



BEA WebLogic Server™

Managing WebLogic Security

Copyright

Copyright © 2003 BEA Systems, Inc. All Rights Reserved.

Restricted Rights Legend

This software and documentation is subject to and made available only pursuant to the terms of the BEA Systems License Agreement and may be used or copied only in accordance with the terms of that agreement. It is against the law to copy the software except as specifically allowed in the agreement. This document may not, in whole or in part, be copied photocopied, reproduced, translated, or reduced to any electronic medium or machine readable form without prior consent, in writing, from BEA Systems, Inc.

Use, duplication or disclosure by the U.S. Government is subject to restrictions set forth in the BEA Systems License Agreement and in subparagraph (c)(1) of the Commercial Computer Software-Restricted Rights Clause at FAR 52.227-19; subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013, subparagraph (d) of the Commercial Computer Software--Licensing clause at NASA FAR supplement 16-52.227-86; or their equivalent.

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

Trademarks or Service Marks

BEA, Jolt, Tuxedo, and WebLogic are registered trademarks of BEA Systems, Inc. BEA Builder, BEA Campaign Manager for WebLogic, BEA eLink, BEA Manager, BEA WebLogic Commerce Server, BEA WebLogic Enterprise, BEA WebLogic Enterprise Platform, BEA WebLogic Express, BEA WebLogic Integration, BEA WebLogic Personalization Server, BEA WebLogic Platform, BEA WebLogic Portal, BEA WebLogic Server, BEA WebLogic Workshop and How Business Becomes E-Business are trademarks of BEA Systems, Inc.

All other trademarks are the property of their respective companies.

Managing WebLogic Security

Part Number	Document Revised	Software Version
N/A	June 30, 2003	BEA WebLogic Server Version 7.0

Contents

About This Document

Audience.....	x
e-docs Web Site.....	x
How to Print the Document.....	xi
Related Information.....	xi
Contact Us!.....	xi
Documentation Conventions.....	xii

1. Overview of Security Management

Audience.....	1-1
How Security Changed in WebLogic Server	1-2
Change in Scope of Security Realms	1-2
Security Providers	1-3
Security Policies Instead of ACLs	1-5
WebLogic Resources.....	1-6
Deployment Descriptors and the WebLogic Server Administration Console .	1-7
The Default Security Configuration in WebLogic Server.....	1-8
Configuration Steps for Security	1-8
What Is Compatibility Security?	1-10
Management Tasks Available in Compatibility Security.....	1-10

2. Customizing the Default Security Configuration

Why Customize the Default Security Configuration?.....	2-1
Creating a New Security Realm	2-3
Setting a New Security Realm as the Default (Active) Security Realm	2-5
Deleting a Security Realm.....	2-6

Reverting to a Previous Security Configuration	2-7
3. Configuring Security Providers	
When Do I Need to Configure a Security Provider?	3-2
Configuring a WebLogic Adjudication Provider	3-4
Configuring a WebLogic Auditing Provider	3-5
Choosing an Authentication Provider	3-7
Configuring an Authentication Provider: Main Steps	3-8
Setting the JAAS Control Flag Attribute	3-10
Configuring an LDAP Authentication Provider	3-11
Requirements for Using an LDAP Authentication Provider	3-11
Configuring a LDAP Authentication Provider	3-12
Setting LDAP Server and Caching Information	3-13
Locating Users in the LDAP Directory	3-16
Locating Groups in the LDAP Directory	3-18
Locating Members of a Group in the LDAP Directory	3-20
Configuring Failover for LDAP Authentication Providers	3-21
Configuring a WebLogic Authentication Provider	3-24
Configuring a Realm Adapter Authentication Provider	3-26
Configuring a WebLogic Identity Assertion Provider	3-28
Using a User Name Mapper with the WebLogic Identity Assertion Provider	3-30
Configuring an LDAP X509 Identity Assertion Provider	3-31
Ordering of Identity Assertion for Servlets	3-40
Configuring a WebLogic Authorization Provider	3-41
Configuring a WebLogic Credential Mapping Provider	3-42
Configuring a WebLogic Keystore Provider	3-43
Configuring a WebLogic Role Mapping Provider	3-44
Configuring a Custom Security Provider	3-45
Deleting a Security Provider	3-46
4. Single Sign-On with Enterprise Information Systems	
Overview	4-1
Using Deployment Descriptors to Create Credential Maps	4-2
Using the WebLogic Administration Console to Create Credential Maps	4-5

5. Managing the Embedded LDAP Server

Configuring the Embedded LDAP Server.....	5-1
Configuring Backups for the Embedded LDAP Server	5-4
Viewing the Contents of the Embedded LDAP Server from an LDAP Browser ...	5-5
Exporting and Importing Information in the Embedded LDAP Server	5-6
Access Control Syntax	5-7
The Access Control File.....	5-7
Access Control Location	5-8
Access Control Scope.....	5-8
Access Rights and Permissions	5-9
Attribute Permissions	5-9
Entry Permissions	5-10
Attributes Types	5-13
Subject Types	5-13
Grant/Deny Evaluation Rules.....	5-14

6. Configuring SSL

SSL: An Introduction	6-2
Private Keys, Digital Certificates and Trusted Certificate Authorities	6-3
One-Way and Two-Way SSL.....	6-3
Setting Up SSL: Main Steps.....	6-4
Obtaining Private Keys, Digital Certificates and Trusted Certificate Authorities ..	6-5
Using the Cert Gen Utility	6-7
Using the Certificate Request Generator Servlet	6-8
Using Certificate Chains	6-11
Converting a Microsoft p7b Format to PEM Format	6-12
Using Your Own Certificate Authority	6-13
Getting a Digital Certificate for a Web Browser.....	6-14
Storing Private Keys, Digital Certificates, and Trusted Certificate Authorities	6-14
Creating a Keystore and Loading Private Keys and Trusted Certificate	
Authorities into the Keystore	6-15
Common Keytool Commands.....	6-17

Configuring the WebLogic Server Keystore Provider to Locate a Keystore ..	
6-18	
Using a JKS Keystore.....	6-20
Enabling the SSL Port	6-21
Setting Attributes for One-Way SSL.....	6-21
Setting Attributes for Two-Way SSL	6-23
Command-Line Arguments for SSL.....	6-24
Enabling SSL Debugging	6-25
SSL Session Behavior	6-26
Using Host Name Verification	6-27
Configuring SSL for the Node Manager	6-28
SSL Requirements for Administration Servers.....	6-29
SSL Requirements for Managed Servers	6-31
SSL Requirements for the Node Manager.....	6-32
Identity and Trust: Demonstration Versus Production.....	6-33
Host Name Verification Requirements	6-34
Node Manager SSL Demonstration Configuration: Main Steps.....	6-34
Node Manager SSL Production Configuration: Main Steps	6-37
Configuring the Administration Server to Use SSL.....	6-39
Configuring a Managed Server to Use SSL	6-40
Configuring the Node Manager to Use SSL.....	6-42
Configuring RMI over IIOP with SSL	6-44
SSL Certificate Validation.....	6-45
Controlling the Level of Certificate Validation.....	6-45
Checking Certificate Chains.....	6-47
Troubleshooting Problems with Certificates	6-48
Using the nCipher JCE Provider with WebLogic Server	6-48
Specifying the Version of the SSL Protocol.....	6-50
Using the SSL Protocol to Connect to WebLogic Server from weblogic.Admin ..	
6-51	
Ensure Two-Way SSL is Disabled on the SSL Server.....	6-52
Use a Secure Port in the URL.....	6-52
Specify Trust for weblogic.Admin.....	6-52
Specify Host Name Verification for weblogic.Admin	6-53
Using the SSL Protocol with a BEA Tuxedo Client and WebLogic Server ...	6-54

7. Protecting User Accounts

Protecting Passwords.....	7-1
Setting Lockout Attributes for User Accounts	7-2
Unlocking a User Account	7-4

8. Configuring Security for a WebLogic Domain

Enabling Trust Between WebLogic Domains.....	8-1
Configuring Connection Filtering	8-3

9. Using Compatibility Security

Running Compatibility Security: Main Steps.....	9-1
The Default Security Configuration in the CompatibilityRealm	9-3
Configuring the Identity Assertion Provider in the Realm Adapter Authentication Provider	9-4
Configuring a Realm Adapter Auditing Provider	9-5
Protecting User Accounts in Compatibilty Security	9-6
Accessing 6.x Security from Compatibility Security	9-8



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 Compatibility 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, “Single Sign-On with Enterprise Information Systems,”](#) explains how to create credential maps so EIS users can access WebLogic resources.
- [Chapter 5, “Managing the Embedded LDAP Server,”](#) describes the management tasks associated with the embedded LDAP server used by the WebLogic security providers.
- [Chapter 6, “Configuring SSL,”](#) describes how to configure SSL for WebLogic Server.
- [Chapter 7, “Protecting User Accounts,”](#) describes setting attributes to protect user accounts and how to unlock a user account.
- [Chapter 8, “Configuring Security for a WebLogic Domain,”](#) describes how to set security attributes on a WebLogic domain.

-
- [Chapter 9, “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 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.

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 security realm, 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.
<i>italics</i>	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. <i>Examples:</i> <pre>import java.util.Enumeration; chmod u+w * config/examples/applications .java config.xml float</pre>

Convention	Usage
<i>monospace</i> <i>italic</i> <i>text</i>	Placeholders. <i>Example:</i> <code>String CustomerName;</code>
UPPERCASE MONOSPACE TEXT	Device names, environment variables, and logical operators. <i>Examples:</i> <code>LPT1</code> <code>BEA_HOME</code> <code>OR</code>
{ }	A set of choices in a syntax line.
[]	Optional items in a syntax line. <i>Example:</i> <code>java utils.MulticastTest -n name -a address</code> <code>[-p portnumber] [-t timeout] [-s send]</code>
	Separates mutually exclusive choices in a syntax line. <i>Example:</i> <code>java weblogic.deploy [list deploy undeploy update]</code> <code>password {application} {source}</code>
...	Indicates one of the following in a command line: <ul style="list-style-type: none">■ 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.
.	
.	



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.

- [“Audience” on page 1-1](#)
- [“How Security Changed in WebLogic Server” on page 1-2](#)
- [“The Default Security Configuration in WebLogic Server” on page 1-8](#)
- [“Configuration Steps for Security” on page 1-8](#)
- [“What Is Compatibility Security?” on page 1-10](#)
- [“Management Tasks Available in Compatibility Security” on page 1-10](#)

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 your 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 Compatibility 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 1-8](#).

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 9, “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:

- **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.
- **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.
- **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.

- **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.
- **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.

- **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: Custom Keystore providers are not supported in this release of WebLogic Server.
- **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)).

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 1-8.](#)

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 9, “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 5, “Managing the Embedded LDAP Server.”](#)
4. Ensure user accounts are properly secured. WebLogic Server provides a set of attribute 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 7, “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. Enable SSL for WebLogic Server. (This step is optional but encouraged.) For more information, see [Chapter 6, “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 4, “Single Sign-On with Enterprise Information Systems.”](#)

- Configure a connection filter. See [Chapter 8, “Configuring Security for a WebLogic Domain.”](#)
- Enable interoperability between WebLogic domains. See [Chapter 8, “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 Compatibility realm. 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. You must run Compatibility security in order to access the *CompatibilityRealm* and the Realm Adapter providers through the WebLogic Server Administration Console. For more information, see [Chapter 9, “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 9-5.](#)

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 9-4.

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 3-26.

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 6, “Configuring SSL”](#)
- [Chapter 8, “Configuring Security for a WebLogic Domain”](#)

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.

- [“Why Customize the Default Security Configuration?” on page 2-1](#)
- [“Creating a New Security Realm” on page 2-3](#)
- [“Setting a New Security Realm as the Default \(Active\) Security Realm” on page 2-5](#)
- [“Deleting a Security Realm” on page 2-6](#)
- [“Reverting to a Previous Security Configuration” on page 2-7](#)

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.
- Add a WebLogic Keystore provider to the default security realm. This involves creating a JKS keystore and then configuring the WebLogic Keystore provider to access it.
- Upgrade from Compatibility security to the security providers in this release of WebLogic Server.
- Change the default attribute settings of the security providers. For more information, 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. 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. Click the Configure a new Realm... link.
4. Enter the name of the new security realm in the Name attribute on the General tab.
5. To control performance, specify how the WebLogic Security Service should perform security checks.

Set the value of the Check Roles and Policies setting as follows:

- The Web Applications and EJBs protected in DD option specifies the WebLogic Security Service only performs security checks on URL (Web) and EJB resources that have security specified in their associated deployment descriptors (that is, the `ejb-jar.xml`, `weblogic-ejb-jar.xml`, `web.xml`, and `weblogic.xml` files). This is the default value.
- The All Web Applications and EJBs options specifies 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 for these WebLogic resources. If you choose this option, you also need to specify what the WebLogic Security Service should do when the Web application or EJB module is redeployed. For more information, see step 6.

Note: Prior to WebLogic Server 7.0 SP3, you had to specify how the WebLogic Security Service would perform security checks using the `fullyDelegateAuthorization` command-line argument. For more information, see “[Understanding the fullyDelegateAuthorization Flag](#)” in *Securing WebLogic Resources*.

6. Specify which technique you want to use to secure URL (Web) and EJB resources.

Set the value of the Deployment Descriptor Security Behavior setting as follows:

- To secure your URL (Web) 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 the Seed Security from Deployment Descriptor option and refer to the [“Using Declarative Security with Web Applications”](#) and [“Using Declarative Security with EJBs”](#) sections of *Programming WebLogic Security* respectively.
- To secure your URL and EJB resources using only the WebLogic Server Administration Console, select the Ignore Security Data in Deployment Descriptions option and follow the instructions for securing Web applications and EJBs described in *Securing WebLogic Resources*.

Warning: Switching the value of the Deployment Descriptor Security Behavior setting is risky and can lead to incorrect or lost security configurations. Carefully read [“Techniques for Securing URL \(Web\) and EJB Resources”](#) in *Securing WebLogic Resources* before defining a value for this setting.

7. To specify that the Credential Mapping providers in the security realm only use credential maps creating using the WebLogic Server Administration Console, check the Ignore Deploy Credential Mapping Deployment Descriptor setting. 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.
8. 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. This attribute changes the runtime behavior of the Servlet container to use a Web resource rather than a URL resource when performing authorization.
9. Click Create.
10. Configure the required security providers for the security realm. In order for a security realm to be valid, you must configure 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. For more information, 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.
11. Optionally, define Identity Assertion, Keystore, and Auditing providers. For more information, see [Chapter 3, “Configuring Security Providers.”](#)
 12. 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. For more information, see [Chapter 5, “Managing the Embedded LDAP Server.”](#)
 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. For more information, see [Chapter 7, “Protecting User Accounts.”](#)
 15. Set the new realm as the default security realm for the WebLogic domain. For more information, see [“Setting a New Security Realm as the Default \(Active\) Security Realm” on page 2-5.](#)
 16. Reboot WebLogic Server.

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

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

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

1. Expand the Domain node (for example, Examples).
2. Select the Security tab.

3. Select the General tab.

The pull-down menu on the 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 minimum 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:

1. 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 5-5](#).

To delete a security realm:

1. 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. Use the following command-line argument to revert to the last security configuration:

```
-Dweblogic.safeCommoBoot=true
```


3 Configuring Security Providers

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

- [“When Do I Need to Configure a Security Provider?” on page 3-2](#)
- [“Configuring a WebLogic Adjudication Provider” on page 3-4](#)
- [“Configuring a WebLogic Auditing Provider” on page 3-5](#)
- [“Choosing an Authentication Provider” on page 3-7](#)
- [“Configuring an Authentication Provider: Main Steps” on page 3-8](#)
- [“Setting the JAAS Control Flag Attribute” on page 3-10](#)
- [“Configuring an LDAP Authentication Provider” on page 3-11](#)
- [“Configuring a WebLogic Authentication Provider” on page 3-24](#)
- [“Configuring a Realm Adapter Authentication Provider” on page 3-26](#)
- [“Configuring a WebLogic Identity Assertion Provider” on page 3-28](#)
- [“Using a User Name Mapper with the WebLogic Identity Assertion Provider” on page 3-30](#)
- [“Configuring an LDAP X509 Identity Assertion Provider” on page 3-31](#)
- [“Ordering of Identity Assertion for Servlets” on page 3-40](#)
- [“Configuring a WebLogic Authorization Provider” on page 3-41](#)
- [“Configuring a WebLogic Credential Mapping Provider” on page 3-42](#)

- “Configuring a WebLogic Keystore Provider” on page 3-43
- “Configuring a WebLogic Role Mapping Provider” on page 3-44
- “Configuring a Custom Security Provider” on page 3-45
- “Deleting a Security Provider” on page 3-46

Note: The Realm Adapter Auditing, Adjudication, and Authorization providers are only available when using Compatibility Security. For information about those providers, see [Chapter 9, “Using Compatibility Security.”](#)

When Do I Need to Configure a Security Provider?

By default, most of the configuration work for 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. For more information, see [“Configuring a WebLogic Identity Assertion Provider” on page 3-28.](#)
- To use a user name mapper to map tokens to a user in a security realm, configure the WebLogic Identity Assertion provider. For more information, see [“Using a User Name Mapper with the WebLogic Identity Assertion Provider” on page 3-30.](#)
- To use auditing in the default (active) security realm, configure either the WebLogic Auditing provider or a custom Auditing provider. For more information, see [“Configuring a WebLogic Auditing Provider” on page 3-5](#) or [“Configuring a Custom Security Provider” on page 3-45.](#)
- To protect private keys and trusted certificate authorities in a keystore rather than a flat file, configure the WebLogic Keystore provider. For more information, see [“Configuring a WebLogic Keystore Provider” on page 3-43.](#)

Note: Custom keystore providers are not supported in this release of WebLogic Server.

- 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. For more information, 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), configure the Realm Adapter Authentication provider. The Realm Adapter Authentication provider can be used instead of or in addition to the WebLogic Authentication provider. For more information, see [“Configuring a Realm Adapter Authentication Provider” on page 3-26](#).
- 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. For more information, see [“Creating a New Security Realm” on page 2-3](#).
 - 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. For more information, 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 must have an Adjudication provider configured. 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 3-45](#).

To configure a WebLogic Adjudication provider:

1. Expand the Security-->Realms nodes.
2. Click 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.

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 3-45](#).

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. Click 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.

The General tab appears.

6. 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

7. Click Create to save your changes.

8. Reboot WebLogic Server.

All auditing information recorded by the WebLogic Auditing provider is saved in `WL_HOME\yourdomain\yourserver\DefaultAuditRecorder.log`. Although, an Auditing provider is configured per security realm, each server writes auditing data to its own log file in the server directory. 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.

Table 3-1 WebLogic Auditing Provider Events

Audit Event	Indicates...
ROLEDEPLOY	A deployRole event occurred.
ROLEUNDEPLOY	An undeployRole event occurred.
POLICYDEPLOY	A deployPolicy event occurred.
POLICYUNDEPLOY	An undeployPolicy event occurred.

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 an pre-defined LDAP providers and customize it.
- *The Realm Adapter Authentication provider* accesses user and group information stored in 6.x security realms.

- *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. Authentication providers are called in the order in which they are configured. Therefore, use caution when configuring Authentication providers. Use the JAAS Control Flag attribute to set up login dependencies between Authentication providers and allow single-sign on between providers. For more information, see [“Setting the JAAS Control Flag Attribute” on page 3-10](#).

Configuring an Authentication Provider: Main Steps

To configure an Authentication provider:

1. Expand the Security-->Realms nodes.
2. Click 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 3-24](#)
 - [“Configuring a Realm Adapter Authentication Provider” on page 3-26](#)
 - [“Configuring a WebLogic Identity Assertion Provider” on page 3-28](#)
 - [“Configuring an LDAP X509 Identity Assertion Provider” on page 3-31](#)
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. For more information, 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 provider 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 for 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 the Authentication test of these 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 providers.

When additional Authentication providers are added to an existing security realm, by default the Control Flag attribute is set to **OPTIONAL**. If necessary, changing 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 3-13](#)
- [“Locating Users in the LDAP Directory” on page 3-16](#)
- [“Locating Groups in the LDAP Directory” on page 3-18](#)
- [“Locating Members of a Group in the LDAP Directory” on page 3-20](#)

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 from which you want to boot WebLogic Server is included in the group.

The Active Directory LDAP directory has a default group called `Administrators`. Add the user from which you 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. Expand the Security-->Realms nodes.
2. Click 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 3-13](#).

Setting LDAP Server and Caching Information

To configure the LDAP server:

1. Click the LDAP tab under the Configuration tab for the LDAP Authentication provider you want to use.

For example, click the iPlanet LDAP tab under the iPlanet Configuration 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.

Table 3-2 Attributes on the LDAP Tab

Attribute	Description
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. Click 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
Group Membership Searching	<p>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.</p> <ul style="list-style-type: none">■ 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	<p>Controls the depth of a group membership search if the Group Membership Searching attribute is specified. Possible values are:</p> <ul style="list-style-type: none">■ 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.
Follow Referrals	<p>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.</p>
Bind Anonymously On Referrals	<p>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.</p>
Results Time Limit	<p>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.</p>

Table 3-3 Attributes on the Details Tab

Attribute	Description
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.

5. Proceed to [“Locating Users in the LDAP Directory” on page 3-16](#).

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 6-1](#).

Locating Users in the LDAP Directory

To specify how users are located in the LDAP directory:

1. Click the Users tab under the Configuration tab for the LDAP server you chose.
For example, click the Users tab under the iPlanet Configuration 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	<p>The attribute of an LDAP user object that specifies the distinguished name of dynamic groups to which this user belongs.</p> <p>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.</p> <p>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.</p> <p>If a group contains other groups, WebLogic Server evaluates the URLs of any of the descendents of the group.</p>
User Base DN	<p>The base DN of the tree in the LDAP directory that contains users.</p> <p>If you store WebLogic Server users under multiple roots in your directory, you can specify these roots as <code>user.dn.1</code>, <code>user.dn.2</code>, and so on. The LDAP Authentication provider will search under each of these roots in turn until it finds a matching user.</p> <p>Note: You must supply a <code>user.filter.n</code> for each <code>user.dn.n</code> entry.</p>
User Search Scope	<p>Specifies how deep in the LDAP directory tree to search for users.</p> <p>Valid values are <code>subtree</code> and <code>onelevel</code>.</p>

Table 3-4 Attributes on the Users Tab

Attribute	Description
User from Name Filter	<p>An LDAP search filter for finding a user given the name of the user.</p> <p>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.</p> <p>Refer to the documentation for your LDAP server for more information about writing an LDAP search filter.</p>
All Users Filter	<p>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.</p> <p>Refer to the documentation for your LDAP server for more information about writing an LDAP search filter.</p>

3. To save your changes, click Apply.
4. Proceed to [“Locating Groups in the LDAP Directory” on page 3-18](#).

Locating Groups in the LDAP Directory

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

1. Click the Groups tab under the Configuration tab.
For example, click the Groups tab under the iPlanet Configuration 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 <code>subtree</code> and <code>onelevel</code> .
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 3-20](#).

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.

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

1. Click on the Membership tab under the Configuration tab.

For example, click the Membership tab under the iPlanet Configuration 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	<p>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.</p> <p>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.</p> <p>Refer to the documentation for your LDAP server for more information about writing an LDAP search filter.</p>
Dynamic Group Object Class	<p>The name of the LDAP object class that stores dynamic groups.</p> <p>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.</p>

Table 3-6 Attributes on the Membership Tab

Attribute	Definition
Dynamic Group Name Attribute	<p>The attribute of a dynamic LDAP group object that specifies the name of the group.</p> <p>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.</p>
Dynamic Member URL Attribute	<p>The attribute of the dynamic LDAP group object that specifies the URLs of the members of the dynamic group.</p> <p>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.</p> <p>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.</p>

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

Configuring Failover for LDAP Authentication Providers

In WebLogic Server 7.0 SP2 and greater, 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. Click the LDAP tab under the Configuration tab for the LDAP Authentication provider for which you want to configure failover.

For example, click the iPlanet LDAP tab under the iPlanet Configuration tab.

2. Click 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. Click the Details tab.
6. Set the Parallel Connect Delay attribute.

The Parallel Connect Delay attribute specifies the number of seconds to delay when making concurrent attempts to connect to multiple servers. An attempt is made to connect to the first server in the list. The next entry in the list is tried only if the attempt to connect to the current host fails. This setting might cause your application to block for 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.

The Connection Timeout attribute specifies 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

When the LDAP attribute values are set to the following:

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 this scenario, WebLogic Server will attempt to connect to `directory.knowledge.com`. After 10 seconds, the connect attempt will timeout and WebLogic Server will try the next host specified in the Host attribute (`people.catalog.com`). WebLogic Server will then use `people.catalog.com` as the LDAP Server for this connection.

Example 2

When the LDAP attribute values are set to the following:

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 this scenario, WebLogic Server will attempt to connect to `directory.knowledge.com`. After 1 second, the connect attempt will timeout and WebLogic Server will try the next host specified in the Host attribute (`people.catalog.com`) and will try to connect to `people.catalog.com` in parallel. WebLogic Server will then use `people.catalog.com` as the LDAP Server for this connection. WebLogic Server will cancel the connect to `directory.knowledge.com` after the connect 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 5, “Managing the Embedded LDAP Server.”](#)
1. Expand the Security-->Realms nodes.
2. Click 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. Define values for the attributes on the Details tab.

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.

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.
 8. Click Apply to save your changes.
 9. Optionally, configure additional Authentication and/or Identity Assertion providers.
 10. 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 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 9-3.

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 9-4.

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. Click 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 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.
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.

Enter X.509 in the Active Types 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 3-45](#).

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 [“Using a User Name Mapper with the WebLogic Identity Assertion Provider” on page 3-30](#).

To define attributes for the WebLogic Identity Assertion provider:

1. Expand the Security-->Realms nodes.
2. Click 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 Identity Asserter... link from the Authenticators tab.

The General tab appears.
5. Configure a user name mapper. For more information, see [“Using a User Name Mapper with the WebLogic Identity Assertion Provider”](#) on page 3-30.
6. In the Trusted Client Principals attribute define the list of client principals that can use CSIv2 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. Define the active token type for the WebLogic Identity Assertion provider. The list of token types supported by the Identity Assertion is displayed in the Supported Types attribute. Type the name of one of the supported token types in the Active Types attribute.
8. Click Apply to save your changes.
9. Click 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.

Using a User Name Mapper with the WebLogic Identity Assertion Provider

When using 2-way SSL, WebLogic Server verifies the digital certificate of the Web browser or Java client when establishing an 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 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. You can also use this interface to map from an X.501 distinguished name to a user name.

To use a user name mapper with the WebLogic Identity Assertion provider:

1. Expand the Security-->Realms nodes.
2. Click the name of the realm you are configuring (for example, TestRealm).
3. Expand the Providers-->Authentication Providers nodes.
4. Choose the Default Identity Assertion provider.

The General tab appears.

5. Enter the name of your implementation of the `weblogic.security.providers.authentication.UserNameMapper` interface in the User Name Mapper Class Name attribute.

The implementation of the `weblogic.security.providers.authentication.UserNameMapper` interface must be specified in your CLASSPATH.

6. Click Apply.
7. Reboot WebLogic Server.

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 its 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	<p>The token type this LDAP X509 Identity Assertion provider uses for authentication. This token type is always set to X509.</p> <p>Ensure no other identity assertion provider configured in the same security realm has this attribute set to X509.</p>

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	<p>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.</p> <p>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.</p> <p>For example: if the Certificate subject is <code>CN=meyer.beasys.com, ou=fred, o=BEASYS, L=SFO, C=US</code> and the mapping is <code>ou=people, ou=\$subj.ou</code>, WebLogic Server uses <code>ou=people, ou=fred, o=BEASYS, c=US</code> as the DN when locating the user.</p>

Table 3-8 Attributes on the Details Tab

Attribute	Definition
User Filter Attributes	<p>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.</p> <p>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.</p> <p>Typically, the value of this attribute is the LDAP object that matches the value of an attribute in the certificate's Subject DN.</p> <p>For example:</p> <p>The <code>uid</code> attribute of the LDAP user object matches the Subject DN attribute, if the syntax is:</p> <pre>LDAPATTRNAME=\$subj.SUBJECTDNATTRNAME</pre> <p>For example: <code>uid=\$subj.DN</code></p> <p>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:</p> <ul style="list-style-type: none">■ 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 <code>objectclass=person</code> to the filter and uses an array of strings that are combined.

Table 3-8 Attributes on the Details Tab

Attribute	Definition
Certificate Attribute	<p>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 <code>userCertificate</code> and <code>userCertificate;binary</code>. The default is <code>userCertificate</code>.</p> <ul style="list-style-type: none">■ If you use the LDAP browser to load a certificate into the LDAP directory, an attribute <code>userCertificate</code> of type binary is created. To access the certificate, define the Certificate attribute as <code>userCertificate</code>.■ If you use <code>ldapmodify</code> to create the new attribute (for example, using the following command):<pre>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...</pre>An attribute <code>userCertificate;binary</code> is created when the certificate data is loaded in the LDAP directory. To access the certificate, define the Certificate attribute as <code>userCertificate;binary</code>.
Username Attribute	<p>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.</p> <p>Typically, this attribute matches the User Name attribute of the LDAP Authentication provider.</p>

Table 3-8 Attributes on the Details Tab

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

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.

Ordering of Identity Assertion for Servlets

When an HTTP request is sent, there may be multiple matches that can be used for identity assertion. Identity assertion in WebLogic Server uses the following ordering:

1. An X.509 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.

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 3-45.

To configure a WebLogic Authorization provider:

1. Expand the Security-->Realms nodes.
2. Click 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.

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 4, “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 3-45](#).

To configure a WebLogic Credential Mapping provider:

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

A keystore is a mechanism designed to create and manage files that store private keys and trusted certificate authorities (CAs). The WebLogic Keystore provider locates instances of keystores. Configuring a WebLogic Keystore provider is one of the options you have when setting up the SSL protocol. For more information, see [“Storing Private Keys, Digital Certificates, and Trusted Certificate Authorities” on page 6-14](#). Configuring the WebLogic Keystore is an optional step when customizing a security realm or creating a new security realm.

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 3-45](#).

To configure an Role Mapping provider:

1. Expand the Security node.
2. Expand the Realms node.
3. Click the name of the realm you are configuring (for example, TestRealm).
4. Click the Providers node.
5. 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.

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

The General tab appears.

7. 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 [The Role Mapping Providers](#) in *Developing Security Services for WebLogic Server*.

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

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. Click on 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.

Deleting a Security Provider

To delete a security provider:

1. Expand the Security-->Realms nodes.
2. Click the name of the realm in which the provider you want to delete is configured (for example, TestRealm).
3. Expand the Providers node.
4. Click 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.

4 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 4-1](#)
- [“Using Deployment Descriptors to Create Credential Maps” on page 4-2](#)
- [“Using the WebLogic Administration Console to Create Credential Maps” on page 4-5](#)

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

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 and the WebLogic Server Administration Console. The following sections describe both techniques.

Using Deployment Descriptors to Create Credential Maps

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 login to the EIS and credentials used to authenticate to WebLogic resources. [Listing 4-1](#) contains a credential map specified in a `weblogic-ra.xml` deployment descriptor file.

Listing 4-1 Sample Credential Map

```
<security-principal-map>
  -<map-entry>
    <initiating-principal>raruser</initiating-principal>
    <initiating-principal>javajoe</initiating-principal>
    -<resource-principal>
      <resource-username>scott</resource-username>
      <resource-password>tiger</resource-password>
    </resource-principal>
  </map-entry>
```

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 4-5](#).

If you deploy a resource adapter that has a `weblogic-ra.xml` deployment descriptor file containing a defined `<security-principal-map>` element, the data in this file can be imported into the embedded LDAP server and then used by the WebLogic Credential Mapping provider.

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.

To avoid overwriting new credential mapping information with old information in a `weblogic-ra.xml` deployment descriptor file, set the Ignore Deploy Credential Mapping Deployment Descriptor setting:

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

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

3. Click the name of the realm you are using.
4. Click the General tab.
5. Check the Ignore Deploy Credential Mapping Deployment Descriptor setting. This setting 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.
6. Click Apply.
7. Reboot WebLogic Server.

BEA Systems also recommends modifying the `weblogic-ra.xml` deployment descriptor file to remove the `<security-principal-map>` element.

For more information about using the `weblogic-ra.xml` deployment descriptor file to specify a credential map, see [Programming WebLogic J2EE Connectors](#).

Using the WebLogic Administration Console to Create Credential Maps

The mapping between credentials can now be done through the WebLogic Server Administration Console. 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 Security Data in Deployment Descriptors 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.

2. Define a user or group for the EIS user. For more information, see Users and Groups in [Securing WebLogic Resources](#).

3. Expand the Connectors node.

4. Right-click on the desired resource adapter.

5. Click the Define Cred Map option.

The Credential Maps table displays all the credential maps defined in the configured Credential Mapper.

6. Click the Configure a New Cred Map... link.

7. Enter the username of the EIS user in the Remote User Cred Map field. For example, scott.

8. Enter the password for the EIS user in the Remote Password field. For example, tiger.

9. Click Apply.

10. Right-click on the desired resource adapter.

11. Click the Define Role Map option.

12. Enter the WebLogic Server user or group name you defined for the EIS user in step 2 in the WLS User field.

4 *Single Sign-On with Enterprise Information Systems*

13. Enter the name of the EIS user in Remote User field.
14. Click Apply.

5 Managing the Embedded LDAP Server

By default, the embedded LDAP server is used as the security provider database for the WebLogic Authentication, Authorization, Credential Mapping and Role Mapping providers. If any of these providers are used, the embedded LDAP server needs to be managed. The following sections explain how to manage the embedded LDAP server.

- [“Configuring the Embedded LDAP Server” on page 5-1](#)
- [“Configuring Backups for the Embedded LDAP Server” on page 5-4](#)
- [“Viewing the Contents of the Embedded LDAP Server from an LDAP Browser” on page 5-5](#)
- [“Exporting and Importing Information in the Embedded LDAP Server” on page 5-6](#)
- [“Access Control Syntax” on page 5-7](#)

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.

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 5-1 Attributes on the Embedded LDAP Tab

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 <code>config.xml</code> file. If you want to connect to the embedded LDAP server using an external LDAP browser and the embedded LDAP administrator account (<code>cn=Admin</code>), change this attribute from the generated value.
Backup Hour	The hour at which to backup 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 <code>ldap/backup</code> directory, and then resumes writes. The default is 23.
Backup Minute	The minute at which to backup 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 5-1 Attributes on the Embedded LDAP Tab

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 <code>ldap/backup</code> 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	<p>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.</p> <p>Use this attribute to propagate a new system password to the Managed and Administration servers in a domain.</p> <p>The default is false.</p>
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.

4. Click Apply to save your changes.

5. Reboot WebLogic Server.

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. Click the Security-->the 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. 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. For more information, see [Table 5-1](#).
 - d. Click Apply.
 - e. Reboot WebLogic Server.
2. Enter the following command at a command prompt to start the LDAP browser:
`lbe.sh`
3. Configure a new connection in the LDAP browser:
 - a. Set the host field to `localhost`.
 - b. Set the port field to `7001` (`7002` if SSL is being used).
 - c. Set the Base DN field to `dc=mydomain` where `mydomain` is the name of the WebLogic Server domain you are using.
 - d. Uncheck the Anonymous Bind option.
 - e. Set the User DN field to `cn=Admin`.
 - f. Set the Password field to the password you specified in Step 1.
4. 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

An LDAP browser can be used to export and import data stored in the embedded LDAP Server. [Table 5-2](#) summarizes where data is stored in the hierarchy of the embedded LDAP server.

Table 5-2 Location of Security Data in the Embedded LDAP Server

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

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.

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.

Access Control Syntax

The embedded LDAP server supports the IETF LDAP Access Control Model for LDAPv3 (March 2, 2001 draft). This section describes how those rules are 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`).

The Access Control File

Note: The default behavior of the embedded LDAP server is to only allow access from the Admin account in WebLogic Server. If you are planning only to use an external LDAP browser, you do not need to edit the `acls.prop` 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:

- The location in the LDAP directory where the rule applies
- The scope within that location to which the rule applies

- The access rights (either grant or deny)
- The permissions (either grant or deny)
- The attributes to which the rule applies
- The subject being granted or denied access

[Listing 5-1](#) shows a sample access control file.

Listing 5-1 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 and Permissions

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 permissions are discrete. One permission does not imply another permission. The permissions specify the type of LDAP operations that can be performed.

Attribute Permissions

The following permissions apply to actions involving attributes.

Table 5-3 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.

Table 5-3 Attribute Permissions

Permission	Description
<code>o</code> Obliterate	Modify and delete attributes. If granted, permits attributes and values to be deleted in a Modify operation.
<code>s</code> Search	Search entries with specified attributes. If granted, permits attributes and values to be included in a Search operation.
<code>c</code> Compare	Compare attribute values. If granted, permits attributes and values to be included in a Compare operation.
<code>m</code> Make	Make attributes on a new 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 5-4 Entry Permissions

Permission	Description
a Add	Add an entry below this 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	<p>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.</p> <p>If the last RDN is changed, <code>Rename</code> permission is also required at the current location.</p> <p>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 of RDN.</p>

Table 5-4 Entry Permissions

Permission	Description
i Import	<p>Import entry and subordinates from specified location.</p> <p>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).</p> <p>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.</p>
n RenameDN	<p>Change the DN of an LDAP entry. Granting the <code>Rename</code> 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.</p> <p>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.</p>
b BrowseDN	<p>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.</p>
t ReturnDN	<p>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.</p>

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 a 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).
- 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.

If a conflict still exists after looking at 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. Final priority is given to rules that apply to **Subtree** and **Public** subjects.

6 Configuring SSL

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

- [“SSL: An Introduction” on page 6-2](#)
- [“Private Keys, Digital Certificates and Trusted Certificate Authorities” on page 6-3](#)
- [“One-Way and Two-Way SSL” on page 6-3](#)
- [“Setting Up SSL: Main Steps” on page 6-4](#)
- [“Obtaining Private Keys, Digital Certificates and Trusted Certificate Authorities” on page 6-5](#)
- [“Storing Private Keys, Digital Certificates, and Trusted Certificate Authorities” on page 6-14](#)
- [“Enabling the SSL Port” on page 6-21](#)
- [“Setting Attributes for One-Way SSL” on page 6-21](#)
- [“Setting Attributes for Two-Way SSL” on page 6-23](#)
- [“Command-Line Arguments for SSL” on page 6-24](#)
- [“Enabling SSL Debugging” on page 6-25](#)
- [“SSL Session Behavior” on page 6-26](#)
- [“Using Host Name Verification” on page 6-27](#)
- [“Configuring RMI over IIOP with SSL” on page 6-44](#)
- [“SSL Certificate Validation” on page 6-45](#)
- [“Using the nCipher JCE Provider with WebLogic Server” on page 6-48](#)

- “Specifying the Version of the SSL Protocol” on page 6-50
- “Using the SSL Protocol to Connect to WebLogic Server from weblogic.Admin” on page 6-51
- “Using the SSL Protocol with a BEA Tuxedo Client and WebLogic Server” on page 6-54

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 TLS 1.0 specifications.

Note: The proxy plug-ins in this release of WebLogic Server only support SSL 3.0.

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 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 visa 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 (also referred to as trusted certificate authority) and digitally signed with the certificate authority's digital certificate. Well-know certificate authorities include Verisign and Entrust.net. A trusted certificate authority establishes *trust* for a server

An application participating in an SSL connection is authenticated when the other party evaluates and accepts their 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 host name 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.
- With two-way SSL, the client also 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 a identity (private key and digital certificates) and trust (certificates for 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, the Certificate Request Generator servlet, 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. Digital certificates are always stored in file in the domain directory of WebLogic Server. Private keys and trusted CA certificates can either be stored in a keystore, a keystore accessed by the WebLogic Keystore provider, or in a file in the domain directory.
3. Enable the SSL port. By default, the SSL port is not enabled because of the performance impact of its use.
4. Set SSL attributes in the WebLogic Server Administration Console or in the server start script. The SSL attributes define the location of the private key, digital certificate, and trusted CA certificates. 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 6-5](#)

[“Storing Private Keys, Digital Certificates, and Trusted Certificate Authorities” on page 6-14](#)

[“Enabling the SSL Port” on page 6-21](#)

[“Setting Attributes for One-Way SSL” on page 6-21](#)

[“Setting Attributes for Two-Way SSL” on page 6-23](#)

Obtaining Private Keys, Digital Certificates and Trusted Certificate Authorities

To use SSL, you need a private key, a digital certificate containing the matching public key, and at least one trusted certificate authority that can verify the data embedded in the digital certificate. WebLogic Server supports private keys, digital certificates, and trusted certificate authorities from the following sources:

- The demonstration digital certificates, private keys, and trusted certificate authorities in the `WL_HOME\server\lib` directory.

When you create a domain through the Configuration Wizard, the demonstration certificates and private keys are copied into the domain directory. The demonstration digital certificates, private keys, and trusted certificate authorities should only be used for demonstration or testing purposes and not in a production environment

- The Cert Gen utility generates digital certificate and private keys that should only be used 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. For more information about using the Cert Gen utility to obtain private keys and digital certificates, see [“Using the Cert Gen Utility” on page 6-7](#).

- The Certificate Request Generator servlet generates a private key file and a Certificate Signature Request (CSR). To acquire a digital certificate from a trusted certificate authority (such as VeriSign or Entrust.net), you must submit your request in a particular format called a CSR. The digital certificate you receive from the certificate authority and the private key from the Certificate Request Generator servlet are suitable for production environments. For more information, see [“Using the Certificate Request Generator Servlet” on page 6-8](#).
- Sun Microsystems’s `keytool` utility can also be used to generate a private key, a self-signed digital certificate for WebLogic Server, and a 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. For more information about Sun's `keytool` utility, see [keytool—Key and Certificate Management Tool](#).

WebLogic Server can use digital certificates in either `.pem` or `.der` format.

A `.pem` (privacy-enhanced mail) 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). However, the ordering of the digital certificates in the file is important. For example, cert A, cert B (must be the issuer of cert A), cert C (must be the issuer of cert B),... to the root CA.

A `.der` format file contains binary data. A `.der` file can be used only for a single certificate, however, 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 6-12](#).

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

Using the Cert Gen Utility

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

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

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

To generate a certificate:

1. Copy the following files to the directory in which you run the CertGen utility:

- WL_HOME\server\lib\CertGenCA.der—The digital certificate for a certificate authority trusted by WebLogic Server.
- WL_HOME\server\lib\CertGenCAKey.der—The private key for a certificate authority trusted by WebLogic Server.

2. Enter the following command at a command prompt:

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

where

- *password* is the password for the private key.
- *certfile* is the name of the digital certificate file. The file is put in the current directory.
- *keyfile* is the name of the generated private key file. The file is put in the domain directory.
- *hostname* is the name of the machine for which you are obtaining a digital certificate. This option allows you to use host name verification.

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 `[export]` as `""`.

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 the Certificate Request Generator Servlet

Before using a WebLogic Server deployment in a production environment, you need to obtain a private key and certificate from a trusted certificate authority such as VeriSign or Entrust.net. To acquire a digital certificate from a certificate authority, you must submit your request in a particular format called a CSR. The Certificate Request Generator servlet collects information from you and generates a private key file and a CSR. You then submit the CSR to a certificate authority.

To generate a CSR, perform the following steps:

1. Copy the `certificate.war` file to the applications directory (copy the file before the server boots or while the server is running). The Configuration Wizard performs this step for you.

2. In a Web browser, enter the URL for the Certificate Request Generator servlet as follows:

```
http (or https)://hostname:port/certificate/
```

The components of this URL are defined as follows:

- `hostname` is the DNS name of the machine running WebLogic Server.
- `port` is the number of the port at which WebLogic Server listens for SSL connections. The default is 7002. 7002 is the default port on which WebLogic Server listens for SSL communications. Any port on which WebLogic Server listens for communications can be specified.

For example, if WebLogic Server is running on a machine named `ogre` and it is configured to listen for SSL communications at the default port 7002 to run the Certificate Request Generator servlet, you must enter the following URL in your Web browser:

```
https://ogre:7002/certificate/
```

3. The Certificate Request Generator servlet loads a form in your Web browser. Complete the form displayed in your browser, using the information in the following table:

Table 6-1 Fields on the Certificate Request Generator Form

Field	Description
Country code	Two-letter ISO code for your country. The code for the United States is US.
Organizational unit name	Name of your division, department, or other operational unit of your organization.
Organization name	Name of your organization. The certificate authority may require any host names entered in this attribute belong to a domain registered to this organization.
E-mail address	E-mail address of the administrator. The digital certificate is mailed to this e-mail address.

Table 6-1 Fields on the Certificate Request Generator Form

Field	Description
Full host name	Fully qualified name of the WebLogic Server on which the digital certificate will be installed. This name is the one used for DNS lookups of the WebLogic Server, for example, <code>node.com</code> . Web browsers compare the host name in the URL to the name in the digital certificate. If you change the host name later, you must request a new digital certificate.
Locality name (city)	Name of your city or town. If you operate with a license granted by a city, this attribute is required; you must enter the name of the city that granted your license.
State name	Name of the state or province in which your organization operates if your organization is in the United States or Canada, respectively. Do not abbreviate.
Private Key Password	The password used to encrypt the private key. Enter a password in this field to use a protected key with WebLogic Server. You are prompted for the password whenever the key is used. The servlet creates a PKCS-8 encrypted private key.
Strength	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. If you have the domestic version of WebLogic Server, choose 512-, 1024-, or 2048-bit keys. The 1024-bit key is recommended.

4. Click Generate Request.

The Certificate Request Generator servlet displays messages informing you if any required attributes are empty or if any attributes contain invalid values. Click Back in your Web browser and correct any errors.

When all attributes have been accepted, the Certificate Request Generator servlet generates the following files in the start directory of your WebLogic Server:

- `hostname-key.der`—The private key file.
- `hostname-request.der`—The certificate request file, in binary format.

- `hostname-request.pem`—The CSR file that you submit to the certificate authority. It contains the same data as the `.dem` file but is encoded in ASCII so that it can be copied into e-mail or pasted it into a Web form.
5. Select a certificate authority and follow the instructions on that authority's Web site to purchase a digital certificate.
 - [VeriSign, Inc.](#) offers two options for WebLogic Server: Global Site Services, which features strong 128-bit encryption for domestic and export Web browsers, and Secure Site Services, which offers 128-bit encryption for domestic Web browsers and 40-bit encryption for export Web browsers.
 - [Entrust.net](#) certificates offer 128-bit encryption for domestic browser versions and 40-bit encryption for export browser versions.

Using Certificate Chains

WebLogic Server supports the use of certificate chains.

To use a 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](#) format. If you are using a digital certificate issued by Microsoft, see “[Converting a Microsoft p7b Format to PEM Format](#)” on page 6-12. You can use the steps in this section to convert other types of digital certificates. You just need to 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 (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. 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 attribute on the SSL Attributes tab in the WebLogic Server Administration Console.

[Listing 6-1](#) shows a sample certificate chain.

Listing 6-1 Sample File with Certificate Chain

[illegible]

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, click on 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 certificate authorities with WebLogic Server:

1. Ensure the trusted CA certificates are in PEM format. If they are in DER format, you can convert them using the [utils.der2pem](#) format. If you are using a digital certificate issued by Microsoft, see [“Converting a Microsoft p7b Format to PEM Format” on page 6-12](#).
2. Either store the trusted CA certificate in a file or in a JKS keystore accessed by the WebLogic Keystore provider. For more information, see [“Storing Private Keys, Digital Certificates, and Trusted Certificate Authorities” on page 6-14](#).
3. Configure WebLogic Server to use the trusted certificate authority. For more information, see [“Setting Attributes for One-Way SSL” on page 6-21](#).

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.

You should note that digital certificates are not universally useful for multiple browsers, or even for different versions of the same browser.

Storing Private Keys, Digital Certificates, and Trusted Certificate Authorities

Once you have obtained private keys, digital certificates, and trusted certificate authorities, you need to store them so that WebLogic Server can use them to verify identity. Digital certificates can only be stored in a file. Private keys can either be stored in a file or in a keystore accessed via the WebLogic Keystore provider. Trusted CA certificates can be stored in a file or in a JKS keystore. The keystore can be accessed via the WebLogic Keystore provider or specified on the command-line.

- If you store the private keys, digital certificates, and trusted CA certificates in files, you need to set the SSL Server Key File Name, Server Certificate File Name, and Trusted CA File Name attributes in the WebLogic Server

Administration Console. For information about setting SSL attributes, see [“Setting Attributes for One-Way SSL” on page 6-21](#).

- If you store the private keys and trusted CA certificates in a keystore, you need to create a keystore and load the private keys and trusted CA certificates into the keystore. This release of WebLogic Server only supports JKS keystores.

For private keys, you must configure the WebLogic Keystore provider to point to the keystore, and set the SSL Server Private Key Alias and Server Private Key Passphrase attributes in the WebLogic Server Administration Console.

For trusted certificate authorities, you can either configure the WebLogic Keystore provider to point to the keystore, and set the Trusted CA File Name attribute in the WebLogic Server Administration Console or use the `-Dweblogic.security.SSL.trustedCAkeystore` command-line argument in the server start script to specify the keystore.

For more information, see:

- [“Creating a Keystore and Loading Private Keys and Trusted Certificate Authorities into the Keystore” on page 6-15](#)
- [“Configuring the WebLogic Server Keystore Provider to Locate a Keystore” on page 6-18](#)
- [“Setting Attributes for One-Way SSL” on page 6-21](#)

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 certificates, and trusted CA certificates. This release of WebLogic Server only supports JKS keystores. The WebLogic Keystore provider locates instances of keystores. Use the following utilities 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 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 6-17](#).

Note: The `keytool` utility requires a digital certificate for every private key when importing a private key into the keystore. This digital certificate is not used by WebLogic Server.

While you can use the `keytool` utility to load new private keys and trusted CA certificates into 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.

Either create one keystore for both private keys and trusted CA certificates or two keystores: one for private keys and one for trusted CA certificates. Although one keystore can be used for both identity and trust, for the following reasons, BEA recommends using separate keystores for both identity and trust:

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

- The Identity keystore may be prohibited by company policy from ever being put in the network while the Trust keystore can be distributed over the network.
- The Identity keystore may be protected by the operating system for both reading and writing by non-authorized users while the Trust keystore only needs to be write protected.
- The password for the trust keystore is generally known by more people than the password for the Identity keystore.

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

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

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

By default, WebLogic Server looks for a private key keystore named `wlDefaultKeyStore.jks` in the domain directory. The keystore for trusted CA certificates can use either the demonstration trusted certificate authorities (`WL_HOME\server\lib\cacerts`) or a trusted certificate authority available from the JDK (`...\jre\lib\security\cacerts`).

Note: The `WL_HOME\server\lib\cacerts` file should be used for demonstration or testing purposes and not in a production environment. This file contains the trusted certificate authorities in `\jre\lib\security\cacerts`.

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.

Trusted certificate authorities are not individually identified to WebLogic Server with aliases. 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.

Aliases are case-insensitive; the aliases `Hugo` and `hugo` would refer to the same keystore entry. Aliases for private keys are specified in the Server Private Key Alias attribute when configuring SSL.

Common Keytool Commands

[Table 6-2](#) the `keytool` commands when creating and using JKS keystores with WebLogic Server.

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

Table 6-2 Commonly Used keytool Commands

Command	Description
<code>keytool -genkey -keystore keystorename -storepass keystorepassword</code>	Creates a keystore.
<code>keytool -alias aliasforprivatkey -import -file privatekeyfile.pem -keypass privatekeypassword -keystore keystorename -storepass keystorepassword</code>	Loads a private key into the keystore.
<code>keytool -alias aliasfortrustedca -trustcacerts -import -file trustedcafilename.pem -keystore keystorename -storepass keystorepassword</code>	Loads a trusted CA certificate into the keystore.
<code>keytool -list -keystore keystorename</code>	Displays what is in the keystore
<code>keytool -keystore keystorename -storepass keystorepassword -delete -alias privatekeyalias</code>	Delete a private key for the specified alias from the keystore.
<code>keytool -help</code>	Provides online help for keytool

Configuring the WebLogic Server Keystore Provider to Locate a Keystore

When using the WebLogic Keystore provider, the `config.xml` file that contains the password for the private key is hashed. This provides an additional level of security not available when storing private keys and trusted CA certificates in files.

Configuring a WebLogic Keystore provider is an optional step. If private keys and digital certificates are stored in files, you do not need to configure the WebLogic Keystore provider. You can also use a JKS keystore without using a WebLogic Keystore provider. For more information, see [“Using a JKS Keystore” on page 6-20](#).

To configure the WebLogic Keystore provider, you need to create a keystore and set attributes in the WebLogic Server Administration Console that point to the keystore.

Note: Weblogic Keystore providers are configured on a domain basis. However, keystores are configured on a per machine basis. If you have multiple security realms or multiple servers running on the same machine, they can all use the same keystore.

To configure the WebLogic Keystore provider:

1. Create a keystore. For more information, see [“Creating a Keystore and Loading Private Keys and Trusted Certificate Authorities into the Keystore” on page 6-15](#).
2. In the WebLogic Server Administration Console, expand the Security-->Realms nodes.
3. Click the name of the realm you are configuring (for example, *myrealm*).
4. Expand the Providers node.
5. Click Key Stores.

The Keystore tab appears. This tab displays the name keystore configured for the security realm. By default, the WebLogic Keystore provider is configured.

Note: The WebLogic Server Administration Console refers to the WebLogic Keystore provider as the DefaultKeystore.

6. Click DefaultKeystore.
7. On the General tab, enter the directory location of the keystore in the Private Key Store Location attribute. The default is `wlDefaultKeyStore.jks`.

This attribute requires both a directory and filename location that is either absolute or relative to root directory (that is, the domain directory) of the server.

Warning: On all supported operating systems, the keystore has to be located in the same place for all servers in the domain. Otherwise, WebLogic Server will not locate the keystore.

8. Enter the directory location of the keystore that contains trusted CA certificates in the Root CA Key Store Location attribute.

This attribute requires both a directory and filename location that is either absolute or relative to root directory (that is, the domain directory) of the server. This step is not required if both private keys and trusted CA certificates are stored in the same keystore.

Leave the Root CA Key Store Location attribute blank, if you plan to use the `-Dweblogic.security.SSL.trustedCAKeyStore` command line argument to specify the location of a keystore that contains certificate authorities that the server trusts.

9. Enter the password you specified when you created in the trusted CA keystore in the Private Keystore Pass Phrase attribute.
10. Click Apply to save your changes.
11. Reboot WebLogic Server.

Using a JKS Keystore

If you don't want to use the WebLogic Keystore provider to store trusted CA certificates and private keys, you have the option of specifying a JKS keystore location as a command line argument when starting WebLogic Server.

Use the `-Dweblogic.security.SSL.trustedCAKeyStore` command line argument to specify the location of a keystore that contains certificate authorities that the server or client trusts. The path value must be an absolute or relative path to the directory from where the server is booted.

Weblogic Server looks for trusted certificate authorities in the following sequence:

- Uses the JKS keystore specified in the `-Dweblogic.security.SSL.trustedCAkeystore` command-line argument.
- Uses the keystore specified in the Root CA Key Store Location attribute for the WebLogic Keystore provider.
- Uses a trusted CA certificate file from a `6.x config.xml` file (if one exists in the domain directory).

- Uses `WL_HOME\server\lib\cacerts`.

Enabling the SSL Port

This procedure explains how to explicitly enable the use of SSL between an individual server and SSL clients. Internally, WebLogic Server uses SSL to protect communications between the Administration Server, Managed Servers, and the Node Manager, WebLogic Server and the LDAP servers used by the LDAP Authentication providers, WebLogic Server and SSL client code, and communications through the Administration Port.

To enable the SSL port, perform the following steps:

1. Expand the Server node.
2. Select the Connections-->SSL Ports tab and set the following attributes
 - The Enabled attribute enables the use of SSL.
 - The SSL Listen Port attribute is the number of the dedicated port on which WebLogic Server listens for SSL connections. The default is 7002.
3. Click Apply to save your changes.
4. Set attributes for One-way or Two-way SSL. For more information, see [“Setting Attributes for One-Way SSL” on page 6-21](#) or [“Setting Attributes for Two-Way SSL” on page 6-23](#).
5. Reboot WebLogic Server.

Setting Attributes for One-Way SSL

To define attributes for one-way SSL:

1. Enable the SSL port. For more information, see [“Enabling the SSL Port” on page 6-21](#).

2. Select the Connections-->SSL tab and set the attributes described in [Table 6-3](#).

Table 6-3 SSL Attributes

SSL Attribute	Description
Server Private Key Alias	<p>The alias specified when loading the private key for WebLogic Server into the keystore. Define this attribute only if you stored the private key for WebLogic Server in a keystore. You must have the WebLogic Keystore provider configured to use this attribute.</p> <p>All private key keystore entries are accessed via unique aliases. You specify the alias when loading the private key into the keystore. Aliases are case-insensitive; the aliases <code>Hugo</code> and <code>hugo</code> would refer to the same keystore entry.</p>
Server Private Key Passphrase	<p>The password specified when loading the private key for WebLogic Server into the keystore. Define this attribute only if you stored the private key for WebLogic Server in a keystore. You must have the WebLogic Keystore provider configured to use this attribute.</p>
Server Certificate File Name	<p>The directory location of the digital certificate for WebLogic Server.</p> <p>If you are using a certificate chain that is deeper than two certificates, you need to include the entire chain in PEM format in the certificate file.</p>
Server Key File Name	<p>The directory location of the private key for WebLogic Server. Specify this attribute only if you stored the private key for WebLogic Server in a file.</p> <p>If you protected the private key file with a password, specify the <code>weblogic.management.pkpassword</code> command-line argument when starting the server.</p>

Table 6-3 SSL Attributes

SSL Attribute	Description
Trusted CA File Name	The name of the file containing the PEM-encoded trusted certificate authorities.

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

Setting Attributes for Two-Way SSL

Two-way SSL requires clients to present a digital certificate to WebLogic Server in order to establish an SSL connection. The client first authenticates the server's digital certificate and then the server authenticates the client's digital certificate.

Before using two-way SSL both the SSL client and server need identity. In addition, the client and server must trust the issuer of each other's certificate.

To define attributes for two-way SSL:

1. Enable SSL.
2. Select the Connections-->SSL tab and set attributes for one-way SSL as described in Step 3 of [“Setting Attributes for One-Way SSL” on page 6-21](#).
3. Enable one of the following two-way SSL attributes on the SSL tab:
 - Check the Client Certificate Enforced attribute to require a client to present a certificate. If a valid and trusted certificate chain is not presented, the SSL connection is terminated.
 - Check the Client Certificate Requested But Not Enforced attribute to ask for a certificate from the client. If this option is enabled, the connection continues if the client does not present a certificate.
4. Click Apply.
5. Reboot WebLogic Server.

Command-Line Arguments for SSL

To use a private key stored in a file with WebLogic Server, use the following command-line argument:

```
-Dweblogic.management.pkpassword=pkpassword
```

where *password* is the password for the private key.

Note: Using this command-line argument presents a security risk as the password is in clear text.

To specify a trusted CA keystore other than the keystore specified in the Root CA Key Store Location attribute in the WebLogic Keystore provider, use the following command line argument:

```
-Dweblogic.security.SSL.trustedCAkeystore=path
```

The *path* value must be a relative to where the server is booted or qualified name to a JKS keystore file. This command-line argument overrides information specified in the WebLogic Server Administration Console. For information about how WebLogic Server obtains trust, see [“Using a JKS Keystore” on page 6-20](#).

To control Basic Constraints certificate validation on WebLogic Server, use the following command-line argument:

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

The default is `strong`. For more information, see [“SSL Certificate Validation” on page 6-45](#).

Note: The `weblogic.security.SSL.sessionCache.size` and `weblogic.security.SSL.sessionCache.ttl` command-line arguments are not supported in this release of WebLogic Server. For more information, see [“SSL Session Behavior” on page 6-26](#).

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 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 standard 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 may 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, check if 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 change in this release of WebLogic Server. The `weblogic.security.SSL.sessionCache.size` and `weblogic.security.SSL.sessionCache.ttl` command-line arguments are 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.HttpURLConnection` 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 `setEnabledSessionCreation(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 `setEnabledSessionCreation` 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, there is one SSL session 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.

Using Host Name Verification

A host name verifier ensures the host name 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 a 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, as a function of the SSL handshake, compares the common name in the SubjectDN in the SSL server's digital certificate with the host name of the SSL server used to initiate the SSL connection. If these names do not match, the SSL connection is dropped. The SSL client is the actual party that drops the SSL connection if the names do not match.

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

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

- In the WebLogic Server Administration Console, check the Hostname Verification Ignored attribute on the SSL tab on the Server node.
- 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.

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.

To use a custom host name verifier:

1. Expand the Server node.
2. Enter the name of the Java class used to load your custom host name verifier in the Hostname Verifier attribute on the SSL tab.
3. Click Apply.
4. Reboot WebLogic Server.

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

```
-D weblogic.security.SSL.hostnameVerifier=classname
```

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

Configuring SSL for the Node Manager

The Node Manager uses 2-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 host name verification and the Administration port must be taken

into consideration. This section describes two scenarios: using the demonstration identity and trust provided by WebLogic Server and using production-quality identity and trust.

For more information, see:

[“SSL Requirements for Administration Servers” on page 6-29](#)

[“SSL Requirements for Managed Servers” on page 6-31](#)

[“SSL Requirements for the Node Manager” on page 6-32](#)

[“Identity and Trust: Demonstration Versus Production” on page 6-33](#)

[“Host Name Verification Requirements” on page 6-34](#)

[“Node Manager SSL Production Configuration: Main Steps” on page 6-37](#)

[“Configuring the Administration Server to Use SSL” on page 6-39](#)

[“Configuring a Managed Server to Use SSL” on page 6-40](#)

[“Configuring the Node Manager to Use SSL” on page 6-42](#)

Note: Although WebLogic Server offers different techniques for storing and specifying identity and trust, this section describes the recommended use case.

SSL Requirements for Administration Servers

To use SSL, Administration Servers require:

- Identity—Administration Servers use private keys stored in JKS keystores accessed through the WebLogic Keystore provider.

The digital certificate for the Administration Server must be stored in a file. Digital certificates must also specify a host name rather than an IP address. The location of the digital certificate is specified in the WebLogic Server Administration Console.

By default, the demonstration digital certificate and private key available in the WLS Domain template are copied to a domain by the Configuration Wizard when the domain is created. The Administration Server uses this digital

certificate and private key for identity. However, the demonstration digital certificate and private key should not be used in production.

- **Trust**—Administration Servers can use trusted CA certificates stored in a JKS keystore. The JKS keystore is specified by the `-Dweblogic.security.SSL.trustedCAkeystore` command-line argument in the start script for the Administration Server.

Note: By default, the start script for the Administration Server specifies trust through the `-Dweblogic.security.SSL.trustedCAkeystore` command-line argument so any trust settings specified through the WebLogic Server Administration Server are ignored.

By default, the Administration Server trusts all the certificate authorities in the `cacerts` files in the `WL_HOME\server\lib`.

In a production environment, the Administration Server needs to trust the certificate authorities for both the Node Manager and any Managed Servers (meaning those trusted CA certificates must be in the trust keystore of the Administration Server).

- **SSL port**—By default, the SSL port is enabled on the Administration Server.
- **Host name verification**—By default, host name verification is not enabled on Administration Servers. Host name verification is only necessary in a production environment.

Host name verification is specified by the `-Dweblogic.security.SSL.ignoreHostnameVerification` command-line argument in the start script for the Administration Server

- **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. If you use the Administration port, the Administration Server and all Managed Servers must be configured to use the Administration port. Managed Servers that are not configured to use the Administration port will be unable to connect to the Administration Server during start.

SSL Requirements for Managed Servers

To use SSL, Managed Servers require:

- **Identity**—Managed Servers use private keys stored in keystores accessed through the WebLogic Keystore provider.

The digital certificate for the Administration Server must be stored in a file. Digital certificates must also specify a host name rather than an IP address. The location of the digital certificate is specified in the WebLogic Server Administration Console.

By default, Managed Servers do not have any configured identity. The demonstration digital certificate and private key provided by the Configuration Wizard can be used for identity for Managed Servers. However, this digital certificate and private key should not be used in a production environment.

- **Trust**—Managed Servers use trusted CA certificates stored in a JKS keystore. The JKS keystore is specified by the

`weblogic.security.SSL.trustedCAkeystore` command-line argument in the remote start arguments for the Managed Server.

By default, Managed Servers trusts all the certificate authorities in the `cacerts` files in the `WL_HOME\server\lib`.

In a production environment, a Managed Server needs to trust the certificate authorities for both the Node Manager and the Administration Server (meaning those trusted CA certificates must be in the trust keystore of the Managed Server).

- **Host name verification**—By default, host name verification is not enabled on Managed Servers. Host name verification is only necessary in a production environment.

Host name verification is specified by the `-Dweblogic.security.SSL.ignoreHostnameVerification` command-line argument in the remote start arguments for the Managed Server

- **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. If you use the Administration port, the Administration Server and all Managed Servers must be configured to use the Administration port. Managed Servers that are not configured to use the Administration port will be unable to connect to the Administration Server during start.

SSL Requirements for the Node Manager

Note: BEA recommends not running the Node Manager as a NT service when using SSL.

To use SSL, the Node Manager requires:

- **Identity**—The Node Manager can only use digital certificates and private keys stored in files. Identity is specified in the start script for the Node Manager by the following command-line arguments:

```
weblogic.nodemanager.keyFile=filename
```

```
weblogic.nodemanager.keyPassword=password_for_the_private_key
```

```
weblogic.nodemanager.certificateFile=filename
```

By default, the WebLogic Server installer copies a demonstration digital certificate and private key to the `WL_HOME\common\nodemanager\config` directory. However, this digital certificate and private key should not be used in a production environment.

- **Trust**—The Node Manager uses trusted CA certificates stored in a JKS keystore for trust. The trust keystore is specified in the start script for the Node Manager by the following command-line argument:

```
weblogic.security.SSL.trustedCAkeystore=keystorename
```

Because 2-way SSL is used, the Node Manager must also trust the certificate authorities used by the Administration Server and any Managed Servers (meaning those trusted CA certificates must be in the trust keystore of the Node Manager).

By default, the Node Manager trusts all the certificate authorities in the `cacerts` files in the `WL_HOME\server\lib`.

- **Host name verification**—By default, host name verification is not enabled on the Node Manager. Host name verification is only necessary in a production

environment. Host name verification is enabled in the start script for the Node Manager by the following command-line argument:

```
weblogic.security.SSL.ignoreHostnameVerification=false
```

- **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 for the Administration Servers for any domain you want to run from this machine. By default, the `nodemanager.hosts` file always defaults to `localhost`.

The hosts may be specified by IP address or name. If you want to use host names, enable Reverse DNS on the Node Manager.

Warning: Using host names may make the WebLogic Server deployment vulnerable to man-in-the-middle attacks.

For more information, see [Set Up the Node Manager Hosts File](#).

Identity and Trust: Demonstration Versus Production

WebLogic Server provides demonstration identity (private keys and digital certificates) and trust (trusted certificate authorities) that can be used by Administration and Managed Servers and the Node Manager

- When the Configuration Wizard creates a domain, the wizard copies the demonstration trust and identity in the WLS Domain template into the domain for use by the Administration Server. These demonstration private keys and digital certificates are suitable for testing or demonstration purposes, however, they should not be used in a production environment.
- Managed servers do not have identity and trust configured by default. However, they can also use the identity and trust provided by the Configuration wizard.
- The installer provides demonstration private keys and digital certificates for the Node Manager. These demonstration private keys and digital certificates are

suitable for testing or demonstration purposes, however, they should not be used in a production environment.

In a production environment, obtain private keys and digital certificates for the Node Manager, Administration Servers, and Managed Servers 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 6-5.

Host Name Verification Requirements

Running with host name verification disabled is fine in a demonstration environment. However, when moving to a production environment, BEA recommends using host name verification to ensure the host you are connecting to is the intended host. To use host name verification, check the following:

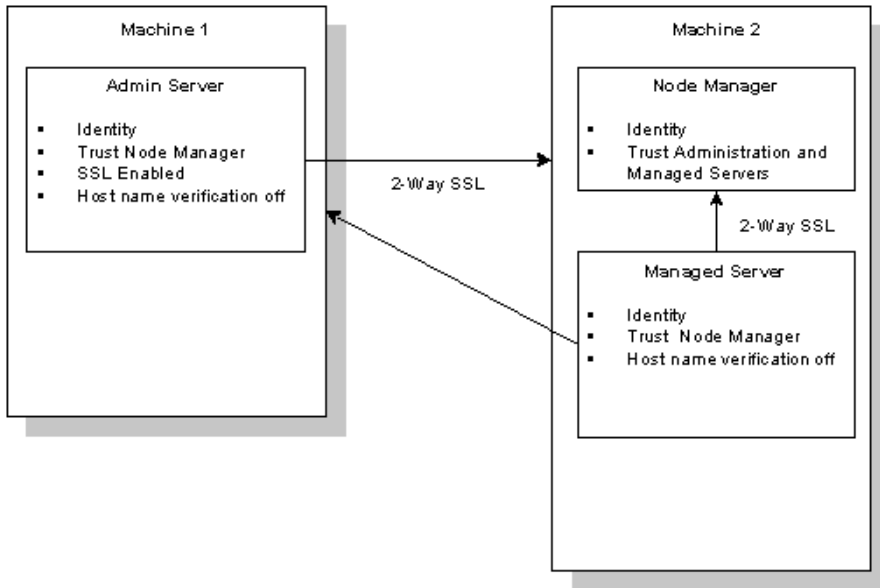
- All digital certificates have host names rather than IP addresses.
- All digital certificates have the correct host name.
- All URLs use host names rather than IP addresses.
- Host name verification is enabled on the Node Manager, the Administration Server, and any Managed Servers.

Node Manager SSL Demonstration Configuration: Main Steps

Using the demonstration identity and trust provided by WebLogic Server to configure SSL for the Node Manager involves verifying the default settings for the SSL attributes and ensuring that the Administration Server and any Managed Servers are listening for SSL communications on different ports. Because the default settings are used with demonstration identity and trust, detailed configuration instructions are not included.

[Figure 6-1](#) illustrates the SSL demonstration configuration.

Figure 6-1 SSL Demonstration Configuration for the Node Manager



To configure the Node Manager to use SSL and the demonstration identity and trust:

1. Run the Configuration Wizard and create a new WebLogic domain.
 - a. Select the Admin Server with Managed Server(s) option.
 - b. Add new Managed Server(s) to the domain. Specify a Listen port for the Managed Server that is different than the listen port of the Administration Server. Set the Listen Address of the Managed Server to `localhost`.

For more information, see [Creating and Configuring WebLogic Server Domains](#).
2. Start the Administration Server. Do not enable the Administration port.
3. Create a machine for the Managed Server. Set the Listen Address to `localhost`. Add the Managed Server to the machine. For more information, see [Configuring a Machine](#).
4. Verify that SSL is enabled for the Administration Server. Because the Administration port is not enabled, SSL must be enabled. Otherwise, the Administration Server will not be able to communicate with the Node Manager.

On the Connections-->SSL Ports tab, verify the Enabled attribute is checked. For more information, see [“Enabling the SSL Port” on page 6-21](#).

5. On the Connections-->SSL tab, verify the settings of the following SSL attributes for the Administration Server:

- Server Key File Name—demokey.pem
- Server Certificate File Name—democert.pem
- Host Name Verification—off

For more information, see [“Setting Attributes for One-Way SSL” on page 6-21](#).

6. In the start script for the Administration Server, verify that trust is set as
`WL_HOME\server\lib\cacerts.`

7. Verify SSL is enabled for the Managed Server. Ensure the SSL port specified is different than the SSL port used by the Administration Server.

On the Connections-->SSL Ports tab, verify the Enabled attribute is checked. For more information, see [“Enabling the SSL Port” on page 6-21](#).

8. On the Connections-->SSL tab, set the SSL attributes for the Managed Server as follows:

- Server Key File Name—full path name to the demonstration private key (`../demokey.pem`) for the domain in which the Managed Server runs.
- Server Certificate File Name—full path name to the demonstration digital certificate (`../democert.pem`) for the domain in which the Managed Server runs.
- Host Name Verification—off

For more information, see [“Setting Attributes for One-Way SSL” on page 6-21](#).

9. Configure start command-line arguments for the Managed Server on the Servers-->Configuration-->Remote Start tab:

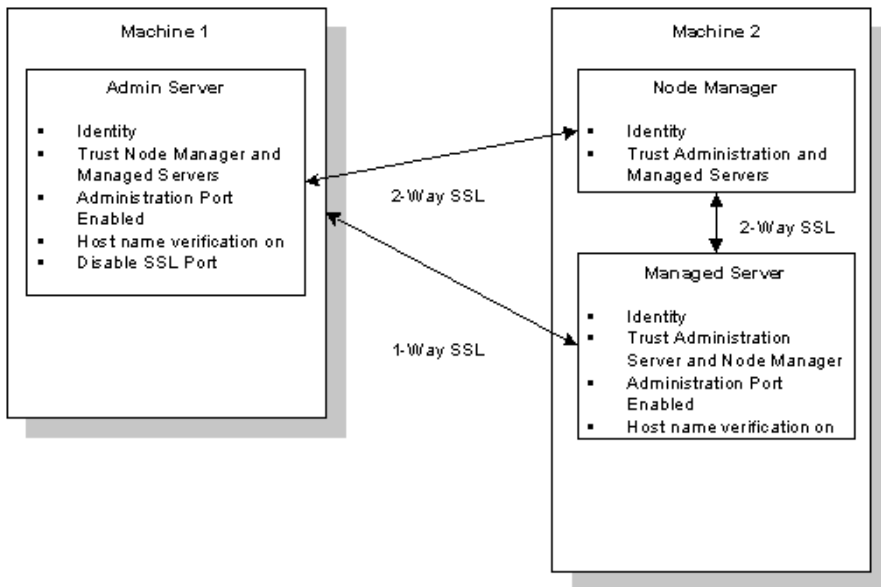
- Username (Required)—Enter the name of a user with privileges to start this Managed Server (for example, `weblogic`).
- Password (Required)—Click change... to change the password used to start the server (for example, `weblogic`). Type the new password in the New Password and Retype to confirm text boxes, then click Apply to apply the change.

10. Stop the Administration Server.
11. Start the Administration Server.

Node Manager SSL Production Configuration: Main Steps

Figure 6-2 illustrates the SSL production configuration.

Figure 6-2 SSL Production Configuration for the Node Manager



To configure SSL for the Node Manager:

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

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

2. Run the Configuration Wizard and create a new WebLogic domain.
3. Start the Administration Server.
4. Configure an Administration port for the WebLogic domain.
5. Create a machine for any Managed Servers. Set the Listen Address to the host name of the machine on which the Managed Server runs. Add the Managed Server to the machine. For more information, see [Configuring a Machine](#).
6. Turn off SSL for the Administration Server. Because the Administration port is enabled, the Node Manager uses this port for all Administration traffic. The Administration port uses SSL to protect all communications therefore you do not need to have an SSL port enabled. Only leave the SSL port enabled if you want to run application traffic over the SSL port.
7. Configure identity for the Administration Server.
8. Turn on host name verification for the Administration Server.
9. Specify the trusted certificate authority for the Administration Server.
10. Enable the Administration port for any Managed Servers.
11. Turn off SSL for the Managed Server.
12. Use remote start arguments to configure identity, trust, and enable host name verification for any Managed Servers.
13. Edit the `nodemanager.hosts` file for the Node Manager. Enter the names of the machines on which the Administration Servers that will be communicating with the Node Manager run. If you want to specify machines by host name rather than by IP address, enable Reverse DNS.
14. Edit the start script for the Node Manager. Specify the location of identity and trust for the Node Manager and enable host name verification.
15. Start the Node Manager.
16. Stop the Administration Server.
17. Start the Administration Server.

Configuring the Administration Server to Use SSL

By default, identity and trust for the Administration Server are configured using demonstration digital certificates and private keys provided by the Configuration Wizard. In addition, SSL is configured by default. For more information, see [“Node Manager SSL Demonstration Configuration: Main Steps” on page 6-34](#). This section describes using the Administration port and production quality identity and trust to configure an Administration Server to use SSL.

To configure the Administration Server to use SSL:

1. Obtain identity and trust for the Administration Server. Ensure the digital certificate for the Administration Server includes a host name.

For more information, see [“Obtaining Private Keys, Digital Certificates and Trusted Certificate Authorities” on page 6-5](#).

2. Run the Configuration Wizard and create a new WebLogic domain.
 - a. Select the Admin Server with Managed Server(s) option.
 - b. Add Managed Server(s) to the domain. Specify a Listen port for the Managed Server that is different than the listen port of the Administration Server. Set the Listen Address of the Managed Server to the hostname of the machine on which the Managed Server runs.

For more information, see [Creating and Configuring WebLogic Server Domains](#).

3. Start the Administration Server.
4. Enable the Administration port for the WebLogic Server domain. For more information, see [Configuring a Domain-Wide Administration Port](#).
5. Turn off SSL by unchecking the Enabled attribute on the Connections-->SSL Ports tab in the WebLogic Server Administration Console.
6. Specify the location of the digital certificate in the Server Certificate File Name attribute on the SSL attributes on the Connections-->SSL tab in the WebLogic Server Administration Console.
7. Create a JKS keystore and store the private key for the Administration Server in the keystore. For more information, see [“Common Keytool Commands” on page 6-17](#).

8. Configure the WebLogic Keystore provider to access the JKS keystore. For more information, see [“Configuring the WebLogic Server Keystore Provider to Locate a Keystore”](#) on page 6-18.
9. Create a JKS keystore for the certificate authorities trusted by the Administration Server. Load the trusted CA certificates in the JKS keystore. For more information, see [“Common Keytool Commands”](#) on page 6-17.
10. Edit the start script for the Administration Server:
 - Use the `-Dweblogic.security.SSL.trustedCAkeystore=path_to_keystore` command-line argument to specify the location of the JKS keystore that contains the trusted certificate authorities for the Administration Server.
 - Use the `-weblogic.security.SSL.ignoreHostnameVerification=false` command-line argument to enable host name verification
11. Configure any Managed Servers. See [“Configuring a Managed Server to Use SSL”](#) on page 6-40.
12. Configure the Node Manager. See [“Configuring the Node Manager to Use SSL”](#) on page 6-42.
13. Start the Node Manager.
14. Stop the Administration Server.
15. Start the Administration Server.

Configuring a Managed Server to Use SSL

By default, no identity is configured for a Managed Server. The demonstration private key and digital certificate provided by the Configuration Wizard can be used to establish identity for a Managed Server, however, this private key and digital certificate should not be used in a production environment. In addition, a Managed Server trusts all the certificate authorities in the `cacerts` files in the `WL_HOME\server\lib` directory by default. For more information, see [“Node Manager SSL Demonstration Configuration: Main Steps”](#) on page 6-34.

This section describes using the Administration port and production quality identity and trust to configure a Managed Server to use SSL.

To configure a Managed Server to use SSL:

1. Obtain identity and trust for the Managed Server. Ensure the digital certificate for the Managed Server includes a host name.
2. Configure the Administration Server in the domain to use SSL. For more information, see [“Configuring the Administration Server to Use SSL” on page 6-39](#).
3. Configure the Administration port on the Managed Server. Specify an Administration port other than the Administration port used by the Administration Server. If the Administration Server runs on the same machine as the Managed Server, set the Local Port Override attribute on the SSL tab to override the Administration Port number.

Since you enabled the Administration port, you must specify trust and host name verification through command-line arguments on the Remote Start tab for the Managed Server.

4. Specify the location of the digital certificate in the Server Certificate File Name attribute on the SSL attributes on the Connections-->SSL tab in the WebLogic Server Administration Console.
5. Create a JKS keystore and store the private key for the Managed Server in the keystore. For more information, see [“Common Keytool Commands” on page 6-17](#).
6. Configure the WebLogic Keystore provider to access the JKS keystore. For more information, see [“Configuring the WebLogic Server Keystore Provider to Locate a Keystore” on page 6-18](#).
7. Create a JKS keystore for the certificate authorities trusted by the Managed Server. Load the trusted CA certificates in the JKS keystore. For more information, see [“Common Keytool Commands” on page 6-17](#).
8. Configure start command-line arguments for Managed Servers on the Servers-->Configuration-->Remote Start tab:
 - Username (Required)—Enter the name of a user with privileges to start this Managed Server (for example, `weblogic`).

- Password (Required)—Click change... to change the password used to start the server (for example, weblogic). Type the new password in the New Password and Retype to confirm text boxes, then click Apply to apply the change.
 - Trust—Use the `weblogic.security.SSL.trustedCAKeyStore` command-line argument to specify the keystore that contains the certificate authorities trusted by the Managed Server. Use the full path name to the keystore.
 - Host Name Verification—Specify the `-Dweblogic.security.SSL.ignoreHostnameVerification=false` command-line argument to enable host name verification.
9. Configure the Node Manager. See [“Configuring the Node Manager to Use SSL” on page 6-42](#).
 10. Start the Node Manager.
 11. Stop the Administration Server.
 12. Start the Administration Server.

Configuring the Node Manager to Use SSL

To configure the Node Manager to use SSL:

1. Configure the Administration Server in the domain to use SSL. For more information, see [“Configuring the Administration Server to Use SSL” on page 6-39](#).
2. Configure any Managed Servers in the domain to use SSL. For more information, see [“Configuring a Managed Server to Use SSL” on page 6-40](#).
3. Obtain identity and trust for the Node Manager. For more information, see [“Obtaining Private Keys, Digital Certificates and Trusted Certificate Authorities” on page 6-5](#).

The Node Manager can only use private keys and digital certificates stored in files.

4. Create a JKS keystore and load the trusted certificate authorities into the keystore. For more information, see [“Common Keytool Commands” on page 6-17](#).

Because 2-way SSL is used, the Node Manager must also trust the certificate authorities used by the Administration Server and any Managed Servers.

5. There is one Node Manager per machine, however, domains managed by the Node Manager can have multiple machines. Steps 4 and 5 must be performed on every machine in the domain managed by the Node Manager.
6. Edit the start script of the Node Manager.

The start script is located in the `WL_HOME\server\bin` directory. The standard Node Manager start script is named `startNodeManager.cmd` on Windows systems and `startNodeManager.sh` on UNIX systems.

Use the following command-line arguments to specify identity and trust:

- 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.
 - Use `weblogic.security.SSL.ignoreHostnameVerification=false` to enable the use of host name verification.
 - Use `weblogic.nodemanager.trustedHosts=pathtofile` to specify the `nodemanager.hosts` file for the Node Manager.
 - Set the `ListenAddress` property to the host name of the machine on which the Managed Server runs.
 - Set the `ReverseDNSEnabled` property to use the host name of an Administration Server rather than an IP address.
7. Edit the `nodemanager.hosts` file located in `WL_HOME\common\nodemanager\config` to include all the Administration Servers on this machine trusted by the Node Manager. 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 `nodemanager.hosts` file defaults to `localhost`.

8. Start the Node Manager.
9. Stop the Administration Server.
10. Start the Administration Server.

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 won't 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 will reject 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 6-4](#) describes the options for the command-line argument.

Table 6-4 Options for -Dweblogic.security.SSL.enforceConstraints

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

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=corbal,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 a download from JDK 1.3. 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. The default JCE provider (SunJCE) provider in the JDK 1.3 has not been tested with this version of WebLogic Server.

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. Hardware accelerators 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 nCipher JCE provider. For more information about the nCipher JCE provider, see <http://www.ncipher.com/solutions/webserver.html>.

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\lib\ext`

For example:

`WL_HOME\jdk131\jre\lib\ext`

UNIX

`JAVA_HOME/lib/ext`

For example:

`WL_HOME/jdk131/jre/libext`

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

Windows NT

`JAVA_HOME\lib\security\java.security`

UNIX

`JAVA_HOME/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.rsa.jca.Provider
```

```
security.provider.3=com.ncipher.provider.km.mCipherKM
```

Note: The JCE providers must be specified in this order to ensure the nCipher JCE providers works properly.

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

Specifying the Version of the SSL Protocol

WebLogic Server supports both the SSL V3.0 and TLS V1.0 protocols. By default, Weblogic Server uses the SSL V3.0 protocol. While in most cases the SSL V3.0 protocol is acceptable there are circumstances (compatibility, SSL performance, and

environments with maximum security requirements) where the TLS V1.0 protocol is desired. The `weblogic.security.SSL.protocolVersion` command-line argument allows you to specify what protocol is used for SSL connections.

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

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

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

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

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

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 will use 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 SSL 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 Server node.
2. Select the server that will be acting as the SSL server.
3. Select the Connections-->SSL tab.
4. Ensure the Check the Client Certificate Enforced attribute is unchecked.
5. Click Apply.
6. 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 digital 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`. However, you have the option of specifying a different trust keystore. Use the following command-line argument to specify a trust keystore other than the JDK trust keystore:

```
-Dweblogic.security.SSL.trustedCAkeystore=pathtokeystore
```

where *pathtokeystore* is the keystore file that contains trusted CA certificates.

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.

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.

Using the SSL Protocol with a BEA Tuxedo Client and WebLogic Server

The SSL protocol can be used to protect communication between a BEA Tuxedo client and WebLogic Server. This section details the steps required to enable the use of the SSL protocol, using the client in the following code example:

```
BEA_HOME/sample/examples/iiop/ejb/stateless/tuxclient
```

To use the SSL protocol and one-way authentication between WebLogic Server and BEA Tuxedo clients:

1. Configure WebLogic Server to use one-way SSL. For more information, see [“Setting Attributes for One-Way SSL” on page 6-21](#).
2. In the Server Certificate File Name attribute in the `config.xml` file for WebLogic Server, specify the name of the file that contains the certificates for WebLogic Server. The order of the certificates in this file is important: the digital certificate for the server must be specified first, followed by the certificate of the certificate authority that issued WebLogic Server’s certificate, followed by the issuer of the certificate authority’s certificate.
3. Reboot WebLogic Server.
4. Obtain an SSL license for the BEA Tuxedo client.
5. Load the certificate for WebLogic Server trusted certificate authority into the BEA Tuxedo Trusted Certificate Authority file (`$TUXDIR/udataobj/security/certs/trust_ca.cer`).
Note: The certificates in this file must be in PEM format.
6. Reboot WebLogic Server.
7. Start the BEA Tuxedo client specifying command-line options that:
 - Enable the use of the SSL protocol.
 - Specify the secure port on which SSL network connections will be accepted.

- Disable peer validation, if the network address at which WebLogic Server is running is not the same as that specified by the domain name in the server's digital certificate.

For example:

```
./Client.exe -ORBid BEA_IIOP -ORBpeerValidate warn\  
-ORBInitRef NameService=corbalocs:iiop:localhost:7002  
/NameService
```


7 Protecting User Accounts

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

- [“Setting Lockout Attributes for User Accounts” on page 7-2](#)
- [“Unlocking a User Account” on page 7-4](#)

Note: For information about protecting user accounts in Compatibility security, see [“Protecting User Accounts in Compatibility Security” on page 9-6](#).

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. Click 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 7-4](#).

5. To save your changes, click Apply.
6. Reboot WebLogic Server.

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

Table 7-1 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	<p>Number of minutes within which invalid login attempts must occur in order for the user's account to be locked.</p> <p>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.</p> <p>The default is 5 minutes.</p>

Table 7-1 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.

Note: The User Lockout attributes apply to the security realm and all its security providers. 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 attribute will not be reset.

Unlocking a User Account

To unlock a user account:

1. Click on the Details link in the table on the Users tab.

The Details tab describes the event that occurred when the user was locked out.

2. Click Unlock.

8 Configuring Security for a WebLogic Domain

The following sections describe how to set security attributes on a WebLogic domain:

- [“Enabling Trust Between WebLogic Domains” on page 8-1](#)
- [“Configuring Connection Filtering” on page 8-3](#)

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 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.

A trust relationship is established when principals in a Subject from one WebLogic Server domain (referred to as the domain) are accepted as principals in the local domain.

This release of WebLogic Server adds more restrictions to the trust relationship between domains. Now a trust relationship is established when the Credential attribute for one domain matches the Credential attribute for another domain.

By default, when you boot an Administration Server for the first time, the Credential attribute is not defined. As the Administration Server boots, it notices that the Credential attribute is not defined and generates a random credential. The Administration Server uses that credential to sign principals in subjects created in that domain. The `config.xml` file which stores the credential is saved after the credential is generated. Managed servers in that domain obtain the credential from the Administration Server when booting.

WebLogic Server performs a validation (comparing how the principal was signed with how a local principal would be signed) whenever the code is asked to create a new subject.

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 7.0 domain, change the Credential attribute in the WebLogic Server 7.0 domain to the password of the `system` user in the WebLogic Server 6.x domain.

If you want two 7.0 domains to interoperate, perform the following procedure in both domains.

To establish a trust relationship between WebLogic Server domains:

1. In the left panel of the console, select the domain name at the top of the tree.
2. Select the Security-->Advanced tab.
3. Uncheck the Enable Generated Credential attribute.
4. Click the Change... link in the Credential attribute.
5. Enter a password for the domain. Choose the password carefully. BEA Systems recommends using a combination of upper and lower case letters and numbers.
6. Confirm the password.
7. Click Apply.
8. Reboot WebLogic Server.

When using inter-domain trust with a WebLogic Server domain that uses custom Principals (meaning a custom Authentication provider is configured in the domain), the domain that is not using custom Principals must have the class for the custom Principal defined in the server's class path in order for authentication to work properly. Otherwise, a `java.lang.ClassNotFound` is thrown.

For example: two domains (Domain 1 and Domain 2) have established trust (meaning their domain credentials are set to the same value).

- Domain 1 has a custom Authentication provider that creates custom Principals of type `myPrincipal`.
- `mySubject` is a Subject authenticated on Domain 1 that contains a Principal of type `myPrincipal`.
- `mySubject` is passed from Domain 1 to Domain 2. Subjects are passed between domains in the following circumstances:
 - When one domain makes an RMI call over T3 to another domain.
 - When one domain makes an RMI call over IIOP and CSIv2 cannot be established.
 - A Subject is passed as a argument to a user's method.
 - When using the JMX Message bridge.
- Domain 2 must have `myPrincipal` defined in the server class path or a `java.lang.ClassNotFound` will be thrown when Domain 2 tries to deserialize the Subject.

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.

WebLogic Server provides a default connection filter called `ConnectionFactoryImpl`. 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. Select the Security-->Filter tab.
3. 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.
4. Enter the syntax for the connection filter rules. For more information about connection filter rules, see [Using Network Connection Filters](#).
5. Click Apply.
6. Reboot WebLogic Server.
7. Expand the Domains node.
8. Click the Security tab.
9. Click the Advanced tab.
10. 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.

11. Click Apply.

9 Using Compatibility Security

The following sections describe how to configure Compatibility security:

- [“Running Compatibility Security: Main Steps” on page 9-1](#)
- [“The Default Security Configuration in the CompatibilityRealm” on page 9-3](#)
- [“Configuring the Identity Assertion Provider in the Realm Adapter Authentication Provider” on page 9-4](#)
- [“Configuring a Realm Adapter Auditing Provider” on page 9-5](#)
- [“Protecting User Accounts in Compatibility Security” on page 9-6](#)
- [“Accessing 6.x Security from Compatibility Security” on page 9-8](#)

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:

9 Using Compatibility Security

1. Make a back-up copy of your 6.x WebLogic domain (including your `config.xml` file) before using Compatibility security. A sample `config.xml` file that can be used to boot Compatibility security can be found in [Booting WebLogic Server in Compatibility Security](#) in the *Upgrade Guide for BEA WebLogic Server 7.0*.
2. Add the following to the 6.x `config.xml` file:

```
<Security Name="mydomain" Realm="mysecurity"/>
<Realm Name="mysecurity" FileRealm="myrealm"/>
<FileRealm Name="myrealm"/>
```
3. Install WebLogic Server 7.0 in a new directory location. Do not overwrite your existing 6.x installation directory. For more information, see the [WebLogic Server Installation Guide](#).
4. Modify the start script for your 6.x server to point to the WebLogic Server 7.0 installation. Specifically, you need to modify:
 - The classpath to point to the `weblogic.jar` file in the WebLogic Server version 7.0 installation.
 - The `JAVA_HOME` variable to point to the WebLogic Server version 7.0 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. Click on your 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 CompatibilityRealm” and “Defining Groups in the CompatibilityRealm” topics in the Compatibility Security online help for the WebLogic Server Administration Console.
 - 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 want to 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 CompatibilityRealm.

For information about configuring a Realm Adapter Authentication provider, see [“Configuring a Realm Adapter Authentication Provider” on page 3-26](#).

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 9-4](#).

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

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. Click the CompatibilityRealm.
3. Expand the Providers node.
4. Click 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-->General tab.

To configure a Realm Adapter Auditing provider:

1. Expand the Compatibility Security-->Realms nodes.
2. Expand the Providers node.
3. Click 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 Compatibility 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.

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

1. Click on the Domains node.
2. Select the Security-->Passwords tab.
3. 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).
4. Click Apply to save your choices.
5. Reboot WebLogic Server.

The following table describes each attribute on the Passwords tab.

Table 9-1 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.

Table 9-1 Password Protection Attributes

Attribute	Description
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 <code>Lockout Reset Duration</code> 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 <code>Lockout Reset Duration</code> attribute. In order to unlock a user account, you need to have the <code>unlockuser</code> permission for the <code>weblogic.passwordpolicy</code> . The default is 30 minutes.
Lockout Reset Duration	<p>Number of minutes within which invalid login attempts must occur in order for the user's account to be locked.</p> <p>An account is locked if the number of invalid login attempts defined in the <code>Lockout Threshold</code> 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.</p> <p>The default is 5 minutes.</p>
Lockout Cache Size	Specifies the intended cache size of unused and invalid login attempts. The default is 5.

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 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

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

