatibility security, do the following:

1. In the WebLogic Server Administration Console, expand the Domain node.
2. Select the desired WebLogic Server domain (referred to as the domain).
3. Click the View the Domain Log link.

   The following message appears in the log:

   ```
   Security initializing using realm CompatibilityRealm
   ```

In addition, a CompatibilitySecurity node will appear in the WebLogic Server Administration Console.

The Default Security Configuration in the CompatibilityRealm

By default, the *CompatibilityRealm* is configured with a Realm Adapter Adjudication provider, a Realm Adapter Authentication provider, a WebLogic Authorization provider, a Realm Adapter Authorization provider, a WebLogic Credential Mapping provider, and a WebLogic Role Mapping provider.

- In the *CompatibilityRealm*, the Realm Adapter Authentication provider is populated with users and groups from the 6.x security realm defined in the `config.xml` file.
  - If you were using the File realm in your 6.x security configuration, you can manage the users and groups in the Realm Adapter Authentication provider following the steps in "Defining Users in the CompatibiltyRealm" and "Defining Groups in the CompatibiltyRealm" topics of the Compatiblility Security section of the Administration Console online help.

  - If you are using an alternate security realm (LDAP, Windows NT, RDBMS, or custom), you must use the administration tools provided by that realm to manage users and groups.

  If you have large numbers of users and groups stored in a Windows NT, RDBMS, UNIX, or a custom security realm and you cannot upgrade to a WebLogic, LDAP or custom Authentication provider, you can configure a Realm Adapter Authentication provider in the new security realm to access your existing 6.x store.

  Note: The Realm Adapter Authentication provider is the only Realm Adapter provider that can be configured in a realm other than the CompatibiltyRealm.

  For information about configuring a Realm Adapter Authentication provider, see "Configuring a Realm Adapter Authentication Provider."

  You can use implementations of the `weblogic.security.acl.CertAuthenticator` class in Compatibility security by configuring the Identity Assertion provider in the Realm Adapter Authentication provider. For more information, see "Configuring the Identity Assertion Provider in the Realm Adapter Authentication Provider."

- Access Control Lists (ACLs) in the 6.x security realm are used to populate the Realm Adapter Authorization provider.
- The Realm Adapter Adjudication provider enables the use of both ACLs and security roles and security policies in Compatibility security. The Realm Adapter Adjudication provider resolves access decision conflicts between ACLs and new security policies set through the WebLogic Server Administration Console.
- The WebLogic Credential Mapping provider allows the use of credential maps in Compatibility security. For more information, see *Programming WebLogic Resource Adapters*.
- You can add a Realm Adapter Auditing provider to access implementations of the `weblogic.security.audit.AuditProvider` class from the *CompatibilityRealm*. For more information, see "Configuring a Realm Adapter Auditing Provider."

Configuring the Identity Assertion Provider in the Realm Adapter Authentication Provider

The Realm Adapter Authentication provider includes an Identity Assertion provider.The Identity Assertion provider provides backward compatibility for implementations of the `weblogic.security.acl.CertAuthenticator` class. The identity assertion is performed on

X.509 tokens. By default, the Identity Assertion provider is not enabled in the Realm Adapter Authentication provider.

For more information, see “Configure an Identity Assertion provider” in the Administration Console online help.

Configuring a Realm Adapter Auditing Provider

The Realm Adapter Auditing provider allows you to use implementations of the `weblogic.security.audit.AuditProvider` class when using Compatibility security. In order for the Realm Adapter Auditing provider to work properly, the implementation of the `weblogic.security.audit.AuditProvider` class must have been defined in Audit Provider Class on the Domain-->Security-->Compatibility-->General page.

For information, see “Configure a Realm Adapter Auditing provider” in the Administration Console online help.

Protecting User Accounts in Compatibilty Security

Weblogic Server provides a set of configuration options to protect user accounts from intruders. By default, these options are set for maximum protection. As a system administrator, you have the option of turning off all the options, increasing the number of login attempts before a user account is locked, increasing the time period in which invalid login attempts are made before locking the user account, and changing the amount of time a user account is locked. Remember that changing the configuration options lessens security and leaves user accounts vulnerable to security attacks.

There are two sets of configuration options available to protect user accounts, one set at the domain and one set at the security realm. You may notice that if you set one set of configuration options (for example, the options for the security realm) and exceed any of the values, the user account is not locked. This happens because the user account options at the domain override the user account options at the security realm. To avoid this situation, disable the user account options at the security realm.

Warning: If you disable the user lockout configuration option at the security realm, you must set the user lockout configuration options on the domain otherwise the user accounts will not be protected.

For information:

See “Protect User Accounts” in the Administration Console online help.

See “Unlock user accounts” in the Administration Console online help.

.Accessing 6.x Security from Compatibility Security

When using Compatibility security, it is assumed you have an existing `config.xml` file with a security realm that defines users and groups and ACLs that protect the resources in your WebLogic Server domain. 6.x security management tasks such as configuring a security realm or defining ACLs should not be required therefore those management tasks are not described in this chapter. However, if you corrupt an existing 6.x security realm and have no choice but to restore it, the following 6.x security management tasks are described in the Compatibility Security topic of the online help for the WebLogic Server Administration Console:

- Configuring the File realm
- Configuring the Caching realm
- Configuring the LDAP V1 security realm
- Configuring the LDAP V2 security realm
- Configuring the Windows NT security realm
- Configuring the UNIX security realm
- Configuring the RDBMS security realm
- Installing a custom security realm
- Defining users
- Deleting users
- Changing the password for a user
- Unlocking a user account
- Disabling the Guest user
- Defining groups
- Deleting groups
- Defining ACLs

Warning: Compatibility security provides backward compatibility only and should not be considered a long-term security solution.

Index

D

E

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

Y