BEA WebLogic Server™

Managing WebLogic Security

Release 8.1
Document Revised: December 9, 2004

Contents

About This Document

This document explains how to configure security for WebLogic Server and how to use Compatibility security. It is organized as follows:

- Chapter 1, “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 2, “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 3, “Configuring Security Providers,” describes the attributes that must be set when configuring the security providers supplied by WebLogic Server and how to configure a custom security provider.
- Chapter 4, “Migrating Security Data,” provides information about migrating security data between security realms and security providers.
- Chapter 5, “Single Sign-On with Enterprise Information Systems,” explains how to create credential maps so EIS users can access WebLogic resources.
- Chapter 7, “Managing the Embedded LDAP Server,” describes the management tasks associated with the embedded LDAP server used by the WebLogic security providers.
- Chapter 8, “Configuring SSL,” describes how to configure SSL for WebLogic Server.

- Chapter 9, "Protecting User Accounts," describes setting attributes to protect user accounts and how to unlock a user account.
- Chapter 10, "Configuring Security for a WebLogic Domain," describes how to set security attributes on a WebLogic domain.
- Chapter 11, "Using Compatibility Security," describes how to use Compatibility security.

Audience

This document is written for Server Administrators and Application Administrators

- **Server Administrators** work closely with Application Architects to design a security scheme for the server and the applications running on the server, to identify potiential security risks, and to propose security configurations that prevent security problems. Related responsibilities may include maintaining critical production systems, setting up and configuring security realms, implementing authentication and authorization schemes for server and application resources, upgrading security features, and maintaining security provider databases. Server Administrators have in-depth knowledge of the Java security architecture, including Web application and EJB security, Public Key security, and SSL.
- **Application Administrators** work with Server Administrators to implement and maintain security configurations and authentication and authorization schemes, and to set up and maintain access to deployed application resources in defined security realms. Application Administrators have general knowledge of security concepts and the Java Security architecture. They understand Java, XML, deployment descriptors, and can identify security events in server and audit logs.

e-docs Web Site

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

How to Print the Document

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

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

PDFs, open the WebLogic Server documentation Home page, click Download Documentation, and select the document you want to print.

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

Related Information

To understand how to secure the resources in a WebLogic Server domain, 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.

Contact Us!

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

In your e-mail message, please indicate the software name and version you are using, as well as the title and document date of your documentation. If you have any questions about this version of BEA WebLogic Server, or if you have problems installing and running BEA WebLogic Server, contact BEA Customer Support through BEA WebSupport at http://www.bea.com. You can also contact Customer Support by using the contact information provided on the Customer Support Card, which is included in the product package.

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

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

Documentation Conventions

The following documentation conventions are used throughout this document.

Convention	Usage
Ctrl+Tab	Keys you press simultaneously.
italics	Emphasis and book titles.

`monospace text`	Code samples, commands and their options, Java classes, data types, directories, and file names and their extensions. Monospace text also indicates text that the user is told to enter from the keyboard.
	Examples:
	`import java.util.Enumeration;`
	`chmod u+w *`
	`config/examples/applications`
	`.java`
	`config.xml`
	`float`

`monospace italic text`	Placeholders. *Example*: `String` *`CustomerName`*`;`

`UPPERCASE MONOSPACE TEXT`	Device names, environment variables, and logical operators.
	*Example*s:
	`LPT1`
	`BEA_HOME`
	`OR`

`{ }`	A set of choices in a syntax line.

`[ ]`	Optional items in a syntax line. *Example*:
	`java utils.MulticastTest -n` *`name`* `-a` *`address`*
	     `[-p` *`portnumber`*`] [-t` *`timeout`*`] [-s` *`send`*`]`

|	Separates mutually exclusive choices in a syntax line. *Example*:
	`java weblogic.deploy [list|deploy|undeploy|update]`
	     `password {application} {source}`

Convention	Usage
. . .	Indicates one of the following in a command line: • An argument can be repeated several times in the command line. • The statement omits additional optional arguments. • You can enter additional parameters, values, or other information
. . .	Indicates the omission of items from a code example or from a syntax line.

CHAPTER 1

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.

Audience

Managing WebLogic Security is intended for Server Administrators and Application Administrators.

- **Server 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 security configurations that prevent security problems. Related responsibilities may include maintaining critical production systems, setting up and configuring security realms, implementing authentication and authorization schemes for server and application resources, upgrading security features, and maintaining security provider databases. Server Administrators have in-depth knowledge of the Java security architecture, including Web application and EJB security, Public Key security, and SSL.

- **Application Administrators** 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.

Server and Application Administrators should read *Securing WebLogic Resources* as well as this document.

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 1-7.

For information about configuring a security realm and setting it as the default security realm, see Chapter 2, "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

  The WebLogic Authentication provider supplies 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. The WebLogic 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).
- **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. The WebLogic Identity Assertion provider supplies 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. The WebLogic 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. The WebLogic Credential Mapping provider supplies this service.
- **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" on page 8-16.

For information about the functionality provided by the WebLogic security providers, see Chapter 3, "Configuring Security Providers."

For information about the default security configuration, see "The Default Security Configuration in WebLogic Server" on page 1-7.

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.

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

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).

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 `weblogic.xml` and `weblogic-ejb-jar.xml` files 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 attributes 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 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 *Introduction to 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 2, "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 "Customizing the Default Security Configuration" on page 2-1.
   - 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 2-1.

   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 2, "Customizing the Default Security Configuration."
3. Optionally, configure embedded LDAP server. By default, attributes for the embedded LDAP server are configured. However, you may want to change those attributes to optimize the use of the embedded LDAP server in your environment. For more information, see Chapter 7, "Managing the Embedded LDAP Server."
4. Ensure user accounts are properly secured. WebLogic Server provides a set of attributes 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 attributes on user accounts are set for maximum protection. If you are creating a new security realm, you need to set the user lockout attributes. For more information, see Chapter 9, "Protecting User Accounts."
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 *Managing WebLogic Security* to ensure security is completely configured for a WebLogic Server deployment.
6. Configure identity and trust and enable SSL for WebLogic Server. (This step is optional but encouraged.) For more information, see Chapter 8, "Configuring SSL."

In addition, you can:

- Map WebLogic Server credentials to credentials for an EIS (for example, the username and password for an Oracle database). See Chapter 5, "Single Sign-On with Enterprise Information Systems."

- Configure a connection filter. See Chapter 10, "Configuring Security for a WebLogic Domain."
- Enable interoperability between WebLogic domains. See Chapter 10, "Configuring Security for a WebLogic Domain."

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 11, "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 11-4.
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 11-3.

   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 3-32.

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 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 8, "Configuring SSL"
- Chapter 10, "Configuring Security for a WebLogic Domain"

CHAPTER 2

Customizing the Default Security Configuration

The following sections provide information about customizing the default security realm, creating a new security realm, and setting a security realm as the default (active) security realm.

For information about migrating security data to a new security realm, see Chapter 4, "Migrating Security Data."

Why Customize the Default Security Configuration?

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 (active) security realm and the WebLogic Adjudication, Authentication, Identity Assertion, Authorization, Credential Mapping, and Role Mapping providers are defined as the security providers. 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 an existing 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 instead of defining users and groups in the WebLogic Authentication provider.
- Add an Auditing provider to the default security realm.
- Upgrade from Compatibility security to the security providers in this release of WebLogic Server.
- Change the default attribute settings of the security providers. See Chapter 3, "Configuring Security Providers."

The easiest way to customize the default security configuration is to modify the default security realm (*myrealm*) to contain the security providers you want. For information about configuring different types of security providers in a security realm, see Chapter 3, "Configuring Security Providers."

However, you can also customize the default security configuration by creating a new security realm, configuring 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 how to create a new security realm and set that security realm as the default (active) security realm.

Creating a New Security Realm

To create a new security realm:

1. In the left pane of the WebLogic Server Administration Console, expand the Security-->Realms node.

   All security realms available for the WebLogic domain are listed in the Realms table.

2. Click the Configure a new Realm... link.
3. In the Name attribute on the General tab, enter the name of the new security realm.
4. Set the Check Roles and Security Policies attribute. The following options are available:

- 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 the Future Redeploys attribute as described in Step 6.

5. Use the Future Redeploys attribute to tell WebLogic Server how URL and EJB resources are to be secured:
   - 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.
6. You have the option of loading credential maps from `weblogic-ra.xml` deployment descriptor files into the embedded LDAP server and then using the WebLogic Server Administration Console to create new credential maps, or 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.

   To avoid overwriting new credential mapping information with old information in a `weblogic-ra.xml` deployment descriptor file, enable the Ignore Deploy Credential Mapping attribute on the new security realm.
7. 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 the Use Deprecated Web Resource attribute on the new security realm. This attribute changes the runtime behavior of the Servlet container to use a Web resource rather than a URL resource when performing authorization.

8. Click Create.
9. 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 Chapter 3, "Configuring Security Providers."

   Note: When creating a new security realm, at least one of the configured Authentication providers must return asserted LoginModules. Otherwise, run-as tags defined in deployment descriptors will not work.

10. Optionally, define Identity Assertion and Auditing providers. See Chapter 3, "Configuring Security Providers."
11. If you configured the WebLogic Authentication, Authorization, Credential Mapping or Role Mapping provider in the new security realm, verify the default attribute settings of the embedded LDAP server. See Chapter 7, "Managing the Embedded LDAP Server."
12. 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 3-27.
13. 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 *Managing WebLogic Security* to ensure security is completely configured for a WebLogic Server deployment.
14. Protect user accounts in the new security realm. See Chapter 9, "Protecting User Accounts."
15. Test the new security realm to ensure it is valid. See "Testing a New Security Realm" on page 2-4.
16. Set the new realm as the default security realm for the WebLogic domain. See "Setting a New Security Realm as the Default (Active) Security Realm" on page 2-5.
17. Reboot WebLogic Server.

Testing a New Security Realm

Configuring a new security realm is a complicated task. If you configure a security realm incorrectly, you cannot set the security realm as the default security realm. WebLogic Server can validate the configuration of a security realm to ensure it is correct.

To validate the configuration of a new security realm:

1. Configure the security realm as described in "Creating a New Security Realm" on page 2-2.
2. In the left pane of the WebLogic Server Administration Console, expand the Security-->Realms nodes.

   The Realms table shows all security realms configured for the WebLogic Server domain.
3. Select the realm you want to validate.
4. Select the Testing tab.
5. Click the Validate this realm... link.

   Any problems with the configuration of the security realm are displayed on the Testing page.

Setting a New Security Realm as the Default (Active) Security Realm

After you define attributes on the new security realm, configure the security providers for the security realm and ensure the configuration of the new security realm is valid, set the new security realm as the default (active) security realm.

To set the new security realm as the default (active) security realm:

1. In the left pane of the WebLogic Server Administration Console, expand the node representing a domain (for example, Examples).
2. Click the View Domain-wide Security Settings link.
3. Select the General tab.

   The pull-down menu forthe Default Realm attribute displays the security realms configured in the WebLogic Server domain.

   Note: If you create a new security realm but do not configure the minimun required security providers in the security realm, the realm will not be available from the pull-down menu.
4. Select the security realm you want to set as the default security realm.
5. Click Apply.

6. Reboot WebLogic Server. If you not reboot WebLogic Server, the new realm is not set as the default security realm.

To verify you set the default security realm correctly:

In the left pane of the WebLogic Server Administration Console, expand the Security-->Realms nodes. The Realms table shows all realms configured for the WebLogic Server domain. The default (active) security realm has the Default Realm attribute set to `true`.

Deleting a Security Realm

When you delete a security realm, the user, group, security role, security policy, and credential map information is not deleted from the embedded LDAP server. Use an external LDAP browser to delete any unnecessary entries from the embedded LDAP server. For more information, see "Viewing the Contents of the Embedded LDAP Server from an LDAP Browser" on page 7-5.

To delete a security realm:

1. In the left pane of the WebLogic Server Administration Console, expand the Security-->Realms nodes.

   The Realms table shows all realms configured for the WebLogic domain.

2. In the table row for the security realm you want to delete, click the trash can icon.

3. Click Yes in response to the following question:

   ```
   Are you sure you want to permanently delete OldRealm from the domain
   configuration?
   ```

   A confirmation message appears when the security realm is deleted.

Reverting to a Previous Security Configuration

It is easy to make a mistake when configuring a new security realm or security providers. A mistake may make it impossible to boot the server or correct the mistake. Reverting the `config.xml` file will reinstate the previous realm configuration.

By default, the Administration Server archives up to 5 previous versions of the `config.xml` files in the *`domain-name`*/`configArchive` directory. To revert to a previous security configuration:

1. Copy all the archived copies to a temporary directory.

2. Copy one of the archived `config.xml` files to the domain directory you are currently using. The archived files are rotated so that the newest file has a suffix with the highest number.

3. Reboot WebLogic Server.

Note this process will only revert your security realm (meaning, the realm and its providers) not users, groups, roles, or security policies. You will have to reconfigure the security realm and its providers, the user, groups, role, and security policies still exist.

Configuring Security Providers

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

- "Changing the Order of Authentication Providers" on page 3-47
- "Configuring a User Name Mapper" on page 3-48
- "Configuring a Custom User Name Mapper" on page 3-49
- "Configuring a WebLogic Authorization Provider" on page 3-50
- "Configuring a WebLogic Credential Mapping Provider" on page 3-51
- "Configuring a WebLogic Keystore Provider" on page 3-52
- "Configuring a WebLogic Role Mapping Provider" on page 3-52
- "Configuring a Custom Security Provider" on page 3-53
- "Deleting a Security Provider" on page 3-54

Note: Only the Realm Adapter Auditing, Adjudication, and Authorization providers are available when you are using Compatibility Security. For information about those providers, see Chapter 11, "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 configure attributes on a security provider:

- Before using the WebLogic Identity Assertion provider, define the active token type. See "Configuring a WebLogic Identity Assertion Provider" on page 3-33.
- 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 3-48.
- 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 3-4 or "Configuring a Custom Security Provider" on page 3-53.
- 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" on page 3-11.
- 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 3-32.

Note: There are no equivalents to the 6.x Windows NT, UNIX, RDBMS security realms in this release of WebLogic Server. Therefore, BEA recommends using the Realm Adapter Authentication provider to access the users and groups stored in these security realms.

- When creating a new security realm, configure security providers for that new realm. When using the default realm (*myrealm*), the WebLogic Adjudication, Authentication, Identity Assertion, Authorization, Credential Mapping, and Role Mapping providers are already configured. See "Creating a New Security Realm" on page 2-2.
- When adding a custom security provider to a security realm or replacing one of the WebLogic security providers with a custom security provider, configure attributes for the custom security provider. When writing a custom security provider, you can implement attributes that are configurable through the WebLogic Server Administration Console. However, those attributes 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.*

The remainder of this section describes the attributes that can be set 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. This section describes how to configure a WebLogic Adjudication provider. For information about configuring a custom security provider (including a custom Adjudication provider), see "Configuring a Custom Security Provider" on page 3-53.

To configure a WebLogic Adjudication provider:

1. In the left pane of the WebLogic Server Administration Console, expand the Security-->Realms nodes.

2. Select the name of the realm you are configuring (for example, TestRealm.)
3. Expand the Providers node.
4. Click Adjudicators.

   The Adjudicators table displays the name of the default Adjudication provider for the realm that is being configured.
5. Click the Configure a new Default Adjudicator... link.

   When working in an existing security realm, click the Replace with a new Default Adjudicator... link.
6. Optionally, on the General tab, set the Require Unanimous Permit attribute.

   The Require Unanimous Permit attribute determines how the WebLogic Adjudication provider handles a combination of `PERMIT` and `ABSTAIN` votes from the configured Authorization providers.

   – If the attribute is enabled, all Authorization providers must vote `PERMIT` in order for the Adjudication provider to vote `true`. By default, the Require Unanimous Permit attribute is enabled.

   – If the attribute is disabled, `ABSTAIN` votes are counted as `PERMIT` votes. To disable the Require Unanimous Permit attribute, click the checkbox.
7. Click Apply to save your changes.
8. Reboot WebLogic Server.

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.

Note: A Common Criteria certified configuration must have the WebLogic Auditing provider enabled. For information about the configuration of BEA WebLogic Server certified through the Common Criteria evaluation and validation process, see the product listing in the Validated Products section of the National Information Assurance Partnership Web site at http://niap.nist.gov/ . Information about the configuration of a product in evaluation is not publicly available. Such information is publicly available for validated products only.

You can use either a WebLogic Auditing provider or a custom Auditing provider in a security realm. This topic describes how to configure a WebLogic Auditing provider. For information about configuring a custom security provider (including a custom Auditing provider), see "Configuring a Custom Security Provider" on page 3-53.

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

To configure the WebLogic Auditing provider:

1. Expand the Security-->Realms nodes.
2. Select the name of the realm you are configuring (for example, TestRealm).
3. Expand the Providers node.
4. Click Auditors.

   The Auditors table displays the name of the default Auditor for the realm that is being configured.
5. Click the Configure a new Default Auditor... link.
6. Click Create.
7. Click the Details tab.
8. Choose the severity level appropriate for your WebLogic Server deployment.

   The Auditing provider audits a particular security event based on the event level specified in the Severity attribute. Auditing can be initiated when the following levels of security events occur:

   - `INFORMATION`
   - `WARNING`
   - `ERROR`
   - `SUCCESS`
   - `FAILURE`

9. Specify how many minutes to wait before creating a new `DefaultAuditRecorder.log` file in the Rotation Minutes attribute. 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.
10. Click Apply.
11. Reboot WebLogic Server.

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-Line Reference."

The WebLogic Auditing provider logs the following events:

Table 3-1 WebLogic Auditing Provider Events

Audit Event	Indicates...
AUTHENTICATE	Simple authentication (username and password) occurred.
ASSERTIDENTITY	Perimeter authentication (based on tokens) occurred.
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.
ISAUTHORIZED	An authorization attempt occurred.
ROLEEVENT	A getRoles event occurred.
ROLEDEPLOY	A deployRole event occurred.

Table 3-1 WebLogic Auditing Provider Events

Audit Event	Indicates...
ROLEUNDEPLOY	An undeployRole event occurred.
POLICYDEPLOY	A deployPolicy event occurred.
POLICYUNDEPLOY	An undeployPolicy event occurred.
AuditCreateConfigurationEvent	A user has created a resource in the domain. Requires you to enable the domain to emit 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. Requires you to enable the domain to emit 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. Requires you to enable the domain to emit configuration Audit Events. See "Configuration Auditing" in the *Administration Console Online Help*.
AuditDeleteConfigurationEvent	A user has deleted a resource in the domain. Requires you to enable the domain to emit configuration Audit Events. See "Configuration Auditing" in the *Administration Console Online Help*.

Choosing an Authentication Provider

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 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. 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 3-22.
- *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 Authentication providers, which offer different types of authentication technologies.
- Custom Identity Assertion providers, which support different types of tokens.

Note: The WebLogic Server Administration Console refers to the WebLogic Authentication provider as the Default Authenticator and the WebLogic Identity Assertion provider as the Default Identity Asserter.

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 attribute to set up login dependencies between Authentication providers and allow single-sign on between providers. See "Setting the JAAS Control Flag Attribute" on page 3-10.

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 configuraiton of the Authentication providers. See "Changing the Order of Authentication Providers" on page 3-47.

Configuring an Authentication Provider: Main Steps

To configure an Authentication provider:

1. In the left pane of the WebLogic Server Administration Console, expand the Security-->Realms nodes.
2. Select the name of the realm you are configuring (for example, TestRealm).
3. Expand the Providers-->Authentication Providers nodes.

   The Authenticators table displays the name of the default Authentication and Identity Assertion providers.
4. Choose an Authentication and/or Identity Assertion provider.
   - Configure a new iPlanet Authenticator...
   - Configure a new Realm Adapter Authenticator...
   - Configure a new Active Directory Authenticator...
   - Configure a new Default Authenticator...
   - Configure a new Default Identity Asserter...
   - Configure a new OpenLDAP Authenticator...
   - Configure a new Novell Authenticator...
5. Go to the appropriate sections to configure an Authentication and/or Identity Assertion provider.
   - "Configuring an LDAP Authentication Provider" on page 3-11

- "Configuring a WebLogic Authentication Provider" on page 3-26
- "Configuring a Realm Adapter Authentication Provider" on page 3-32
- "Configuring a WebLogic Identity Assertion Provider" on page 3-33

6. Repeat these steps to configure additional Authentication and/or Identity Assertion providers.
7. If you are configuring multiple Authentication providers, set the JAAS control flag. See "Setting the JAAS Control Flag Attribute" on page 3-10.
8. After you finish configuring Authentication and/or Identity Assertion providers, reboot WebLogic Server.

Setting the JAAS Control Flag Attribute

When you configure multiple Authentication providers, use the JAAS Control Flag attribute on the Authenticator-->General tab 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 attribute is set to OPTIONAL. If necessary, change the setting of the Control Flag so that the Authentication provider works properly in the authentication sequence.

Note: The WebLogic Server Administration Console actually sets the JAAS Control Flag to OPTIONAL when creating a security provider. MBeans for the security providers actually default to REQUIRED.

Configuring an LDAP Authentication Provider

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

For more information, see:

- "Setting LDAP Server and Caching Information" on page 3-12
- "Locating Users in the LDAP Directory" on page 3-18
- "Locating Groups in the LDAP Directory" on page 3-20
- "Locating Members of a Group in the LDAP Directory" on page 3-21
- "Accessing Other LDAP Servers" on page 3-22

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 the Group Base Distinguished Name (DN) attribute 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.

Configuring a LDAP Authentication Provider

To configure an LDAP Authentication provider:

1. In the left pane of the WebLogic Server Administration Console, expand the Security-->Realms nodes.
2. Select the name of the realm you are configuring (for example, TestRealm).
3. Expand the Providers-->Authentication Providers nodes.

   The Authenticators table displays the name of the default Authentication and Identity Assertion providers.
4. Choose an LDAP Authentication provider.
   - Configure a new iPlanet Authenticator...
   - Configure a new Active Directory Authenticator...
   - Configure a new OpenLDAP Authenticator...
   - Configure a new Novell Authenticator...
5. If you using multiple Authentication providers, define a value for the Control Flag attribute on the General tab. The Control Flag attribute determines how the LDAP Authentication provider is used with other LDAP Authentication providers. For more information, see "Configuring an LDAP Authentication Provider" on page 3-11.
6. Click Apply to save your changes.
7. Proceed to "Setting LDAP Server and Caching Information" on page 3-12.

Setting LDAP Server and Caching Information

To configure the LDAP server:

1. Select the Configuration-->LDAP tab for the LDAP Authentication provider you want to use.

   For example, select the iPlanet Configuration-->iPlanet tab.

2. Enable communication between WebLogic Server and the LDAP server by defining values for the attributes shown on the LDAP tab.

   The following table describes the attributes you set on the LDAP tab.

Table 3-2 Attributes on the LDAP Tab

Attribute	Description
Host	The host name of the computer on which the LDAP server is running.
Port	The port number on which the LDAP server is listening. If you want WebLogic Server to connect to the LDAP server using the SSL protocol, use the LDAP server's SSL port in this attribute.
SSL Enabled	Option for enabling the SSL protocol to protect communications between the LDAP server and WebLogic Server. Disable this attribute if the LDAP server is not configured to use the SSL protocol.
Principal	The Distinguished name (DN) of the LDAP user used by WebLogic Server to connect to the LDAP server. Generally, this user is the system administrator of the LDAP directory server. If you want to change passwords, this attribute must be the system administrator.
Credential	Password that authenticates the LDAP user defined in the Principal attribute.
Cache Enabled	Enables the use of a data cache with the LDAP server.
Cache Size	Maximum size of lookups in cache. The default is 32kb.
Cache TTL	Number of seconds to retain the results of an LDAP lookup.

3. To save your changes, click Apply.

4. Select the Details tab to configure additional attributes that control the behavior of the LDAP server.

   The following table describes the attributes you set on the Details tab.

Table 3-3 Attributes on the Details Tab

Attribute	Description
Use Token Groups for Group Membership Lookup	**Note:** This attribute applies to the Active Directory Authentication provider only. Indicates whether to use the Active Directory `TokenGroups` attribute lookup algorithm instead of the standard recursive group membership lookup algorithm. Active Directory `TokenGroups` attribute holds the entire flattened group membership for a user as an array of SID values. The SID values are specially indexed in the Active Directory LDAP server and yield extremely fast lookup response. By default, this attribute is not enabled. **Note:** Access to the `TokenGroups` attribute is required (meaning, the user accessing the LDAP directory must have privileges to read the `TokenGroups` attribute and the `TokenGroups` attribute must be in the schema for user objects).
Enable SID to Group Lookup Caching	**Note:** This attribute applies to the Active Directory Authentication provider only. Indicates whether or not SID to group name lookup results are cached. This attribute only applies if the Use Token Groups for Group Membership Lookup attribute is enabled.

Table 3-3 Attributes on the Details Tab

Attribute	Description
Max SID To Group Lookups In Cache	**Note:** This attribute applies to the Active Directory Authentication provider only. The maximum size of the Least Recently Used (LRU) cache for holding SID to group lookups. This attribute applies only if both the Use Token Groups for Group Membership Lookup and Enable SID to Group Lookup Caching attributes are enabled.
Group Membership Searching	Controls whether group searches are limited in depth or unlimited. This attribute controls how deeply a search should recursive into nested groups. For configurations that use only the first level of nested group hierarchy, this attribute allows improved performance during user searches by limiting the search to the first level of the group. • If a limited search is specified, the Max Group Membership Search Level attribute must be specified. • If an unlimited search is specified, the Max Group Membership Search Level attribute is ignored.

Table 3-3 Attributes on the Details Tab

Attribute	Description
Max Group Membership Search Level	Controls the depth of a group membership search if the Group Membership Searching attribute is specified. 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 attribute 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.
Use Retrieved User Name as Principal	Specifies that the user name retrieved from the LDAP directory should be added as the principal instead of the username supplied for authentication.
Follow Referrals	Specifies that a search for a user or group within the LDAP Authentication provider will follow referrals to other LDAP servers or branches within the LDAP directory. By default, this attribute is enabled.
Bind Anonymously On Referrals	By default, an LDAP Authentication provider uses the same DN and password used to connect to the LDAP server when following referrals during a search. If you want to connect as an anonymous user, enable this attribute. Contact your LDAP system administrator for more information.

Table 3-3 Attributes on the Details Tab

Attribute	Description
Results Time Limit	The maximum number of milliseconds for the LDAP server to wait for results before timing out. If this attribute is set to 0, there is no maximum time limit. The default is 0.
Connect Timeout	The maximum time in seconds to wait for the connection to the LDAP server to be established. If this attribute is set to 0, there is not a maximum time limit. The default is 0.
Parallel Connect Delay	The delay in seconds when making concurrent attempts to attempt to multiple LDAP servers. If this attribute is set to 0, connection attempts are serialized. 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. If this attribute is not set and an LDAP server is unavailable, an application may be blocked for a long time. If this attribute is greater than 0, another connection is started after the specified time.
Connection Retry Limit	The number of times WebLogic Server should try to establish a connection with an LDAP server, if the LDAP server initially throws an exception when an LDAP Authentication provider tries to connect to it. This attribute solves the problem of having two identical LDAP Authentication providers trying to simultaneously connect to the same LDAP server. In past releases, the connection would fail and WebLogic Server would need to be rebooted. Now the server will try to reconnect the specified number of times.

5. Proceed to "Locating Users in the LDAP Directory" on page 3-18.

For a more secure deployment, BEA recommends using the SSL protocol to protect communications between the LDAP server and WebLogic Server. For more information, see "Configuring SSL" on page 8-1.

Locating Users in the LDAP Directory

To specify how users are located in the LDAP directory:

1. Select the Configuration-->Users tab for the LDAP server you chose.

   For example, select the iPlanet Configuration-->Users tab.

2. Define information about how users are stored and located in the LDAP directory by defining values for the attributes shown on the Users tab.

   The following table describes the attributes you set on the Users tab.

Table 3-4 Attributes on the Users Tab

Attribute	Description
User Object Class	The LDAP object class that stores users.
User Name Attribute	The attribute on an LDAP user object that specifies the name of the user.
User Dynamic Group DN Attribute	The attribute of an LDAP user object that specifies the distinguished name of dynamic groups to which this user belongs. Dynamic groups are not supported with the Active Directory, Open LDAP, or Novell NDS directory servers, so set this attribute to NULL for these servers. If this attribute does not exist, WebLogic Server looks at the Dynamic Group Object Class attribute to determine the groups to which this user belongs. If a group contains other groups, WebLogic Server evaluates the URLs of any of the descendents of the group.

Table 3-4 Attributes on the Users Tab

Attribute	Description
User Base DN	The base DN of the tree in the LDAP directory that contains users. If you store WebLogic Server users under multiple roots in your directory, you can specify these roots as `user.dn.1`, `user.dn.2`, and so on. The LDAP Authentication provider will search under each of these roots in turn until it finds a matching user. **Note:** You must supply a user filter (`user.filter.n`) for each `user.dn.n` entry.
User Search Scope	Specifies how deep in the LDAP directory tree to search for users. Valid values are `subtree` and `onelevel`.
User from Name Filter	An LDAP search filter for finding a user given the name of the user. If a search filter is not specified (that is, if the attribute is null or empty), a default search filter is created based on the user schema. Refer to the documentation for your LDAP server for more information about writing an LDAP search filter.
All Users Filter	An LDAP search filter for finding all users beneath the base DN. If a search filter is not specified (that is, if the attribute is null or empty), a default search filter is created based on the user schema. Refer to the documentation for your LDAP server for more information about writing an LDAP search filter.

3. To save your changes, click Apply.
4. Proceed to "Locating Groups in the LDAP Directory" on page 3-20.

Locating Groups in the LDAP Directory

To define how groups are stored and located in the LDAP directory:

1. Select the Configuration-->Groups tab.

   For example, select the iPlanet Configuration-->Groups tab.

2. Define information about how groups are stored and located in the LDAP directory by defining values for the attributes shown on the Groups tab.

   The following table describes the attributes you set on the Groups tab.

Table 3-5 Attributes on the Groups Tab

Attribute	Description
Group Base DN	The base DN of the tree in the LDAP directory that stores groups.
Group Search Scope	Specifies how deep in the LDAP directory tree to search for groups. Valid values are `subtree` and `onelevel`.
Group From Name Filter	An LDAP search filter for finding a group given the name of the group. Refer to the documentation for your LDAP server for more information about writing an LDAP search filter.
All Groups Filter	An LDAP search filter for finding all groups beneath the base group DN. If the attribute is not specified (that is, if the attribute is null or empty), a default search filter is created based on the Group schema. Refer to the documentation for your LDAP server for more information about writing an LDAP search filter.
Static Group Object Class	The name of the LDAP object class that stores static groups.
Static Group Name Attribute	The attribute of a static LDAP group object that specifies the name of the group.

3. To save your changes, click Apply.
4. Proceed to "Locating Members of a Group in the LDAP Directory" on page 3-21.

Locating Members of a Group in the LDAP Directory

Note: The iPlanet Authentication provider supports dynamic groups. To use dynamic groups, set the Dynamic Group Object Class, Dynamic Group Name Attribute, and Dynamic Member URL Attribute attributes. If you use dynamic groups, consider the configuration information in "Configuring Dynamic Groups in the iPlanet Authentication Provider to Improve Performance" on page 3-30.

To define how groups members are stored and located in the LDAP directory:

1. Select on the Configuration-->Membership tab.

   For example, select the iPlanet Configuration-->Membership tab.

2. Define information about how group members are stored and located in the LDAP directory by defining values for the attributes shown on the Membership tab.

   The following table describes the attributes you set on the Membership tab.

Table 3-6 Attributes on the Membership Tab

Attribute	Definition
Static Member DN Attribute	The attribute of an LDAP group object that specifies the DNs of the members of the group.
Static Group DNs from Member DN Filter	An LDAP search filter that, given the DN of a member of a group, returns the DNs of the static LDAP groups that contain that member. If the attribute is not specified (that is, if the attribute is null or empty), a default search filter is created based on the group schema. Refer to the documentation for your LDAP server for more information about writing an LDAP search filter.

Table 3-6 Attributes on the Membership Tab

Attribute	Definition
Dynamic Group Object Class	The name of the LDAP object class that stores dynamic groups. Dynamic groups are not currently supported by Active Directory, Open LDAP, or Novell NDS directory servers. Do not set this attribute if you are using these servers.
Dynamic Group Name Attribute	The attribute of a dynamic LDAP group object that specifies the name of the group. Dynamic groups are not currently supported by Active Directory, Open LDAP, or Novell NDS directory servers. Do not set this attribute if you are using these servers.
Dynamic Member URL Attribute	The attribute of the dynamic LDAP group object that specifies the URLs of the members of the dynamic group. **Note:** A Malformed URL exception can result when this attribute contains a string value with a reserved value. To avoid this exception, avoid using `:`, `,`, `?`, and `/` in the attribute. Dynamic groups are not currently supported by Active Directory, Open LDAP, or Novell NDS directory servers. Do not set this attribute if you are using these servers.

3. To save your changes, click Apply.
4. Optionally, configure additional Authentication and/or Identity Assertion providers.
5. Reboot WebLogic Server.

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 attributes for the new LDAP server.

To create an LDAP Authentication provider for new LDAP server:

1. In the left pane of the WebLogic Server Administration Console, expand the Security-->Realms nodes.
2. Select the name of the realm you are configuring (for example, TestRealm).
3. Expand the Providers-->Authentication Providers nodes.

   The Authenticators table displays the name of the default Authentication and Identity Assertion providers.
4. Choose the LDAP Authentication provider that is most like the new LDAP server.
5. On the General tab, enter a name for the new LDAP Authentication provider in the Name attribute (for example, MyLdapServer).
6. If you using multiple Authentication providers, define a value for the Control Flag attribute on the General tab. The Control Flag attribute determines how the LDAP Authentication provider is used with other LDAP Authentication providers. For more information, see "Configuring an LDAP Authentication Provider" on page 3-11.
7. Click Create.
8. Select the LDAP Server tab (for example, iPlanet LDAP).
9. Enable communication between WebLogic Server and the new LDAP server by modifying values for the attributes shown on the LDAP tab. See "Setting LDAP Server and Caching Information" on page 3-12.
10. Select the Users tab and define how users are located in the LDAP directory. Modify the existing values to match the values for the new LDAP server. See "Locating Users in the LDAP Directory" on page 3-18.
11. Select the Groups tab and define how groups are located in the LDAP directory. Modify the existing values to match the values for the new LDAP server. See "Locating Groups in the LDAP Directory" on page 3-20.
12. Select the Membership tab and define how members of a group are located in the LDAP directory. Modify the existing values to match the values for the new LDAP server. See "Locating Members of a Group in the LDAP Directory" on page 3-21.
13. Reboot WebLogic Server.

Configuring Failover for LDAP Authentication Providers

You can configure an external LDAP provider with multiple LDAP servers and enable failover if one LDAP server is not available.

To configure failover of the LDAP servers configured for an LDAP Authentication provider, perform the following steps:

1. Select the Configuration-->LDAP tab for the LDAP Authentication provider for which you want to configure failover.

   For example, select the iPlanet Configuration-->iPlanet tab.

2. Select the LDAP tab.

3. Specify more than one LDAP server name in the Host attribute on the LDAP tab. The attribute must contain a space-delimited list of host names. Each host name may include a trailing colon and port number. For example:

   ```
   directory.knowledge.com:1050 people.catalog.com 199.254.1.2
   ```

4. Click Apply.

5. Select the Details tab.

6. Set the Parallel Connect Delay attribute.

   Specify 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 attribute is set to a value greater than 0, another connection setup thread is started after the specified number of delay seconds has passed. If the attribute is set to 0, connection attempts are serialized.

7. Set the Connection Timeout attribute.

   Specify the maximum number of seconds to wait for the connection to the LDAP server to be established. If the attribute is set to 0, there is no maximum time limit and WebLogic Server will wait until the TCP/IP layer times out to return a connection failure. This attribute may be set to a value over 60 seconds depending upon the configuration of TCP/IP.

8. Click Apply.

9. Reboot WebLogic Server.

The following examples present use scenarios that will occur when the LDAP attributes are set for LDAP failover.

Example 1

LDAP Attribute	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

In the preceeding 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 host specified in the Host attribute (`people.catalog.com`). WebLogic Server then uses `people.catalog.com` as the LDAP Server for this connection.

Example 2

LDAP Attribute	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

In the preceeding scenario, WebLogic Server attempts to connect to `directory.knowledge.com`. After 1 second, the connect attempt times out and WebLogic Server tries to connect to the next host specified in the Host attribute (`people.catalog.com`) and `directory.knowledge.com` in parallel. 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.

Configuring a WebLogic Authentication Provider

Note: The WebLogic Server Administration Console refers to the WebLogic Authentication provider as the Default Authenticator.

The WebLogic Authentication provider is case insensitive. Ensure user names are unique.

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.

To configure the WebLogic Authentication provider:

1. Configure the embedded LDAP server as described in Chapter 7, “Managing the Embedded LDAP Server.”

1. Expand the Security-->Realms nodes.

2. Select the name of the realm you are configuring (for example, TestRealm).

3. Expand the Providers-->Authentication Providers nodes.

   The Authenticators table displays the name of the default Authentication and Identity Assertion providers.

4. Choose the Configure a new Default Authenticator... link.

5. Define values for the attributes on the General tab.

- The Minimum Password Length attribute applies to the passwords you specify when defining users in the WebLogic Authentication provider.
- The Control Flag attribute determines how the WebLogic Authentication provider is used with other Authentication providers. For more information, see "Setting the JAAS Control Flag Attribute" on page 3-10.

6. Click Apply to save your changes.
7. Optionally, configure additional Authentication and/or Identity Assertion providers.
8. Reboot WebLogic Server.

Improving the Performance of WebLogic and LDAP Authentication Providers

WebLogic Server provides a set of attributes which can be used to optimize the performance of the WebLogic and LDAP Authentication providers. Performance can be improved in the following ways:

- Configure the Active Directory Authentication provider to perform group membership lookups using the `tokenGroups` attribute. The `tokenGroups` attribute holds the entire flattened group membership for a user as an array of SID values. The SID values are specially indexed in the Active Directory and yield extremely fast lookup response.
- Optimize the configuration of the group membership caches used by the WebLogic and LDAP Authentication providers.
- Configure the handling of dynamic groups in the iPlanet Authentication provider.
- 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 the Active Directory Authentication provider to use the `tokenGroups` attribute:

1. In the left pane of the WebLogic Server Administration Console, expand the Security-->Realms nodes.
2. Select the name of the realm in which the Active Directory Authentication provider is configured (for example, TestRealm).
3. Expand the Providers-->Authentication Providers nodes.

   The Authenticators table displays the name of the configured Authentication and Identity Assertion providers.
4. Select the Active Directory Authenticator from the Authenticators table.
5. Select the Details tab.
6. Set the following attributes on the Details tab:

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

  Note: Access to the `tokenGroups` attribute is required (meaning, the user accessing the LDAP directory must have privileges to read the `tokenGroups` attribute and the `tokenGroups` attribute 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 attribute only applies if the Use Token Groups for Group Membership Lookup attribute 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 attribute applies only if both the Use Token Groups for Group Membership Lookup and Enable SID to Group Lookup Caching attributes are enabled.

7. Click Apply.
8. Reboot WebLogic Server.

Optimizing the Group Membership Caches

To optimize the group membership caches for WebLogic and LDAP Authentication providers:

1. In the left pane of the WebLogic Server Administration Console, expand the Security-->Realms nodes.
2. Select the name of the realm in which the Authentication provider is configured (for example, TestRealm).
3. Expand the Providers-->Authentication Providers nodes.

   The Authenticators table displays the name of the configured Authentication and Identity Assertion providers.
4. Select the desired Authentication provider.
5. Select the Details tab.
6. Set the following attributes on the Details tab:

- Group Membership Searching—Controls whether group searches are limited or unlimited in depth. This attribute controls how deeply a search should recursive into nested groups. For configurations that use only the first level of nested group hierarchy, this attribute allows improved performance during user searches by limiting the search to the first level of the group.
  - If a limited search is specified, the Max Group Membership Search Level attribute must be specified.
  - If an unlimited search is specified, the Max Group Membership Search Level attribute is ignored.
- Max Group Membership Search Level—Controls the depth of a group membership search if the Group Membership Searching attribute is specified. 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 attribute 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 attribute only applies if the Group Membership attribute is enabled. By default, the attribute is disabled.
- Max Group Hierarchies in Cache—The maximum size of the Least Recently Used (LRU) cache that holds group membership hierarchies. This attribute only applies if the Enable Group Membership Lookup Hierarchy Caching attribute is enabled.
- Group Hierarchy Cache TTL—The number of seconds cached entries stay in the cache. The default is 60 seconds.

7. Click Apply.
8. Reboot WebLogic Server.

Configuring Dynamic Groups in the iPlanet Authentication Provider to Improve Performance

Dynamic groups do not list the names of their members. Instead, the membership of the dynamic group is constructed by matching user attributes. Since group membership needs to be computed dynamically for dynamic groups, there is a risk of performance problems for large groups. Configuring the iPlanet Authentication provider appropriately can improve performance where dynamic groups are involved.

In the iPlanet Authentication provider, the User Dynamic Group DN Attribute attribute specifies the attribute of an LDAP user object that specifies the distinguished names (DNs) of dynamic groups to which this user belongs. If such an attribute does not exist, WebLogic Server determines if a user is a member of a group by evaluating the URLs on the dynamic group. By default, User Dynamic Group DN Attribute is null. If you set User Dynamic Group DN Attribute to some other value, to improve performance set the following attributes for the iPlanet Authentication provider:

```
UserDynamicGroupDNAttribute="wlsMemberOf"

DynamicGroupNameAttribute="cn"

DynamicGroupObjectClass=""

DynamicMemberURLAttribute=""
```

To set these attributes in the Administration Console:

1. Expand Security-->Realms-->*realm name*-->Providers-->Authentication.
2. On the Users tab for your iPlanet Authentication provider, set User Dynamic Group DN Attribute.
3. On the Membership tab, set Dynamic Group Object Class and Dynamic Member URL Attribute to null (delete anything in the fields) and leave Dynamic Group Name Attribute set to `cn`.

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:

1. Expand the Security node.
2. Expand the Realms node.

   All the security realms available for the WebLogic domain are listed in the Realms table.
3. Select the desired security realm.
4. Select the General tab.
5. Set the following attributes on the General tab:

- Enable WebLogic Principal Validator Cache—Indicates whether the WebLogic Principal Validation provider uses a cache. This attribute only applies if there are Authentication providers configured in the security realm that use the WebLogic Principal Validation provider and WLSAbstractPrincipals. By default, this attribute 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 attribute only applies if the Enable WebLogic Principal Validator Cache attribute is enabled.

6. Click Apply.
7. Reboot WebLogic Server.

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. (There are no equivalents to the 6.x Windows NT, UNIX, RDBMS security realms in this release of WebLogic Server). 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 11-2.

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 which provides identity assertion 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 11-3.

When adding a Realm Adapter Authentication provider to a security realm with an Authentication provider already configured, the WebLogic Server Administration Console sets the Control Flag attribute 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 using subjects produced by the Realm Adapter Authentication provider there is no way to iterate the complete set of groups to which a user belongs.

To define attributes for the Realm Adapter Authentication provider:

1. Expand the Security-->Realms nodes.
2. Select the name of the realm you are configuring (for example, TestRealm).

3. Expand the Providers-->Authentication Providers nodes.

   The Authenticators table displays the name of the default Authentication and Identity Assertion providers.

4. Click the Configure a new Realm Adapter Authenticator... link.

5. Set the Control Flag attribute on the General tab. The Control Flag attribute determines how the Realm Adapter Authentication provider is used with other Authentication providers. See "Setting the JAAS Control Flag Attribute" on page 3-10.

6. Click Apply to save your changes.

7. Reboot WebLogic Server.

   After you reboot, the Compatibility Security tab appears and you have access to the pages in the WebLogic Server Administration Console used to manage the 6.x security realms. Note that you are not running Compatibility security.

8. Optionally, configure the Identity Assertion provider in the Realm Adapter Authentication provider to use implementations of the `weblogic.security.acl.CertAuthenticator` class with this release of WebLogic Server. The Identity Assertion provider uses X.509 tokens to perform identity assertion.

   Set the Active Type for the Identity Asserter in the Realm Adapter Authentication provider.

   a. In the Available list box, click X.509 to highlight it.

   b. Click the right arrow to move X.509 to the Chosen list box.

9. Click Apply to save your changes.

10. Reboot WebLogic Server.

11. Optionally, configure additional Authentication and/or Identity Assertion providers.

12. Reboot WebLogic Server.

Configuring a WebLogic Identity Assertion Provider

Note: The WebLogic Server Administration Console refers to the WebLogic Identity Assertion provider as the Default Identity Asserter.

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 via tokens (as opposed to simple authentication, where WebLogic Server establishes trust via 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.

You can use either a WebLogic Identity Assertion provider or a custom Identity Assertion provider in a security realm. This section describes how to configure a WebLogic Identity Assertion provider. For information about configuring a custom security provider (including a custom Identity Assertion provider), see "Configuring a Custom Security Provider" on page 3-53.

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. When using the WebLogic Identity Assertion provider, configure the active token type. The WebLogic Identity Assertion provider supports identity assertion using X509 certificates and CORBA Common Secure Interoperability version 2 (CSI v2).

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

When using the WebLogic Identity Assertion provider in a security realm, you also have the option of using 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 3-48.

To define attributes for the WebLogic Identity Assertion provider:

1. Expand the Security-->Realms nodes.
2. Select the name of the realm you are configuring (for example, TestRealm).
3. Expand the Providers-->Authentication Providers nodes.

   The Authenticators table displays the name of the default Authentication and Identity Assertion providers.
4. On the Authenticators tab, click the Configure a new Default Identity Asserter... link.

   The General tab appears.
5. Configure a user name mapper. For more information, see "Configuring a User Name Mapper" on page 3-48 or "Configuring a Custom User Name Mapper" on page 3-49.

6. In the Trusted Client Principals attribute define the list of client principals that can use CSI v2 identity assertion. You can use an asterisk (`*`) to specify all client principals. This attribute is only required if you are using CSI v2 identity assertion.
7. Set the Active Type for the WebLogic Identity Assertion provider. The list of token types supported by the WebLogic Identity Assertion provider is displayed in the Available list box. To set the Active Type:
    a. In the Available list box, select the desired token type.
    b. Click the right arrow to move the token type to the Chosen list box.
8. Click Apply to save your changes.
9. Select the Details tab.
10. Verify the setting of the Base64 Decoding Required attribute.

    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 attribute 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 attribute.
11. Click Apply.
12. Optionally, configure additional Authentication and/or Identity Assertion providers.
13. Reboot WebLogic Server.

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 attributes) 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.

Configuring an LDAP X509 Identity Assertion Provider

Note: The configuration of the functionality offered in the LDAP X509 Identity Assertion provider will change in future releases of WebLogic Server. This version of the LDAP X509 Identity Assertion provider is not upward compatible with future releases of WebLogic Server. In addition, the provider has only been tested with the Sun One LDAP server. For information about the usability of the LDAP X509 Identity Assertion provider, see the *WebLogic Server Release Notes*.

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.

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

1. An application needs to be set up to use perimeter authentication (in other words, users or system process use tokens to assert their identity). 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.

2. 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.
3. 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.

Typically, if you use the LDAP X509 Identity Assertion provider, you will 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.

Using an LDAP X509 Identity Assertion provider involves:

1. Obtaining certificates for users and putting them in an LDAP Server. 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 attributes for the certificate and the username that will be used in the Subject.
2. Configuring the LDAP X509 Identity Assertion provider to find the LDAP object for the user in the LDAP directory given the certificate's Subject DN. Basically, the user's DN in LDAP and/or the LDAP object for the user must contain attributes that match values in the certificate's Subject DN.

   Example 1: LDAP object matches Subject DN attribute.

   Certificate's Subject DN:
   `CN=`**`fred`**`, ou=Acme, c=US.`

   LDAP DN:
   `ou=people, cn=flintstone`

   LDAP Object:
   `uid=`**`fred`**`, CN=flintstone`(username), `usercert=cert`

   Example 2: LDAP DN attribute matches Subject DN component.

   Certificate's Subject DN:
   `DN: CN=`**`fred`**`, ou=Acme, c=US.`

   LDAP DN:
   `ou=people, uid=`**`fred`**

   LDAP Object:
   `SN=flintstone`(username), `uid=fred,usercert=cert`

3. Configuring 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 attribute 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 the holds the certificate.
   - A search filter that only returns LDAP objects that match a defined set of attributes. The filter narrows the LDAP search. Configure the User Filter Search attribute 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 attribute must match the values of the attributes in the certificate's Subject DN.
4. Configuring the Certificate attribute of the LDAP X509 Identity Assertion provider to specify which attribute of the LDAP object for the user holds the certificate. The LDAP object must contain an attribute the holds the certificate.
5. Configuring the Username attribute 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. Configuring the LDAP server connection for the LDAP X509 Identity Assertion provider. The LDAP server attribute information should be the same as the information defined for the LDAP Authentication provider configured in this security realm.
7. Configuring an LDAP Authentication provider for use with the LDAP X509 Identity Assertion provider. The LDAP server attribute information should be the same the information defined for the LDAP X509 Identity Assertion provider configured in Step 6.

To define attributes for the LDAP X509 Identity Assertion provider:

1. Expand the Security-->Realms nodes.
2. Select the name of the realm you are configuring (for example, TestRealm).
3. Expand the Providers-->Authentication Providers nodes.

   The Authenticators table displays the name of the default Authentication and Identity Assertion providers.
4. On the Authenticators tab, click the Configure a new LDAP X509 Identity Asserter... link.

   The General tab appears.

5. Define name and token information for the LDAP X509 Identity Assertion provider.

   The following table lists the attributes you set on the General tab.

Table 3-7 Attributes on the General Tab

Attribute	Description
Name	The name of this LDAP X509 Identity Assertion provider.
Description	A short description of this LDAP X509 Identity Assertion provider.
Version	The version number of this LDAP X509Identity Assertion provider.
Supported Types	The supported token types of this LDAP X509 Identity Assertion provider. This attribute is always set to X509.
Active Types	The token type this LDAP X509 Identity Assertion provider uses for authentication. This token type is always set to X509. Ensure no other identity assertion provider configured in the same security realm has this attribute set to X509.

6. Click Apply to save your changes.
7. Select the Details tab.
8. Define information about the attributes used to map the username in the LDAP directory to the username the certificate and the LDAP server to be used by the LDAP X509 Identity Assertion provider.

   The following table describes the attributes you set on the Details tab.

   Note: `$subj` indicates the Subject attribute in the certificate. For example: `CN=meyer.beasys.com, ou=CCE, o=BEASYS, L=SFO, C=US.`

Table 3-8 Attributes on the Details Tab

Attribute	Definition
Certificate Mapping	Specifies how to construct the base LDAP DN used to locate the LDAP object for the user. This attribute defines how to find the object from the certificate's Subject DN. Typically, this value is the same as the User Base DN attribute in the LDAP Authentication providers. You may include the fields from the Subject DN in this base DN. For example: if the Certificate subject is `CN=meyer.beasys.com, ou=fred, o=BEASYS, L=SFO, C=US` and the mapping is `ou=people, ou=$subj.ou,` WebLogic Server uses `ou=people, ou=fred, o=BEASYS, c=US` as the DN when locating the user.

Table 3-8 Attributes on the Details Tab (Continued)

Attribute	Definition
User Filter Attributes	Specifies how to select the LDAP object for the user from the LDAP objects beneath the base LDAP DN defined in the Certificate Mapping attribute. This attribute defines how to find the LDAP object from the certificate's Subject DN. The LDAP object's class must be person. This attribute contains an array of strings, each of which is an attribute that the LDAP object must match. Typically, the value of this attribute is the LDAP object that matches the value of an attribute in the certificate's Subject DN. For example: The `uid` attribute of the LDAP user object matches the Subject DN attribute, if the syntax is: `LDAPATTRNAME=$subj.SUBJECDNATTRNAME` For example: `uid=$subj.DN` This attribute is very similar to the User Name Filter attribute on LDAP Authentication providers which maps a username to a search filter. The differences are: ■ This attribute maps a certificate's Subject DN to a filter and the LDAP Authentication provider uses a single string giving the system administrator complete control over the filter. ■ The LDAP X509 Authentication provider adds `objectclass=person` to the filter and uses an array of strings that are combined.

Table 3-8 Attributes on the Details Tab (Continued)

Attribute	Definition

Certificate Attribute	Specifies the attribute on the LDAP object for the user that contains the user's certificate. This attribute defines how to find the certificate. Valid values are `userCertificate` and `userCertificate;binary`. The default is `userCertificate`.
	■ If you use the LDAP browser to load a certificate into the LDAP directory, an attribute `userCertificate` of type binary is created. To access the certificate, define the Certificate attribute as `userCertificate`.
	■ If you use `ldapmodify` to create the new attribute (for example, using the following command):
	`ldapmodify -p 1155 -D Principal -w Password`
	`dn: cn=support@bea.com, ou=Certs, dc=bea, dc=com`
	`changetype: modify`
	`add: UserCertificate`
	`userCertificate;binary::`
	`MIICxDCCAi2gAwIBAgIDIDANbgkqn...`
	An attribute `userCertificate;binary` is created when the certificate data is loaded in the LDAP directory. To access the certificate, define the Certificate attribute as `userCertificate;binary`.

Username Attribute	Specifies the attribute on the LDAP object for the user that contains the user's name. The user's name should appear in the Subject. This attribute defines how to find the user's name. Typically, this attribute matches the User Name attribute of the LDAP Authentication provider.

Table 3-8 Attributes on the Details Tab (Continued)

Attribute	Definition
Base64 Decoding Required	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 attribute.
Host	The host name of the computer on which the LDAP server is running.
Port	The port number on which the LDAP server is listening. If you want WebLogic Server to connect to the LDAP server using the SSL protocol, use the LDAP server's SSL port in this attribute.
SSL Enabled	Option for enabling the SSL protocol to protect communications between the LDAP server and WebLogic Server. Disable this attribute if the LDAP server is not configured to use the SSL protocol.
Principal	The Distinguished name (DN) of the LDAP user used by WebLogic Server to connect to the LDAP server. Generally, this user is the system administrator of the LDAP directory server. If you want to change passwords, this attribute must be the system administrator.
Credential	Password that authenticates the LDAP user defined in the Principal attribute.
Cache Enabled	Enables the use of a data cache with the LDAP server.
Cache Size	Maximum size of lookups in cache. The default is 32kb.
Cache TTL	Number of seconds to retain the results of an LDAP lookup.

Table 3-8 Attributes on the Details Tab (Continued)

Attribute	Definition
Follow Referrals	Specifies that a search for a user or group within the LDAP X509 Identity Assertion provider will follow referrals to other LDAP servers or branches within the LDAP directory. By default, this attribute is enabled.
Bind Anonymously On Referrals	By default, the LDAP X509 Identity Assertion provider uses the same DN and password used to connect to the LDAP server when following referrals during a search. If you want to connect as an anonymous user, enable this attribute. Contact your LDAP system administrator for more information.
Results Time Limit	The maximum number of milliseconds for the LDAP server to wait for results before timing out. If this attribute is set to 0, there is not maximum time limit. The default is 0.
Connect Timeout	The maximum time in seconds to wait for the connection to the LDAP server to be established. If this attribute is set to 0, there is not a maximum time limit. The default is 0.
Parallel Connect Delay	The delay in seconds when making concurrent attempts to attempt to multiple LDAP servers. If this attribute is set to 0, connection attempts are serialized. 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. If this attribute is not set and an LDAP server is unavailable, an application may be blocked for a long time. If this attribute is greater than 0, another connection is started after the specified time.

9. Click Apply.
10. Optionally, configure additional Authentication and/or Identity Assertion providers.
11. Reboot WebLogic Server.

Configuring a Single Pass Negotiate Identity Assertion Provider

Note: This version of the Single Pass Negotiate Identity Assertion provider is not upward compatible with future releases of WebLogic Server.

The Single Pass Negotiate Identity Assertion provider enables single sign-on (SSO) with Microsoft clients. The identity assertion provider decodes Simple and Protected Negotiate (SPNEGO) tokens to obtain Kerberos tokens, validates the Kerberos tokens, and maps Kerberos tokens to WebLogic users. The Single Pass Negotiate Identity Assertion provider utilizes the Java Generic Security Service (GSS) Application Programming Interface (API) to accept the GSS security context via Kerberos.

The Single Pass Negotiate Identity Assertion provider is an implementation of the Security Service Provider Interface (SSPI) as defined by the WebLogic Security Framework and provides the necessary logic to authenticate a client based on the client's SPNEGO token.

For information about using the Single Pass Negotiate Identity Assertion provider with Microsoft SSO, see [xref].

To configure a Single Pass Negotiate Identity Assertion provider:

1. Expand the Security-->Realms nodes.
2. Select the name of the realm you are configuring (for example, TestRealm).
3. Expand the Providers-->Authentication Providers nodes.

   The Authenticators table displays the name of the default Authentication and Identity Assertion providers.
4. On the Authenticators tab, click the Configure a new Single Pass Negotiate Identity Asserter... link.

   The General tab appears.
5. Define name and token information for the Single Pass Negotiate Identity Assertion provider.

   The following table lists the attributes you set on the General tab.

Table 3-9 Attributes on the General Tab

Attribute	Description
Name	The name of this Single Pass Negotiate Identity Assertion provider.
Description	A short description of this Single Pass Negotiate Identity Assertion provider.
Version	The version number of this Single Pass Negotiate Identity Assertion provider.
Active Types Chooser	The token type this Single Pass Negotiate Identity Assertion provider uses for authentication. This token type is always set to Authorization. Ensure no other identity assertion provider configured in the same security realm has this attribute set to Authorization.

6. Click Apply to save your changes.
7. Select the Details tab.
8. Ensure the Base64 Decoding Required attribute is checked.
9. Click Apply.
10. Optionally, configure additional Authentication and/or Identity Assertion providers.
11. Reboot WebLogic Server.

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 then X.509 is used for identity assertion.
- If a request comes in with a `FOO` header and a `WL-Proxy-Client-BAR` header then the `BAR` token is used for identity assertion.
- If a request comes in with a `FOO` header and a `BAR` cookie then 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.

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. Click the Re-order the Configured Authentication Providers... link to change the order of the providers. Be aware that the way each Authentication provider's Control Flag attribute is set affects the outcome of the authentication process. For more information, see "Setting the JAAS Control Flag Attribute" on page 3-10.

To change the ordering of Authentication providers:

1. Expand the Security-->Realms nodes.
2. Select the name of the realm you are configuring (for example, TestRealm).
3. Expand the Providers-->Authentication Providers nodes.
4. Click the Re-order the Configured Authentication Providers... link.
5. Select an Authentication provider from the list of configured Authentication providers to highlight it.
6. Use the arrow buttons to move the Authentication provider up or down in the list.
7. Click Apply to save your changes.
8. Reboot WebLogic Server.

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 attributes from 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`).

To use the default user name mapper:

1. Expand the Security-->Realms nodes.
2. Select the name of the realm you are configuring (for example, TestRealm).
3. Expand the Providers-->Authentication Providers nodes.
4. Select the Default Identity Assertion provider.
5. Select the Details tab.
6. Check the Use the Default User Name Mapper attribute to enable the user name mapper.
7. Specify the following attributes:
   - Default User Name Mapper Attribute Type—The attribute of 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—The attribute that ends the username. The user name mapper uses everything to the left of the attribute to create a username.
8. Click Apply.
9. Reboot WebLogic Server.

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.

To install a custom user name mapper:

1. Expand the Security-->Realms nodes.

2. Select the name of the realm you are configuring (for example, TestRealm).
3. Expand the Providers-->Authentication Providers nodes.
4. Select the Default Identity Assertion provider.
5. Select the General tab.
6. Enter the class name of the implementation of the `weblogic.security.providers.authentication.UserNameMapper` interface in the User Name Mapper Class Name attribute.
7. Click Apply.
8. Reboot WebLogic Server.

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.

You can use either a WebLogic Authorization provider or a custom Authorization provider in a security realm. This section describes how to configure a WebLogic Authorization provider. For information about configuring a custom security provider (including a custom Authorization provider), see "Configuring a Custom Security Provider" on page 3-53.

To configure a WebLogic Authorization provider:

1. Expand the Security-->Realms nodes.
2. Select the name of the realm you are configuring (for example, TestRealm).
3. Expand the Providers node.
4. Click Authorizers.

   The Authorizers table displays the name of the default Authorization provider for the realm that is being configured.
5. Click the Configure a new Default Authorizer... link.
6. Define values for the attributes on the General tab.

   The Policy Deployment Enabled attribute specifies whether or not this Authorization provider stores policy information (as opposed to retrieving policy information) for the security realm. In order to support the Policy Deployment Enabled attribute, an Authorization provider must implement the `DeployableAuthorizationProvider` Security Service Provider Interface (SSPI). By default, the WebLogic Authorization provider has this attribute enabled. The policy information is stored in the embedded LDAP server.

7. Click Apply to save your changes.
8. Reboot WebLogic Server.

Note: The WebLogic Authorization provider improves performance by caching the roles, predicates, and resource data that it looks up. For information on configuring these caches, see Best Practices: Configure Entitlements Caching When Using WebLogic Providers in *Securing WebLogic Resources*.

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 Chapter 5, "Single Sign-On with Enterprise Information Systems," and the Security topic in *Programming WebLogic J2EE Connectors*.

You can use either a WebLogic Credential Mapping provider or a custom Credential Mapping provider in a security realm. This section describes how to configure a WebLogic Credential Mapping provider. For information about configuring a custom security provider (including a custom Credential Mapping provider), see "Configuring a Custom Security Provider" on page 3-53.

To configure a WebLogic Credential Mapping provider:

1. Expand the Security-->Realms nodes.
2. Select the name of the realm you are configuring (for example, TestRealm).
3. Expand the Providers node.
4. Select Credential Mappers.
5. Click the Configure a new Default Credential Mapper... link.
6. On the General tab, set the Credential Mapping Deployment Enabled attribute.

The Credential Mapping Deployment Enabled attribute specifies whether or not this Credential Mapping provider imports credential maps from deployment descriptors (`weblogic-ra.xml` files) into the security realm. In order to support the Credential Mapping Deployment Enabled attribute, a Credential Mapping provider must implement the `DeployableCredentialProvider` SSPI. By default, the WebLogic Credential Mapping provider has this attribute enabled. 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.*

7. Click Apply to save your changes.
8. Reboot WebLogic Server.

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. For more information about using a WebLogic Keystore provider, see Chapter 8, “Configuring SSL.”

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.

You can use either a WebLogic Role Mapping provider or a custom Role Mapping provider in a security realm. This topic describes how to configure a WebLogic Role Mapping provider. For information about configuring a custom security provider (including a custom Role Mapping provider), see “Configuring a Custom Security Provider” on page 3-53.

To configure an Role Mapping provider:

1. In the left pane of the WebLogic Server Administration Console. expand the Security-->Realms nodes.
2. Select the name of the realm you are configuring (for example, TestRealm).
3. Click the Providers node.
4. Click Role Mappers.

The Role Mappers table appears. This table displays the name of the default Role Mapping provider for the realm that is being configured.

5. Click the Configure a new Default Role Mapper... link.

   The General tab appears.

6. Define values for the attributes on the General tab.

   The Role Mapping Deployment Enabled attribute specifies whether or not this Role Mapping provider imports information from deployment descriptors for Web applications and EJBs into the security realm. In order to support the Role Mapping Deployment Enabled attribute, a Role Mapping provider must implement the `DeployableRoleProvider` SSPI. By default, the WebLogic Role Mapping provider has this attribute enabled. Roles are stored in the embedded LDAP server.

   For more information, see Role Mapping Providers in *Developing Security Services for WebLogic Server.*

7. Click Apply to save your changes.

8. Reboot WebLogic Server.

Note: The WebLogic Role Mapping provider improves performance by caching the roles, predicates, and resource data that it looks up. For information on configuring these caches, see Best Practices: Configure Entitlements Caching When Using WebLogic Providers in *Securing WebLogic Resources*.

Configuring a Custom Security Provider

To configure a custom security provider:

1. Write a custom security provider. For more information, see *Developing Security Providers for WebLogic Server*.

   Put the MBean JAR file for the provider in the `WL_HOME\lib\mbeantypes` directory.

2. Start the WebLogic Server Administration Console.

3. Expand the Security-->Realms nodes.

4. Select the name of the realm you are configuring (for example, TestRealm.)

5. Expand the Providers node.

6. Expand the node for the type of provider you are configuring. For example, expand the Authenticator node to configure a custom Authentication provider.

The tab for the provider appears.

7. Click the Configure a new custom *Security_Provider_Type*... link

   where *`Security_Provider_Type`* is the name of your custom security provider. This name is read from the `DisplayName` attribute in the *`MBeanType`* tag of the MBean Definition File (MDF).

8. The General tab appears.

   The Name attribute displays the name of your custom Security provider.

9. If desired, adjust the values for the attributes for the custom Security provider.

10. Click Apply to save your changes.

11. Reboot WebLogic Server.

Note: A Common Criteria certified configuration must not include any custom security providers. For information about the configuration of BEA WebLogic Server certified through the Common Criteria evaluation and validation process, see the product listing in the Validated Products section of the National Information Assurance Partnership Web site at http://niap.nist.gov/ . Information about the configuration of a product in evaluation is not publicly available. Such information is publicly available for validated products only.

Deleting a Security Provider

To delete a security provider:

1. Expand the Security-->Realms nodes.

2. Select the name of the realm in which the provider you want to delete is configured (for example, TestRealm).

3. Expand the Providers node.

4. Select the type of provider you want to delete (for example, TestRealm-->Authorizers).

5. The table page for the provider appears (for example, the Authorizers table). The table page for the provider displays the names of all the configured providers.

6. To delete a provider, click on the trash can icon in the provider table.

7. Reboot WebLogic Server.

Note: Deleting and modifying configured security providers by using the WebLogic Server Administration Console may require manual clean up of the security provider database.

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.admin` utility.

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.

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.

Importing and Exporting Security Data from Security Realms

To export security data:

1. Expand the Security-->Realms nodes.
2. Select the name of the realm you are configuring (for example, TestRealm).
3. Select the Migration-->Export tabs.
4. Specify the directory and filename in which to export the security data in the Export Directory on Server attribute. The directory must exist; the Migration utility will not create a new directory.

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.

5. Click Export.

To import security data:

Note: Once the data is exported from the security realm, it can be imported at any time.

1. Expand the Realms node.
2. Select the name of the security realm in which the security data is to be imported.
3. Select the Migration-->Import tabs.
4. Specify the directory location and file name of the file that contains the exported security data in the Import Directory on Server attribute.
5. Click Import.

To verify the security data was imported correctly:

1. Expand the Security-->Realms nodes.
2. Select the name of the realm into which the security data was imported.
3. Click Users.
4. Users from the security realm from which you exported the security data should appear in the Users table.

Note: Arguments for security policies and roles are stored in lowercase on the Windows platform and mixed case on UNIX platforms. If security policies and security roles are defined in a domain on one platform, exported, and then imported into a domain on a platform with different case handling, the security roles and policies may not work properly. Specifically, Web Application pages that are protected on one operating system may not be protected on a different operating system.

Importing and Exporting Security Data from Security Providers

Provider-specific security data can also be exported and imported between providers in different security realms. 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 4-1 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.

To export security data from a security provider:

1. Expand the Security-->Realms nodes.
2. Select the name of the realm you are configuring (for example, TestRealm).
3. Select the type of provider from which you want to export security data (for example, Authentication Providers).
4. Select the security provider from which you want to export security data.
5. Select the Migration-->Export tabs.

6. Specify the directory and filename in which to export the security data in the Export Directory attribute. The directory must exist; the Import/Export feature will not create a new directory.

   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.

7. Optionally, define a set of security data to be exported in the Export Constraints box.
8. Click Export.

To import security data into a security provider:

Note: Once the data is exported from the security provider, it can be imported at any time.

1. Expand the Realms node.
2. Select the name of the security realm in which the security data is to be imported.
3. Expand the Providers node.
4. Select the security provider in which the security data is to be imported.
5. Select the Migration-->Import tab.
6. Specify the directory location and file name of the file that contains the exported security data in the Import Directory on Server attribute or use the Browse button to locate the exported file on your computer.
7. Click Import.

Note: Arguments for security policies and roles are stored in lowercase on the Windows platform and mixed case on UNIX platforms. If security policies and security roles are defined in a domain on one platform, exported, and then imported into a domain on a platform with different case handling, the security roles and policies may not work properly. Specifically, Web Application pages that are protected on one operating system may not be protected on a different operating system.

Using the weblogic.Admin Utility

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 ""
```

Single Sign-On with Enterprise Information Systems

This section explains how to create credential maps that allow Enterprise Information System (EIS) users to access protected WebLogic Resources.

- "Overview" on page 5-1
- "Using Deployment Descriptors to Create Credential Maps (Deprecated)" on page 5-2
- "Using the WebLogic Administration Console to Create Credential Maps" on page 5-4

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.

Overview

Single sign-on allows user information to be propagated from an EIS to WebLogic Server so that users are not required to authenticate themselves multiple times as they access WebLogic Server resources. Resource adapters defined by the J2EE Connector Architecture can acquire the credentials necessary to authenticate users defined in an EIS when they request access to a protected WebLogic resource. The container in WebLogic Server that hosts resource adapters can retrieve the appropriate set of credentials for the WebLogic resource using a credential map. A credential map creates an association between a user in WebLogic Server security realm and an identity (a username and password combination) used to authenticate that user in an EIS such as an Oracle database, a SQL server, or a SAP application.

Creating a credential map is a two-step process:

1. Create a WebLogic Server user or group for the EIS user. The user or group needs to be defined in the configured Authentication provider. Multiple WebLogic Server users or groups can be mapped to the same remote user or group. For more efficient management, BEA recommends using groups to create credential maps.
2. Create a credential map for the EIS users. Use the username and password under which the user is authenticated to the EIS or the group in which the EIS user is a member to define the user. These credential maps are stored in the embedded LDAP server.

For more information using security in resource adapters, see the Security topic in *Programming WebLogic J2EE Connectors.*

WebLogic Server provides two techniques for creating credential maps: deployment descriptors (deprecated) and the WebLogic Server Administration Console. The following sections describe both techniques.

Using Deployment Descriptors to Create Credential Maps (Deprecated)

Credentials maps can be specified in the `<security-principal-map>` element of the `weblogic-ra.xml` deployment descriptor file. The `<security-principal-map>` element provides the association between the credentials used to log in to the EIS and credentials used to authenticate to WebLogic resources. The deployment descriptor technique for creating credential maps is deprecated in this release of WebLogic Server. Instead, use the WebLogic Server Administration Console to create credential maps. For more information, see "Using the WebLogic Administration Console to Create Credential Maps" on page 5-4.

If you deployed a resource adapter that has a `weblogic-ra.xml` deployment descriptor file containing a defined `<security-principal-map>` element, BEA recommends importing the data into the embedded LDAP server and where it can be used by the WebLogic Credential Mapping provider.

Importing Information from weblogic-ra.xml into the Embedded LDAP Server

To import the information from the `weblogic-ra.xml` deployment descriptor file into the embedded LDAP server, enable the Credential Mapping Deployment Enabled attribute on the Credential Mapping provider in the default (active) security realm. When the resource adapter is deployed, the credential map information is loaded into the Credential Mapping provider.

In order to support the Credential Mapping Deployment Enabled attribute, a Credential Mapping provider must implement the DeployableCredentialProvider SSPI. By default, the WebLogic Credential Mapping provider has this attribute enabled. Therefore, information from a `weblogic-ra.xml` deployment descriptor file is automatically loaded into the WebLogic Credential Mapping provider when the resource adapter is deployed.

It is important to understand that 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 credential mapping information may be lost.

Avoiding Overwriting of Credential Mapping Information

To avoid overwriting new credential mapping information with old information in a `weblogic-ra.xml` deployment descriptor file, enable the Ignore Security Data in Deployment Descriptors attribute:

1. Expand the Security-->Realms nodes.

   All security realms available for the WebLogic domain are listed in the Realms table.

2. Select the name of the realm you are using.
3. Select the General tab.
4. Check the Ignore Deploy Credential Mapping attribute. This attribute specifies that the Credential Mapping providers in the security realm will use only credential maps created using the WebLogic Server Administration Console. By default, this attribute is not checked meaning the Credential Mapping provider will load credential maps specified in a `weblogic-ra.xml` deployment descriptor file.
5. Click Apply.
6. Reboot WebLogic Server.

After performing the preceeding procedure, BEA Systems recommends modifying the `weblogic-ra.xml` deployment descriptor file to remove the `<security-principal-map>` element.

Using the WebLogic Administration Console to Create Credential Maps

You can now use the WebLogic Server Administration Console to create credential maps. If you are using the WebLogic Credential Mapping provider, the credential maps are stored in the embedded LDAP server.

To create a credential map:

1. Verify the Ignore Deploy Credential Mapping attribute is enabled on the default (active) security realm. Otherwise, you risk overwriting credential maps with old information in `weblogic-ra.xml` deployment descriptor files. For more information, see "Setting a New Security Realm as the Default (Active) Security Realm" on page 2-5.
2. Define a user or group for the EIS user. For more information, see Users and Groups in *Securing WebLogic Resources*.
3. Deploy a resource adapter. For more information, see *Programming WebLogic J2EE Connectors*.
4. In the left pane of the WebLogic Server Administration Console, expand Deployments-->Connector Modules nodes.
5. Right-click the name of the Connector for which you want to create a credential map, and select Define Credential Mappings... to display the Credential Mappings page.

   If available, a table of currently defined credential maps appears in the right pane.
6. Click the Configure a New Credential Mapping... link.

   If multiple WebLogic Credential Mapping providers are configured in the security realm, select which WebLogic Credential Mapping provider's database should store information for the new credential map.
7. Enter the WebLogic Server user or group name you defined for the EIS user in step 2 in the WLS User field.
8. Enter the name of the EIS user in the Remote User field.
9. Click Apply to save your changes.

   The EIS user appears in the Credential Maps table.
10. Click the name of the EIS user in the Remote User column of the Credential Maps table.
11. Enter the password for the EIS user in the Remote Password field.

12. Confirm the password.
13. Click Apply.

Configuring Single Sign-On with Microsoft Clients

This section explains how to set up single sign-on (SSO) with Microsoft clients, using Windows authentication based on the Simple and Protected Negotiate (SPNEGO) mechanism and the Kerberos protocol, together with the WebLogic Negotiate Identity Assertion provider.

Single Sign-on with Microsoft Clients: Main Steps

Single sign-on (SSO) with Microsoft clients allows cross-platform authentication between Web applications or Web Services running in a WebLogic Server domain and .NET Web Service clients or browser clients (for example, Internet Explorer) in a Microsoft domain. The Microsoft clients must use Windows authentication based on the Simple and Protected Negotiate (SPNEGO) mechanism.

Cross-platform authentication is achieved by emulating the negotiate behavior of native Windows-to-Windows authentication services that use the Kerberos protocol. In order for cross-platform authentication to work, non-Windows servers (in this case, WebLogic Server) need to parse SPNEGO tokens in order to extract Kerberos tokens which are then used for authentication.

Configuring SSO with Microsoft clients involves set up procedures in the Microsoft Active Directory, the client, and the WebLogic Server domains.

- The Kerberos protocol uses the Active Directory server in the Microsoft domain to store the necessary security information. Therefore, you need to define a principal in Active Directory to represent the WebLogic Server.
- Any Microsoft client you want to access in the Microsoft domain must be set up to use Windows Integrated authentication, sending a Kerberos ticket when available.
- In the WebLogic Server domain, a Negotiate Identity Assertion provider needs to be configured in the security realm. The Web application or Web Service used in SSO needs to have authentication set in a specific manner. A JAAS login file that defines the location of the Kerberos identification for WebLogic Server must be created.

The main configuration steps are as follows:

1. Configure your network domain to use Kerberos. See

1. Create a Kerberos identification for WebLogic Server.

   a. Create a user account in Active Directory for the host on which WebLogic Server is running.

   b. Create a Service Principal Name for this account.

   c. Create a user mapping and keytab file for this account.

   For more information, see "Creating a Kerberos Identification for WebLogic Server" on page 6-5.

2. Choose a Microsoft client (either a Web Service or a browser) and configure it to use Windows Integrated authentication. For more information, see "Configuring Microsoft Clients to Use Windows Integrated Authentication" on page 6-7.

3. Set up the WebLogic Server domain to use Kerberos authentication.

   a. Create a JAAS login file that points to the Active Directory server in the Microsoft domain and the keytab file created in Step 1. For more information, see "Creating a JAAS Login File" on page 6-9.

   b. Configure a Negotiate Identity Assertion provider in the WebLogic Server security realm. For more information, see "Configuring a Single Pass Negotiate Identity Assertion Provider" on page 3-45.

   c. Enable the WebLogic Server Web application or Web Service to use the `CLIENT-CERT` authentication mechanism. For more information, see "Enabling the WebLogic Server Web Application or Web Service to Use the Negotiate Token" on page 6-10.

4. Start WebLogic Server using specific start-up arguments. For more information, see "Startup Arguments for Using Kerberos Authentication with WebLogic Server" on page 6-12

The following sections describe these steps in detail.

System Requirements for SSO with Microsoft Clients

To use SSO with Microsoft clients, you must comply with these system requirements:

A host computer with:

- Windows 2000 or greater installed
- Fully-configured Active Directory authentication service. Specific Active Directory requirements include:
  - User accounts for mapping Kerberos services
  - Service Principal Names (SPNs) for those accounts
  - Key tab files created and copied to the start-up directory in the WebLogic Server domain
- WebLogic Server installed and configured properly to authenticate via Kerberos, as described in this section, Chapter 6, "Configuring Single Sign-On with Microsoft Clients."

Client systems with:

- Windows 2000 Professional SP2 or greater installed
- One of the following types of clients:
  - A properly configured Internet Explorer browser. Internet Explorer 6.01 or greater is supported.

– .NET Framework 1.1 and a properly configured Web Service client.

Clients must be logged on to a Windows 2000 domain and have Kerberos credentials acquired from the Active Directory server in the domain. Local logons will not work.

Configuring your Network Domain to Use Kerberos

A Windows domain controller can server as the Kerberos Key Distribution Center (KDC), using the Active Directory and the Kerberos services. On any domain controller, the Active Directory and the Kerberos services will be running automatically. To configure Kerberos in your Windows domain controller, you need to configure each machine that will access the KDC to locate the Kerberos realm and available KDC servers. For Windows machines, this configuration is in the `krb5.ini` file, in the `C:\winnt` folder. For UNIX machines, this configuration is in the `krb5.conf` file, the default location of which is `/etc/krb5/`. For example:

Listing 6-1 Sample krb5.ini File

```
[libdefaults]
default_realm = MYDOM.COM (Identifies the default realm. Set its value to
your Kerberos realm)
default_tkt_enctypes = des-cbc-crc
default_tgs_enctypes = des-cbc-crc
ticket_lifetime = 600

[realms]

MYDOM.COM = {
kdc = <IP address for MachineA> (host running the KDC)
(For Unix systems, you need to specify port88, as in <IP-address>:88)
admin_server = MachineA
default_domain = MYDOM.COM

[domain_realm]
.mydom.com = MYDOM.COM


[appdefaults]
autologin = true
forward = true
forwardable = true
encrypt = true
```

Creating a Kerberos Identification for WebLogic Server

Active Directory provides support for service principal names (SPN) which are a key component in Kerberos authentication. SPNs are unique identifiers for services running on servers. Every service that uses Kerberos authentication needs to have an SPN set for it so that clients can identify the service on the network. An SPN usually looks something like name@YOUR.REALM. You need to define an SPN to represent your WebLogic Server in the Kerberos realm. If an SPN is not set for a service, clients have no way of locating that service. Without correctly set SPNs, Kerberos authentication is not possible. Keytab files are the mechanism for storing the SPNs. Keytab files are copied to the WebLogic Server domain and are used in the login process. This configuration step describes how to create an SPN, user mapping, and keytab file for WebLogic Server.

This configuration step requires the use of the following Active Directory utilities:

- `setspn`–Windows 2000 Resource Kit
- `ktpass`–Windows 2000 distribution CD in `Program Files\Support Tools`

Note: The `setspn` and `ktpass` Active Directory utilities are products of Microsoft. Therefore, BEA Systems does not provide complete documentation for this utilities. For more information, see the appropriate Microsoft documentation.

To create a Kerberos identification for WebLogic Server:

1. Create a user account for the host computer on which WebLogic Server runs in the Active Directory server. (Select New > User, not New > Machine.)

   When creating the user account, use the simple name of the computer. For example, if the host is named `myhost.example.com`, create a user in Active Directory called `myhost.`

   Note the password you defined when creating the user account. You will need it in step 3. Do not select the `User must change password at next logon` option, or any other password options.

2. Configure the new user account to comply with the Kerberos protocol. The user account's encryption type must be DES and the account must require Kerberos pre-authentication.

   a. Right-click the name of the user account in the Users tree in the left pane and select Properties.

   b. Select the Account tab and check the box "Use DES encryption types for this account." Make sure no other boxes are checked, particularly the box "Do not require Kerberos pre-authentication."

   c. Setting the encryption type may corrupt the password. Therefore, you should reset the user password by right-clicking the name of the user account, selecting Reset Password, and re-entering the same password specified earlier.

3. Use the `setspn` utility to create the Service Principal Names (SPNs) for the user account created in step 1. Enter the following commands:

   ```
   setspn -a host/myhost.example.com myhost
   ```

   ```
   setspn -a HTTP/myhost.example.com myhost
   ```

4. Check which SPNs are associated with your user account, using the following command:

   ```
   setspn -L account name
   ```

   This is an important step. If the same service is linked to a different account in the Active Directory server, the client will not send a Kerberos ticket to the server.

5. Create a user mapping using the `ktpass` utility:

   Windows

   ```
   ktpass -princ host/myhost@Example.CORP -pass password -mapuser myhost
   -out c:\temp\myhost.host.keytab
   ```

6. Create a keytab file. On Windows, the `ktab` utility manages principal name and key pairs in the key table and allows you to list, add, update, or delete principal names and key pairs. On UNIX, it is preferable to use the `ktpass` utility.

   Windows

   a. Run the `ktab` utility on the host on which WebLogic Server is running to create the keytab file:

      ```
      ktab -k keytab-filename -a myhost@Example.CORP
      ```

   b. Copy the keytab file to the startup directory in the WebLogic Server domain.

   UNIX

   a. Create a user mapping using the `ktpass` utility, using a command like this, where *`password`* is the password for the user account created in step 1:

      ```
      ktpass -princ HTTP/myhost@Example.CORP -pass password -mapuser myhost
      -out c:\temp\myhost.HTTP.keytab
      ```

   b. Copy the keytab file created in Step a to the startup directory in the WebLogic Server domain.

c. Login as root and then merge them into a single keytab using the `ktutil` utility as follows:

```
ktutil: "rkt myhost.host.keytab"
ktutil: "rkt myhost.HTTP.keytab"
ktutil: "wkt mykeytab"
ktutil: "q"
```

7. Run the `kinit` utility to verify Kerberos authentication is working properly.

```
kinit -k -t keytab-file account-name
```

The output should be something similar to:

```
New ticket is stored in cache file C:\Documents and
Settings\Username\krb5cc_MachineB
```

Configuring Microsoft Clients to Use Windows Integrated Authentication

Ensure the Microsoft client you want to use for single sign-on is configured to use Windows Integrated authentication. The following sections describe how to configure a .NET Web server and an Internet Explorer browser to use Windows Integrated authentication.

Configuring a .NET Web Service

To configure a .NET Web Service to use Windows authentication, perform the following steps:

1. In the `web.config` file for the Web Service, set the authentication mode to Windows for IIS and ASP.NET as follows:

```
<authentication mode="Windows" />
```

This setting is usually the default.

2. Add the statement needed for the Web Services client to pass to the proxy Web Service object so that the credentials are sent through SOAP.

For example, if you have a Web Service client for a Web Service that is represented by the proxy object `conv`, the syntax is as follows:

```
/*
* Explicitly pass credentials to the Web Service
*/
conv.Credentials =
System.Net.CredentialCache.DefaultCredentials;
```

Configuring an Internet Explorer Browser

To configure an Internet Explorer browser to use Windows authentication, follow these procedures in Internet Explorer:

Configure Local Intranet Domains

1. In Internet Explorer, select Tools > Internet Options.
2. Select the Security tab.
3. Select Local intranet and click Sites.
4. In the Local intranet popup, ensure that the "Include all sites that bypass the proxy server" and "Include all local (intranet) sites not listed in other zones" options are checked.
5. Click Advanced.
6. In the Local intranet (Advanced) dialog box, add all relative domain names that will be used for WebLogic Server instances participating in the SSO configuration (for example, `myhost.example.com`) and click OK.

Configure Intranet Authentication

1. Select Tools > Internet Options.
2. Select the Security tab.
3. Select Local intranet and click Custom Level....
4. In the Security Settings dialog box, scroll to the User Authentication section.
5. Select Automatic logon only in Intranet zone. This option prevents users from having to re-enter logon credentials, which is a key piece to this solution.
6. Click OK.

Verify the Proxy Settings

If you have a proxy server enabled:

1. Select Tools > Internet Options.
2. Select the Connections tab and click LAN Settings.
3. Verify that the proxy server address and port number are correct.

4. Click Advanced.
5. In the Proxy Settings dialog box, ensure that all desired domain names are entered in the Exceptions field.
6. Click OK to close the Proxy Settings dialog box.

Set Integrated Authentication for Internet Explorer 6.0

In addition to the previous settings, one additional setting is required if you are running Internet Explorer 6.0.

1. In Internet Explorer, select Tools > Internet Options.
2. Select the Advanced tab.
3. Scroll to the Security section.
4. Make sure that Enable Integrated Windows Authentication option is checked and click OK.
5. If this option was not checked, restart the computer.

Creating a JAAS Login File

If you are running WebLogic Server on either the Windows or UNIX platforms, you need a JAAS login file. The JAAS login file tells the WebLogic security framework to use Kerberos authentication and defines the location of the keytab file which contains Kerberos identification information for WebLogic Server. You specify the location of this file in the `java.security.auth.login.config` startup argument for WebLogic Server, as described in "Startup Arguments for Using Kerberos Authentication with WebLogic Server" on page 6-12.

Listing 6-2 contains a sample JAAS login file for Kerberos authentication.

Listing 6-2 Sample JAAS Login File for Kerberos Authentication

```
com.sun.security.jgss.initiate {

     com.sun.security.auth.module.Krb5LoginModule required
     principal="myhost@Example.CORP" useKeyTab=true
     keyTab=mykeytab storeKey=true;
};
```

```
com.sun.security.jgss.accept {

     com.sun.security.auth.module.Krb5LoginModule required
     principal="myhost@Example.CORP" useKeyTab=true
     keyTab=mykeytab storeKey=true;

};
```

Configuring the Identity Asssertion Provider

WebLogic Server includes a security provider, the Negotiate Identity Assertion provider, to support single sign-on (SSO) with Microsoft clients. This identity assertion provider decodes Simple and Protected Negotiate (SPNEGO) tokens to obtain Kerberos tokens, validates the Kerberos tokens, and maps Kerberos tokens to WebLogic users. You need to configure a Negotiate Identity Assertion provider in your WebLogic security realm in order to enable SSO with Microsoft clients. For information, see "Configuring a Single Pass Negotiate Identity Assertion Provider" on page 3-45.

Enabling the WebLogic Server Web Application or Web Service to Use the Negotiate Token

To enable a WebLogic Server Web Service or Web application to participate in SSO with Microsoft clients you must define a security constraint for the WebLogic Server resources users will access using SSO. Do this by configuring the Web Service or Web application to use `CLIENT-CERT` as the authentication mechanism. This configuration is done by editing the `web.xml` deployment descriptor and declaring `CLIENT-CERT` in the `<auth-method>` element.

Listing 6-3 shows a sample `web.xml` file for a WebLogic Web Service resource named `Conversation` with the authentication mechanism set to `CLIENT-CERT`.

Listing 6-3 Sample web.xml file that uses Client-Cert Authentication

```
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application
2.3//EN" "http://java.sun.com/dtd/web-app_2_3.dtd">

<web-app>
```

```
<security-constraint>
       <display-name>Security Constraint on Conversation
       </display-name>

       <web-resource-collection>
              <web-resource-name>Conversation web service
       </web-resource-name>

       <description>Only those granted the ConversationUsers
       role may access the Conversation web service.
       </description>
              <url-pattern>/async/Conversation.jws/*</url-pattern>
              <http-method>GET</http-method>
              <http-method>POST</http-method>
       </web-resource-collection>

       <auth-constraint>
              <role-name>ConversationUsers</role-name>
       </auth-constraint>
</security-constraint>

       <login-config>
              <auth-method>CLIENT-CERT</auth-method>
       </login-config>
       <security-role>
              <description>Role description</description>
              <role-name>ConversationUsers</role-name>
       </security-role>
</web-app>
```

For more information about adding security to Web applications, see Securing Web Applications in *Programming WebLogic Security*.

For more information about adding security to Web Services, see Configuring Security in *Programming WebLogic Web Services*.

Startup Arguments for Using Kerberos Authentication with WebLogic Server

To use Kerberos authentication with WebLogic Server, use the following start-up arguments when you start WebLogic Server:

```
-Djava.security.krb5.realm=Example.CORP
-Djava.security.krb5.kdc=ADhostname
-Djava.security.auth.login.config=krb5Login.conf
-Djavax.security.auth.useSubjectCredsOnly=false
-Dweblogic.security.enableNegotiate=true
```

where

- `java.security.krb5.realm` defines the Microsoft domain in which the Active Directory server runs.
- `java.security.krb5.kdc` defines the host name on which the Active Directory server runs.
- `java.security.auth.login.config` defines the location of the Kerberos login information.
- `javax.security.auth.useSubjectCredsOnly` specifies that it is permissible to use an authentication mechanism other than Subject credentials.

`weblogic.security.enableNegotiate` enables the Servlet container in WebLogic Server to support the Negotiate token used by SPNEGO.

Verifying that SSO with Microsoft Clients Works

To verify that SSO with Microsoft clients is properly configured, point a browser (that you have configured as described in "Configuring an Internet Explorer Browser" on page 6-8) to the Microsoft Web application or Web Service you want to use. If you are logged on to a Windows domain and have Kerberos credentials acquired from the Active Directory server in the domain, you should be able to access the Web application or Web Service without providing a username or password.

CHAPTER 7

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" on page 7-1
- "Embedded LDAP Server Replication" on page 7-4
- "Configuring Backups for the Embedded LDAP Server" on page 7-4
- "Viewing the Contents of the Embedded LDAP Server from an LDAP Browser" on page 7-5
- "Exporting and Importing Information in the Embedded LDAP Server" on page 7-6
- "LDAP Access Control Syntax" on page 7-7

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 attribute. 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.

To configure the embedded LDAP server:

1. Expand the Domain node (for example, Examples).
2. Select the Security-->Embedded LDAP tab.
3. Set attributes on the Embedded LDAP tab. The following table describes each attribute on the Embedded LDAP tab.

Table 7-1 Attributes on the Embedded LDAP Tab

LDAP Attribute	Description
Credential	The credential (usually a password) used to connect to the embedded LDAP server. If this password has not been set, WebLogic Server generates a password at startup, initializes the attribute, and saves the configuration to the `config.xml` file. If you want to connect to the embedded LDAP server using an external LDAP browser and the embedded LDAP administrator account (`cn=Admin`), change this attribute from the generated value.
Backup Hour	The hour at which to back up the embedded LDAP server data files. This attribute is used in conjunction with the Backup Minute attribute to determine the time at which the embedded LDAP server data files are backed up. At the specified time, WebLogic Server suspends writes to the embedded LDAP server, backs up the data files into a zip file in the `ldap/backup` directory, and then resumes writes. The default is 23.
Backup Minute	The minute at which to back up the embedded LDAP server data files. This attribute is used in conjunction with the Back Up Hour attribute to determine the time at which the embedded LDAP server data files are backed up. The default is 5 minutes.

Table 7-1 Attributes on the Embedded LDAP Tab

LDAP Attribute	Description
Backup Copies	The number of backup copies of the embedded LDAP server data files. This value limits the number of zip files in the `ldap/backup` directory. The default is 7.
Cache Enabled	Specifies whether or not a cache is used with the embedded LDAP server. This cache is used when a Managed Server is reading or writing to the master embedded LDAP server that is running on the Administration Server.
Cache Size	The size of the cache (in K) that is used with the embedded LDAP server. The default is 32K.
Cache TTL	The time-to-live (TTL) of the cache in seconds. The default is 60 seconds.
Refresh Replica At Startup	Specifies whether or not a Managed Server should refresh all replicated data at boot time. This attribute is useful if you have made a large number of changes while the Managed Server was not active and you want to download the entire replica instead of having the Administration Server push each change to the Managed Server. Use this attribute to propagate a new system password to the Administration and Managed Servers in a domain. The default is false.

Table 7-1 Attributes on the Embedded LDAP Tab

LDAP Attribute	Description
Master First	Specifies that connections to the master LDAP server (running on the Administration Server) should always be made instead of connections to the local replicated embedded LDAP server. This causes the Managed Server to retrieve security data from the embedded LDAP server in the Administration Server instead of going to the local embedded LDAP server that contains a replica of the information in the Administration Server.
Anonymous Bind Allowed	Specifies whether to permit anonyomous access to the embedded LDAP server. By default, anonymous access is disabled, since it might constitute a security vulnerability.

4. Click Apply to save your changes.
5. Reboot WebLogic Server.

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.

Configuring Backups for the Embedded LDAP Server

To configure the backups of the embedded LDAP server:

1. Expand the Domain node (for example, Examples).
2. Select the Security--> Embedded LDAP tab.
3. Set the Backup Hour, Backup Minute, and Backup Copies attributes on the Embedded LDAP tab.
4. Click Apply to save your changes.
5. Reboot WebLogic Server.

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. Download 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 tab.
   c. Change the Credential attribute from its generated value to a password you will use in the LDAP browser. See Table 7-1.
   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 2c.

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 tab 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 7-2 summarizes where data is stored in the hierarchy of the embedded LDAP server.

Table 7-2 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 and to block all anonymous access. The WebLogic security providers use only 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 7-1 shows a sample access control file.

Listing 7-1 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 7-3 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.
`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 7-4 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.
`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.

Table 7-4 Entry Permissions

Permission	Description
`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.
`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.

Configuring SSL

This 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.

Private Keys, Digital Certificates, and Trusted Certificate Authorities

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

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.

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

For more information on these steps, see:

- "Obtaining Private Keys, Digital Certificates, and Trusted Certificate Authorities" on page 8-4
- "Storing Private Keys, Digital Certificates, and Trusted Certificate Authorities" on page 8-12
- "Configuring Keystores" on page 8-16
- "Configuring SSL" on page 8-18
- "Configuring Two-Way SSL" on page 8-20

Obtaining Private Keys, Digital Certificates, and Trusted Certificate Authorities

To use SSL, the server needs 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 `keytool` 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" on page 8-6.

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 "Common Keytool Commands" on page 8-14.

When using the deprecated file-based private keys, digital certificates, and trusted CA, 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 8-10.

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.

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 provides command line options to specify a CA certificate and key to be used for issuing generated certificates. It will default to using the `CertGenCA.der` and `CertGenCAKey.der` files which contain the digital certificate and private key of the demonstration CA certificate. The digital certificates generated by the CertGen utility have the host name of the machine on which they were generated as the value for its common name field (`cn`) by default only. Command line options let you specify values for the `cn` and other Subject domain name (DN) fields, such as `orgunit`, `organization`, `locality`, `state`, `countrycode`.

The CertGen utility generates public certificate and private key files in PEM and DER formats. On Windows, double-click `.der` files to view the details of the generated digital certificate. The `.pem` files can be used when you boot WebLogic Server or use the digital certificates with a client.

Note: By default, the CertGen utility looks for the `CertGenCA.der` and `CertGenCAKey.der` files in the current directory, or in the *`WL_HOME`*`/server/lib` directory, as specified in the `weblogic.home` system property or the `CLASSPATH`.

If you want to use the default settings you need not specify CA files on the command line. Alternatively, you can specify CA files on the command line, as shown in the following command syntax.

1. To generate a certificate, enter the following command at a command prompt:

```
$ java utils.CertGen

[-cacert <ca_cert_filename>] [-cakey <ca_key_filename>]
[-cakeypass <ca_key_password>] [-selfsigned]
[-certfile <certfile>] [-keyfile <privatekeyfile>]
[-keyfilepass <keyfilepassword>] [-strength <keystrength>]
[-cn <commonname>] [-ou <orgunit>] [-o <organization>]
[-l <locality>] [-s <state>] [-c <countrycode>]
[-subjectkeyid <subjectkeyidentifier>]
[-subjectkeyidformat UTF-8|BASE64]
```

Argument	Definition
`ca_cert_filename`	The file name of the issuer's CA public certificate.
`ca_key_filename`	The file name of the issuer's CA private key.
`ca_key_password`	The password for the issuer's CA private key.
`selfsigned`	Generates a self-signed certificate that can be used as a trusted CA certificate. If this argument is specified, the *`ca_cert_filename`*, *`ca_key_filename`*, and *`ca_key_password`* arguments should not be specified.
`certfile`	The name of the generated certificate file.
`privatekeyfile`	The name of the generated private key file.
`keyfilepassword`	The password for the private key.
`keystrength`	The length (in bits) of the keys to be generated. The longer the key, the more difficult it is for someone to break the encryption.

Argument	Definition
`commonname`	The name to be associated with the generated certificate.
`orgunit`	The name of the organizational unit associated with the generated certificate.
`organization`	The name of the organization associated with the generated certificate.
`locality`	The name of a city or town.
`state`	The name of the state or province in which the organizational unit (ou) operates if your organization is in the United States or Canada, respectively. Do not abbreviate.
`countrycode`	Two-letter ISO code for your country. The code for the United States is US.
`subjectkeyidentifi er`	Generates a certificate with the Subject Key identifier extension and the ID value specified on the command line.
`UTF-8|BASE64`	Format of the `subjectkeyid` value. Allowed values are UTF-8 or BASE64, with UTF-8 assumed by default.

2. Use the`ImportPrivateKey` utility load the digital certificate and private key into a keystore. See utils.ImportPrivateKey in the *WebLogic Server Command Reference*.

   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 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 Microsoft p7b Format to PEM Format" on page 8-10. You can use the steps in "Converting a Microsoft p7b Format to PEM Format" 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 the Server Certificate File Name attribute in the SSL Configuration portion of the Keystores and SSL tab in the WebLogic Server Administration Console.

Listing 8-1 shows a sample certificate chain.

Listing 8-1 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-----
```

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.

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 8-10.
   - If the trusted CA certificate has a custom file type, use the steps in "Converting a Microsoft p7b Format to PEM Format" on page 8-10 to convert the trusted CA certificate to PEM format.

2. Create a Trust keystore. For more information, see "Common Keytool Commands" on page 8-14.

3. Store the trusted CA certificate in the Trust keystore. For more information, see "Common Keytool Commands" on page 8-14.

4. Configure WebLogic Server to use the Trust keystore. For more information, see "Configuring Keystores" on page 8-16.

Getting 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.

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 attributes when using a file or a JKS keystore accessed via the WebLogic Keystore provider.

For more information, see:

- "Configuring Keystores" on page 8-16

- "Configuring SSL" on page 8-18

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 "Common Keytool Commands" on page 8-14. 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.

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 likey 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.

By default, WebLogic Server looks for an Identity keystore named `DemoIdentity.jks` in the *`WL_HOME`*`\server\lib` directory and Trust keystores named `DemoTrust.jks` in the *`WL_HOME`*`\server\lib` directory and `cacerts` in the `JAVA_HOME\jre\lib\security` directory.

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 the Private Key Alias attribute 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.

Common Keytool Commands

Table 8-1 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 8-1 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.
`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.

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

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.

Use the steps in this section to configure Identity and Trust keystores for production use.

Before you perform the steps in this section, you need to:

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 8-4.
2. Create Identity and Trust keystores. For more information, see "Storing Private Keys, Digital Certificates, and Trusted Certificate Authorities" on page 8-12.

3. Load the private keys and trusted CAs into the Identity and Trust keystores. For more information, see "Storing Private Keys, Digital Certificates, and Trusted Certificate Authorities" on page 8-12.

To configure Identity and Trust keystores for WebLogic Server:

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

   The information about the demonstration keystores is displayed in the Keystore Configuration.
4. Click the Change... link in the Keystore Configuration to configure new keystores.
5. Choose the type of keystore configuration being used. The following options are available:
   - Demo Identity and Demo Trust—The demonstration Identity and Trust keystores located in the `WL_HOME\server\lib` directory and configured by default and the `cacerts` file in the `JAVA_HOME\jre\lib\security` directory.
   - Custom Identity and Java Standard Trust—An Identity keystore you create and the trusted CAs defined in the cacerts file in the `JAVA_HOME\jre\lib\security` directory.
   - Custom Identity and Custom Trust—Identity and Trust keystores you create.
   - Custom Identity and Command-Line Trust—An Identity keystore you create and command-line arguments that specify the location of the Trust keystore. Use this option in a production environment when the Administration port is enabled and Managed servers are started on the command-line instread of by the Node Manager.
6. Click Continue.
7. Define attributes for the Identity keystore.
   - Custom Identity Keystore File Name—The fully qualified path to the Identity keystore.
   - Custom Identity Keystore Type—The type of the keystore. Generally, this attribute is `jks`. If this attribute is not specified, the default keystore type defined in the security policy file for the JDK is used.
   - Custom Identity Keystore PassPhrase—The password defined when creating the keystore. This attribute is optional or required 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. Confirm the password.

Note: The passphrase for the Demo Identity keystore is `DemoIdentityKeyStorePassPhrase`.

8. Define attributes for the Trust keystore.

   If you choose Java Standard Trust, specify the password defined when creating the keystore. Confirm the password.

   If you choose Custom Trust, define the following attributes:

   – Custom Trust Keystore File Name—The fully qualified path to the trust keystore.

   – Custom Trust Keystore Type—The type of the keystore. Generally, this attribute is `jks`. If this attribute is not specified, the default keystore type defined in the security policy file for the JDK is used.

   – Custom Trust Keystore PassPhrase—The password defined when creating the keystore. This attribute is optional or required 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. Confirm the password.

9. Click Continue.

10. Click Finish.

11. Optionally, configure the SSL attributes for WebLogic Server. You do not have to perform this step if you are using keystores for the purpose of digital signing.

12. Reboot WebLogic Server.

Configuring SSL

By default, SSL is enabled and configured to use the demonstration Identity and Trust keystores. For testing and development purposes, the SSL configuration is complete. Use the steps in this section to configure SSL for production use.

To configure SSL:

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

   Information about the demonstration Identity and Trust keystores is displayed in the Keystore Configuration.
4. Configure new Identity and Trust keystores for WebLogic Server. For more information, see "Configuring Keystores" on page 8-16.

Note: 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. If you migrated from a previous version of WebLogic Server, WebLogic Server uses the existing security configuration and sets the Files or Key Store Providers option. In this circumstance, step 4 can be skipped.

5. Click the Change... link in the SSL Configuration to configure attributes for SSL.

   The Configure SSL page appears.
6. Specify how the identity and trust for WebLogic Server is stored. The following options are available:
   - Key Stores—Use this option if you created Identity and Trust keystores for WebLogic Server. If you choose this option, go to step 8.
   - Files or Key Store Providers—Use this option if you stored private keys and trusted CA certificates in a file or in a JKS keystore accessed via the WebLogic Keystore provider (as supported in previous releases of WebLogic Server). If you choose this option, go to step 9. This option is available for the purpose of backward compatibility only and it automatcally set with security information from a previous release of WebLogic Server.
7. Click Continue.
8. Specify the alias used to load the private key into the keystore in the Private Key Alias and the password used to retrieve the private key from the keystore in the Passphrase attibute. You may have specified this information when creating the Identity keystore; however, for the purpose of SSL configuration specify the information again. Skip to step 10.

Note: You do not have to specify this information for the Trust keystore because trusted CA certificates are not individually identified to WebLogic Server with aliases. All trusted

CA certificates in a keystore identified as trusted by WebLogic Server are trusted. Therefore, WebLogic Server does not require an alias when retrieving a trusted CA certificate from the keystore.

9. Specify information about the location of identity and trust for WebLogic Server.

   Note: This step only applies if the Files or Key Store Providers option is specified.

   – Private Key File Name—The directory location of the private key for WebLogic Server. Specify a value for this attribute only if you stored the private key for WebLogic Server in a file (versus a WebLogic Keystore provider).

   – Private Key Alias—The alias specified when loading the private key for WebLogic Server from the keystore. Specify a value for this field only if you stored the private key for WebLogic Server in a keystore accessed by the WebLogic Keystore provider.

   – Passphrase—The password specified when loading the private key for WebLogic Server into the keystore. Specify a value for this field only if you stored the private key for WebLogic Server in a keystore accesssed by the WebLogic Keystore provider. Confirm the password. If you protected the private key file with a password, specify the `weblogic.management.pkpassword` command-line argument when starting the server.

   – Server Certificate File Name— The directory location of the digital certificate for WebLogic Server. If you are using a certificate chain that is deeper than two certificates, you to need to include the entire chain in PEM format in the certificate file.

   – Trusted CA File Name—The name of the file containing the PEM-encoded trusted certificate authorities.

10. Click Continue.

11. Click Finish.

12. Reboot WebLogic Server.

Configuring Two-Way SSL

By default, WebLogic Server is configured to use one-way SSL (the server passes its identity to the client). For a more secure SSL connection, use two-way SSL. In a two-way SSL connection, the client verifies the identity and trust of the server and then passes its identity to the server. The server then validates the identity and trust of the client before completing the SSL connection. The server determines whether or not two-way SSL is used.

Before configuring two-way SSL, ensure the Trust key store for the server includes the certificate for the trusted certificate authority that signed the certificate for the client.

To enable two-way SSL:

1. Configure one-way SSL as described in "Configuring SSL" on page 8-18.
2. Expand the Servers node.
3. Select the name of the server for which you want to configure two-way SSL (for example, exampleserver).
4. Select the Configuration-->Keystores and SSL tab.
5. Click the Show link under Advanced Options.
6. Go to the Server attributes section of the window.
7. Set the Two Way Client Cert Behavior attribute. The following options are available:
   - Client Certs Not Requested—The default (meaning one-way SSL).
   - Client Certs Requested But Not Enforced—Requires a client to present a certificate. If a certificate is not presented, the SSL connection continues.
   - Client Certs Requested And Enforced—Requires a client to present a certificate. If a certificate is not presented or if the certificate is not trusted, the SSL connection is terminated.
8. Click Apply.
9. Reboot WebLogic Server.

Disabling the SSL Port

You may encounter a circumstance where you need to disable the SSL port. For example, the Administration Port uses SSL internally to protect communications between the Administration Server, Managed Servers, and the Node Manager. When using the Administration Port, you might want to disable the SSL port.

To disable the SSL port:

1. Expand the Server node.
2. Select the General tab.
3. Uncheck the SSL Port Enabled option.

4. Click Apply to save your changes.
5. Reboot WebLogic Server.

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.

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.

To verify that host name verification is enabled for your WebLogic Server deployment, perform the following steps for each server in your domain.

Note: The following steps only apply when a WebLogic Server instance is acting as an SSL client. Java clients acting as SSL clients specifiy the use of host name verification via command-line arguments.

1. Expand the Servers node.
2. Select the name of the server (for example, exampleserver).
3. Select the Configuration-->Keystores and SSL tab.
4. Click the Show link under Advanced Options.
5. Go to the Client attributes section of the window.
6. Verify that the the Hostname Verification field is set to BEA Hostname Verifier.

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.

Turn off host name verification in one of the following ways:

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

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

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

- In the WebLogic Server Administration Console:

1. Expand the Servers node.
2. Select the name of the server (for example, exampleserver).
3. Select the Configuration-->Keystores and SSL tab.
4. Click the Show link under Advanced Options.
5. Go to the Client attributes section of the window.
6. Set the Hostname Verification field to None.
7. Reboot WebLogic Server.

You can also 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.

To configure a custom host name verifier:

1. Expand the Servers node.
2. Select the name of the server (for example, exampleserver).
3. Select the Configuration-->Keystores and SSL tab.
4. Click the Show link under Advanced Options.
5. Go to the Client attributes section of the window.
6. Set the Hostname Verification field to Custom Hostname Verifier.
7. Enter the name of the implementation of the `weblogic.security.SSL.HostnameVerifier` interface in the Custom Hostname Verifier field.
8. Reboot WebLogic Server.

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 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 tab 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. This behavior changed in this release of WebLogic Server. The `weblogic.security.SSL.sessionCache.size` command-line argument is ignored.

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.

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 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.

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.

Configuring SSL for the Node Manager

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. The following sections describe 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.

- "SSL Requirements for Administration Servers" on page 8-26
- "SSL Requirements for Managed Servers" on page 8-27
- "SSL Requirements for the Node Manager" on page 8-28
- "Host Name Verification Requirements" on page 8-30
- "Host Name Verification Requirements" on page 8-30
- "Node Manager SSL Production Configuration: Main Steps" on page 8-34
- "Node Manager SSL Production Configuration: Main Steps" on page 8-34
- "Using Files and the WebLogic Keystore Provider" on page 8-38

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.

If the Administration port is enabled, a Managed Server needs to use the SSL protocol to contact the Administration Server to download its configuration information as defined in the `config.xml` file. However, the `config.xml` file contains SSL configuration for the Managed Server. Therefore, the Managed Server needs uses the SSL protocol to obtain its SSL configuration information as defined in the `config.xml` file. To break the deadlock, the Managed Server defaults to demonstration trust until it has contacted the Administration Server and downloaded its configuration information. The Managed Server can also use command-line switches which explicitly set trust. Use these switches in a production environment.

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 congfiguration.

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" on page 8-4.

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.

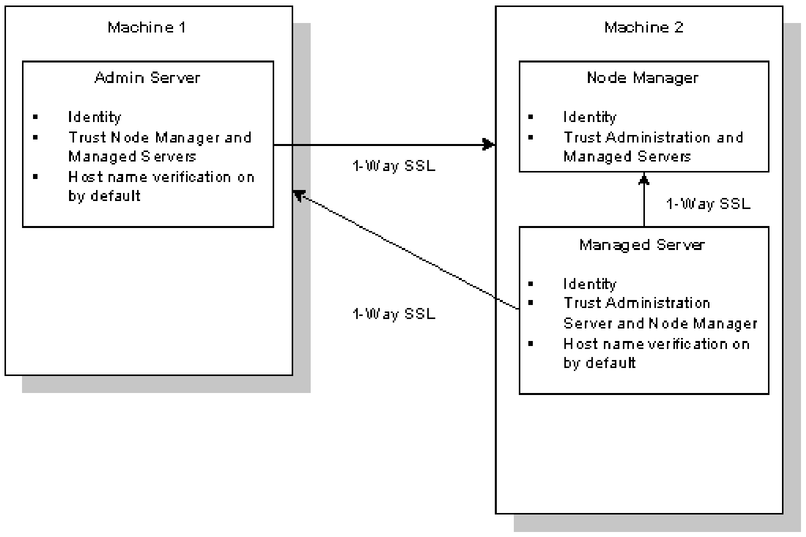
5. Edit the `nodemanager.properties` file to specify the keystore location and passwords for the Node Manager.

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 attributes and ensuring that the Administration Server and any Managed Servers are listening for SSL communications on different ports.

Figure 8-1 illustrates the SSL demonstation configuration for the Node Manager.

Figure 8-1 SSL Demonstration Configuration for the Node Manager


Machine 1
Admin Server
Identity
Trust Node Manager and Managed Servers
Host name verification on by default
1-Way SSL
1-Way SSL
Machine 2
Node Manager
Identity
Trust Administration and Managed Servers
1-Way SSL
Managed Server
Identity
Trust Administration Server and Node Manager
Host name verification on by default

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.

   See *Example: Creating a Domain with Administration Server and Clustered Managed Servers*.

2. Start the Administration Server. Do not enable the Administration port.

3. On the Server-->Keystores and SSL tab, verify the settings of the following Identity keystore attributes for the Administration Server:

- Demo Identity Keystore—`DemoIdentity.jks`
- Type—`JKS`
- Passphrase—`DemoIdentityKeyStorePassPhrase`

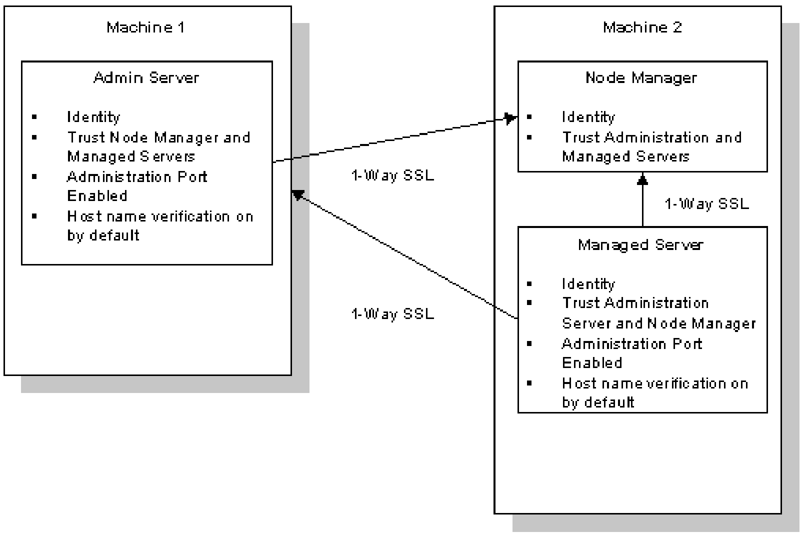
4. On the Server-->Keystores and SSL tab, verify the settings of the following Trust keystore attributes for the Administration Server:
   - Demo Trust Keystore—`DemoTrust.jks`
   - Type—`JKS`
   - Passphrase—`DemoTrustKeyStorePassPhrase`
5. On the Server-->Keystores and SSL tab, verify the settings of the following Identity keystore attributes for the Managed Server:
   - Demo Identity Keystore—`DemoIdentity.jks`
   - Type—`JKS`
   - Passphrase—`DemoIdentityKeyStorePassPhrase`
6. On the Server-->Keystores and SSL tab, verify the settings of the following Trust keystore attributes 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 8-2 illustrates the SSL production configuration for the Node Manager.

Figure 8-2 SSL Production Configuration for the Node Manager


Machine 1
Admin Server
Identity
Trust Node Manager and Managed Servers
Administration Port Enabled
Host name verification on by default
1-Way SSL
1-Way SSL
Machine 2
Node Manager
Identity
Trust Administration and Managed Servers
1-Way SSL
Managed Server
Identity
Trust Administration Server and Node Manager
Administration Port Enabled
Host name verification on by default

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 digitial 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" on page 8-12.
3. Load the private keys into the Identity keystores. See "Storing Private Keys, Digital Certificates, and Trusted Certificate Authorities" on page 8-12.
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" on page 8-12.
5. Load the trusted CA certificates into the Trust keystore. See "Storing Private Keys, Digital Certificates, and Trusted Certificate Authorities" on page 8-12.
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" on page 8-16.
11. Configure the Trust keystore for the Administration Server. See "Configuring Keystores" on page 8-16.

12. If you have multiple servers running on the same machine, ensure the Administration Port for each server is unique. Use the Local Administration Port Override attribute on the Server-->General tab 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" on page 8-16.
15. Configure the Trust keystore for the Managed Server. See "Configuring Keystores" on page 8-16.
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
       ```

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

The `CustomIdentityKeyStorePassPhrase` attribute 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`

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

The `CustomTrustKeyStorePassPhrase` attribute 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`

The `JavaStandardTrustKeyStorePassPhrase` attribute 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

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 tab.

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

   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 for this attribute only if you stored the private key for WebLogic Server in a file (versus a WebLogic Keystore provider).

- Private Key Alias—The alias specified when loading the private key for WebLogic Server from the keystore. Specify a value for this field only if you stored the private key for WebLogic Server in a keystore accessed by the WebLogic Keystore provider.
- Passphrase—The password specified when loading the private key for WebLogic Server into the keystore. Specify a value for this field only if you stored the private key for WebLogic Server in a keystore accesssed by the WebLogic Keystore provider. Confirm the password. If you protected the private key file with a password, specify the `weblogic.management.pkpassword` command-line argument when starting the server.
- Server Certificate File Name— The directory location of the digital certificate for WebLogic Server. If you are using a certificate chain that is deeper than two certificates, you to need to include the entire chain in PEM format in the certificate file.

8. Click Continue.
9. Click Finish.
10. Reboot WebLogic Server.
11. When starting the Administration Server, use the following command-line argument to specify the location of the Trust keystore:

    ```
    -Dweblogic.security.SSL.TrustedCAKeyStore=path_to_keystore
    ```

12. Specify the location of the Trust keystore for the Managed Server on the Servers-->Configuration-->Remote Start tab as follows:

    ```
    weblogic.security.SSL.trustedCAKeyStore
    ```

To use files or a JKS keystore to store identity and trust for the Node Manager, specify the following command-line arguments when starting the Node Manager:

- Use `weblogic.nodemanager.keyFile=`*`filename`* to specify the location of the private key file.
- If you password protected the private key file, use `weblogic.nodemanager.keyPassword=`*`password`* to specify the password.
- Use `weblogic.nodemanager.certificateFile=`*`filename`* to specify the location of the digital certificate for the Node Manager.
- Use `weblogic.security.SSL.trustedCAKeyStore=`*`keystorename`* to specify the location of the JKS trusted keystore.

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 8-2 describes the options for the command-line argument.

Table 8-2 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.

`strict`	Use this option to check whether 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 turn off checking for the Basic Constraints extension. The rest of the certificate validation still happens. CA certificates from most commercial certificate authorities should work with the default `strong` option. For example: `-Dweblogic.security.SSL.enforceConstraints=off` BEA does not recommend using this option in a production environment. Instead, purchase new CA certificates that comply with the IETF RFC 2459 standard.

Accepting Certificate Policies in Certificates

WebLogic Server offers limited support for Certificate Policy Extensions in X.509 certificates. Use the `weblogic.security.SSL.allowedcertificatepolicyids` argument to provide a comma separated list of Certificate Policy IDs. When WebLogic Server receives a certificate with a critical Certificate Policies Extension, it verifies whether any Certificate Policy is on the list of allowed certificate policies and whether therea are any unsupported policy qualifiers. This release of WebLogic Server supports Certification Practice Statement (CPS) Policy qualifiers and does not support User Notice qualifiers. A certificate is also accepted if it contains a special policy `anyPolicy` with the ID 2.5.29.32.0, which indicates that the CA does not wish to limit the set of policies for this certificate.

To enable acceptance of Certificate Policies, start WebLogic Server with the following argument:

```
-Dweblogic.security.SSL.allowedcertificatepolicyids
<identifier1>,<identifier2>,...
```

This argument should contain a comma-separated list of Certificate Policy identifiers for all the certificates with critical extensions that might be present in the certificate chain, back to the root certificate, in order for WebLogic Server to accept such a certificate chain.

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 Certificates

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 strengtth 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, when acting as an SSL server, will agree to either SSL 3.0 or TLS 1.0 protocol versions, and use whichever of these the client has specified as preferred in its client hello message. When WebLogic Server is acting as an SSL client, it will specify TLS1.0 as the preferred protocol in its SSL V2.0 client hello message, but will agree to SSL V3.0 as well, if that's the highest version that the SSL server on the other end supports. The peer must respond with an SSL V3.0 or TLS V1.0 message or the SSL connection is dropped.

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.

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

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.

To disable two-way SSL when using WebLogic Server:

1. Expand the Servers node.
2. Select the server that will be acting as the SSL server.
3. Select the Configuration-->Keystores and SSL tab.
4. Click the Show link in the Advanced Options section.
5. Set the Two Way Client Cert Behavior attribute to Client Certs Not Requested or Client Certs Requested But Not Enforced.
6. Click Apply.
7. Reboot WebLogic Server.

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 digitial 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.

Protecting User Accounts

This following sections describe how to protect user accounts and how to unlock a user account:

- "Setting Lockout Attributes for User Accounts" on page 9-2
- "Unlocking a User Account" on page 9-4

Note: For information about protecting user accounts in Compatibility security, see "Protecting User Accounts in Compatibilty Security" on page 11-5.

Protecting Passwords

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.

Setting Lockout Attributes for User Accounts

WebLogic Server defines a set of attributes to protect user accounts from intruders. In the default security configuration, these attributes are set for maximum protection. When creating a new security realm, you need to define these attributes.

As a system administrator, you have the option of turning off all the attributes, 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 attributes lessens security and leaves user accounts vulnerable to security attacks.

To set the user lockout attributes:

1. Expand the Security-->Realms nodes.
2. Select the name of the realm you are configuring (for example, *myrealm*).
3. Select the User Lockout tab.
4. Define the desired attributes on this tab by entering values at the appropriate prompts and selecting the required checkboxes. (For details, see the following table).

   If a user account exceeds the values set for the attributes on this tab, the user account becomes locked and the table on the Users tab has the word `Details` in the table row for the user account. For more information, see "Unlocking a User Account" on page 9-4.
5. To save your changes, click Apply.
6. Reboot WebLogic Server.

The following table describes each attribute on the User Lockout tab.

Table 9-1 User Lockout Attributes

Attribute	Description
Lockout Enabled	Requests the locking of a user account after invalid attempts to log in to that account exceed the specified Lockout Threshold. By default, this attribute is enabled.
Lockout Threshold	Number of failed user password entries that can be tried before that user account is locked. Any subsequent attempts to access the account (even if the username/password combination is correct) raise a Security exception; the account remains locked until it is explicitly unlocked by the system administrator or another login attempt is made after the lockout duration period ends. Invalid login attempts must be made within a span defined by the Lockout Reset Duration attribute. The default is 5.
Lockout Duration	Number of minutes that a user's account remains inaccessible after being locked in response to several invalid login attempts within the amount of time specified by the Lockout Reset Duration attribute. The default is 30 minutes.
Lockout Reset Duration	Number of minutes within which invalid login attempts must occur in order for the user's account to be locked. An account is locked if the number of invalid login attempts defined in the Lockout Threshold attribute happens within the amount of time defined by this attribute. For example, if the value in Lockout Reset Duration attribute is 5 minutes, the Lockout Threshold is 3, and 3 invalid login attempts are made within a 6 minute interval, then the account is not locked. If 3 invalid login attempts are made within a 5 minute period, however, then the account is locked. The default is 5 minutes.

Table 9-1 User Lockout Attributes

Attribute	Description
Lockout Cache Size	Specifies the intended cache size of unused and invalid login attempts. The default is 5.
Lockout GC Threshold	The maximum number of invalid login records that the server keeps in memory. If the number of invalid login records is equal to or greater than the value of this attribute, the server's garbage collection purges the records that have expired. A record expires when a user is unlocked or when the lockout reset duration has expired for that record. The default is 400 records.

The User Lockout attributes apply to the default security realm and all its security providers. The User Lockout attributes do not work with custom security providers in a security realm other than the default security realm. To use the User Lockout attributes 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 the Lockout Enabled attribute.

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 attributes will not be reset.

Unlocking a User Account

To unlock a locked user account on a managed server, a user with Admin privileges can use the following command:

```
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.

To unlock a user account using the Administration Console:

1. Expand the Monitoring-->Security tab for the server.
2. In the User table, click on the Details link for the user to be unlocked.
3. Click Unlock.

Configuring Security for a WebLogic Domain

The following sections describe how to set security attributes on 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: Enabling trust between WebLogic Server domains opens the servers up to man-in-the-middle attacks. Great care should be taken when enabling trust in a production environment. BEA recommends having strong network security such as a dedicated communication channel or protection by a strong 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 attribute for one domain matches the Domain Credential attribute 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.

By default, when you boot an Administration Server for the first time, the Domain Credential attribute is not defined. As the Administration Server boots, it notices that the Domain Credential attribute 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 attribute). 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 the Domain Credential attribute in the WebLogic Server domain to the password of the `system` user in the WebLogic Server 6.x domain.

If you want two WebLogic Server domains to interoperate, perform the following procedure in both domains:

1. Expand the Domains node (for example, Examples).
2. Click the View Domain-Wide Security Settings link on the Domain-->General page.
3. Select the Security Configuration-->Advanced tab.
4. Uncheck the Enable Generated Credential attribute.
5. Enter a password for the domain in the Credential text field. Choose the password carefully. BEA Systems recommends using a combination of upper and lower case letters and numbers.
6. Confirm the password by entering it in the Confirm Credential text field.
7. Click Apply.

If you are enabling this feature 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 server's class path in all domains that share trust.

Configuring Connection Filtering

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. Expand the Domains node.
2. On the Domain-->General tab, click the View Domain-Wide Security Settings link.
3. Select the Security-->Filter tab.
4. Click the Connection Logger Enabled attribute to enable the logging of accepted messages. This attribute logs successful connections and connection data in the server. This information can be used to debug problems relating to server connections.
5. Specify the connection filter to be used in the domain.
   - To configure the default connection filter, specify `weblogic.security.net.ConnectionFilterImpl` in the Connection Filter attribute field.
   - To configure a custom connection filter, specify the class that implements the network connection filter in the Connection Filter attribute. This class must also be specified in the CLASSPATH for WebLogic Server.
6. Enter the syntax for the connection filter rules. For more information about connection filter rules, see "Using Network Connection Filters".
7. Click Apply.
8. Reboot WebLogic Server.

Viewing MBean Attributes

The Anonymous Admin Lookup Enabled attribute 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 attribute is checked by default to sure backward compatibility.

To verify the setting of the Anonymous Admin Lookup Enabled attribute:

1. Expand the Domains node.
2. On the Domain-->General tab, click the View Domain-Wide Security Settings link.

3. Select the Security-->General tab.
4. Verify that the Anonymous Admin Lookup Enabled attribute is checked. Unchecking this attribute will make the server more secure; however, it might cause applications that require anonymous access to MBeans to stop working properly.
5. Reboot WebLogic Server.

Using Compatibility Security

The following sections describe how to configure Compatibility security:

- "Running Compatibility Security: Main Steps" on page 11-1
- "The Default Security Configuration in the CompatibilityRealm" on page 11-2
- "Configuring the Identity Assertion Provider in the Realm Adapter Authentication Provider" on page 11-3
- "Configuring a Realm Adapter Auditing Provider" on page 11-4
- "Protecting User Accounts in Compatibilty Security" on page 11-5
- "Accessing 6.x Security from Compatibility Security" on page 11-7

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" on page 3-32.

  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" on page 11-3.

- 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 can be used only with the Realm Adapter Authentication provider and the WebLogic Authorization provider. It resolves access decision conflicts between ACLs and new security policies set through the WebLogic Administration Console. The Realm Adapter Adjudication provider permits access if the one authorization provider votes PERMIT and the other authorization provider votes DENY.

- The WebLogic Credential Mapping provider allows the use of credential maps in Compatibility security. For more information, see Chapter 5, "Single Sign-On with Enterprise Information Systems."

- 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" on page 11-4.

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.

To enable identity assertion in the Realm Adapter Authentication provider:

1. Expand the Security-->Realms nodes.
2. Select the CompatibilityRealm.
3. Expand the Providers node.
4. Select Authentication Providers.
5. Click the Realm Adapter Authenticator link in the Realms table.

   The General tab appears.
6. Enter X.509 in the Active Types list box.

   This step enables the use of 6.x Cert Authenticators.
7. Click Apply.
8. Reboot WebLogic Server.

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 the Audit Provider class attribute on the Domain-->Security-->Compatibility-->General tab.

To configure a Realm Adapter Auditing provider:

1. Expand the Compatibility Security-->Realms nodes.
2. Expand the Providers node.
3. Select Auditors.
4. Click Configure a Realm Adapter Auditor... link.

   The General tab appears
5. Click Create to save your changes.

6. Reboot WebLogic Server.

Protecting User Accounts in Compatibilty Security

Weblogic Server provides a set of attributes to protect user accounts from intruders. By default, these attributes are set for maximum protection. As a system administrator, you have the option of turning off all the attributes, 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 attributes lessens security and leaves user accounts vulnerable to security attacks.

There are two sets of attributes 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 attributes (for example, the attributes for the security realm) and exceed any of the values, the user account is not locked. This happens because the user account attributes at the domain override the user account attributes at the security realm. To avoid this situation, disable the user account attributes at the security realm.

To protect the user accounts in your WebLogic Server domain, perform the following steps:

1. Expand the Domain node (for example, mydomain).
2. At the bottom of the General tab, click the View Domain-Wide Security Settings link.
3. Select the Compatibility-->Passwords tab.
4. Define the desired attributes on this tab by entering values at the appropriate prompts and selecting the required checkboxes. (For details, see the following table).
5. Click Apply to save your choices.
6. Reboot WebLogic Server.

The following table describes each attribute on the Passwords tab.

Table 11-1 Password Protection Attributes

Attribute	Description
Minimum Password Length	Number of characters required in a password. Passwords must contain a minimum of 8 characters. The default is 8.
Lockout Enabled	Requests the locking of a user account after invalid attempts to log in to that account exceed the specified Lockout Threshold. By default, this attribute is enabled.
Lockout Threshold	Number of failed password entries for a user that can be tried to log in to a user account before that account is locked. Any subsequent attempts to access the account (even if the username/password combination is correct) raise a Security exception; the account remains locked until it is explicitly unlocked by the system administrator or another login attempt is made after the lockout duration period ends. Invalid login attempts must be made within a span defined by the `Lockout Reset Duration` attribute. The default is 5.
Lockout Duration	Number of minutes that a user's account remains inaccessible after being locked in response to several invalid login attempts within the amount of time specified by the `Lockout Reset Duration` attribute. In order to unlock a user account, you need to have the `unlockuser` permission for the `weblogic.passwordpolicy`. The default is 30 minutes.

Table 11-1 Password Protection Attributes

Attribute	Description
Lockout Reset Duration	Number of minutes within which invalid login attempts must occur in order for the user's account to be locked. An account is locked if the number of invalid login attempts defined in the `Lockout Threshold` attribute happens within the amount of time defined by this attribute. For example, if the value in this attribute is five minutes and three invalid login attempts are made within a six-minute interval, then the account is not locked. If five invalid login attempts are made within a five-minute period, however, then the account is locked. The default is 5 minutes.
Lockout Cache Size	Specifies the intended cache size of unused and invalid login attempts. The default is 5.

To disable the user account attributes at the security realm:

1. Expand the Security-->Realms nodes.
2. Expand the CompatibilityRealm node.
3. Select the User Lockout tab.
4. Uncheck the Lockout Enabled attribute.
5. Click Apply.
6. Reboot WebLogic Server.

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

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 section 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